TUGAS AKHIR - IF184802

PENERAPAN POINT CLOUD RECOGNITION PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

MUHAMMAD IVAN RIANSYAH PUTRA
NRP 05111640000093

Dosen Pembimbing
Imam Kuswardayan, S.Kom., MT.
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – IF184802

PENERAPAN POINT CLOUD RECOGNITION PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

MUHAMMAD IVAN RIANSYAH PUTRA
NRP 05111640000093

Dosen Pembimbing
Imam Kuswardayan, S.Kom., MT.
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – IF184802

APPLICATION OF POINT CLOUD RECOGNITION ON NIGHTMARE: ARACHNOPHOBIA GAME

MUHAMMAD IVAN RIANSYAH PUTRA
NRP 05111640000093

Advisor
Imam Kuswardayan, S.Kom., MT.
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

DEPARTMENT of INFORMATICS ENGINEERING
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENERAPAN POINT CLOUD RECOGNITION PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

TUGAS AKHIR

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

Oleh:
MUHAMMAD IVAN RIANSYAH PUTRA
NRP: 05111640000093

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Imam Kuswardayan, S.Kom., MT. .....
NIP. 197612152003121001 (pembimbing 1)

Dr. Eng. Nanik Suciati, S.Kom., M.Kom. .....
NIP. 197104281994122001 (pembimbing 2)

SURABAYA
JUNI 2020

[Halaman ini sengaja dikosongkan]

PENERAPAN POINT CLOUD RECOGNITION PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

Nama Mahasiswa	: Muhammad Ivan Riansyah Putra
NRP	: 05111640000093
Departemen	: Teknik Informatika FTEIC-ITS
Dosen Pembimbing 1	: Imam Kuswardayan, S.Kom., MT.
Dosen Pembimbing 2	: Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

ABSTRAK

Aplikasi permainan merupakan salah satu media hiburan yang paling banyak diminati pada masyarakat modern. Perkembangan pesat teknologi pada bidang aplikasi permainan telah sampai pada tahap dimana pemain merasa apa yang sedang dimainkannya sangat mirip dengan dunia nyata.

Ide yang digunakan dalam tugas akhir ini adalah membangun sebuah permainan First Person Shooter (FPS) yang memiliki genre Survival dengan menerapkan Point Cloud Recognition sebagai kontrol serangan pemain. Permainan ini menjadikan pola gambar yang dibuat pemain sebagai masukan dalam melakukan interaksi. Tujuan dari permainan ini adalah untuk mengasah kemampuan pemain dalam mengambil keputusan. Permainan ini dibangun dengan Unity Versi 2019.2.6f1 dengan bahasa pemrograman C# dan Library Point Cloud Recognition. Asset permainan sebagian besar mengambil dari Asset Store Unity dan sebagian lainnya dibuat menggunakan tools Blender, Corel Draw X8, dan Adobe Photoshop CC 2015.

Hasil dari implementasi ini diuji dengan dua metode yaitu pengujian beta dan pengujian oleh sepuluh orang pengguna. Pengujian beta menguji apakah semua fungsionalitas dalam permainan telah berjalan dengan semestinya. Sedangkan pengujian oleh pengguna bertujuan menguji permainan secara subjektif dengan memberikan kuisioner kepada pengguna. Dengan

hasil dari pengujian beta dan pengujian oleh pengguna dapat disimpulkan permainan Nightmare: Arachnophobia telah mengimplementasikan perancangan dengan baik.

Kata kunci: Aplikasi Permainan, Arachnophobia, Point-Cloud Recognition.

APPLICATION OF POINT CLOUD RECOGNITION ON NIGHTMARE: ARACHNOPHOBIA GAME

Name	: Muhammad Ivan Riansyah Putra
NRP	: 05111640000093
Department	: Informatics Engineering ELECTICS-ITS
Supervisor I	: Imam Kuswardayan, S.Kom., MT.
Supervisor II	: Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

ABSTRACT

Digital Game is one of the most popular entertainment media in modern society. The rapid development of technology in the digital game sector has reached a level where the palyers feel that what they are playing is very similar to the real world.

The idea in this final project is to build a First Person Shooter (FPS) that has Survival genre by implementing Point Cloud Recognition as player control. This game turn the pattern of images made by player into input for game interaction. The purposes of the game is to train the player's ability to make decisions. The game is built with Unity Version 2019.2.6f1 with c# programming language and Point Cloud Recognition Library. The asset of the games mostly taken from Unity Asset Store and some others are made using tools Blender, Corel Draw X8, and Adobe Photoshop CC 2015.

The results of this implementation were tested by two methods, beta testing and user testing by ten users. Beta testing tests whether all functionality in the game is working properly. While testing by users aims to test the game subjectively by giving questionnaires to users. With the results of beta testing and user testing it can be concluded the Nightmare: Arachnophobia game has implemented the design properly.

Keywords: Digital Game, Arachnophobia, Point-Cloud Recognition.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah Subhanahu Wata'ala karena atas karunia, nikmat dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

PENERAPAN POINT CLOUD RECOGNITION PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

Sholawat serta salam semoga tak lupa kita sampaikan kepada Rasulullah Shallallahu 'Alaihi Wassalam, yang telah membawa risalah Islam dan Qur'an kepada seluruh umat manusia.

Melalui lembar ini, penulis ingin menyampaikan ucapan terima kasih dan penghormatan yang sebesar-besarnya kepada:

1. Orang tua dan keluarga yang senantiasa mendoakan, memotivasi dan mendukung lahir maupun batin penulis dalam menyelesaikan tanggung jawab ini.
2. Bapak Imam Kuswardayan, S.Kom., MT. dan Ibu Dr. Eng. Nanik Suciati, S.Kom, M.Kom. selaku dosen pembimbing I dan II penulis, yang senantiasa membimbing, memberikan saran, arahan, serta bantuan-bantuan lainnya dalam menyelesaikan tugas akhir ini.
3. Teman-teman TC 2016 yang banyak membantu penulis dalam segala hal dan berjuang bersama-sama penulis dari awal masuk perkuliahan hingga akhir masa perkuliahan.
4. Teman-teman group main yang senantiasa membantu, mendukung, menghibur, dan menemani keseharian penulis selama masa perkuliahan.
5. Teman-teman kontrakan barokah yang selalu memberikan tempat berkumpul di kontrakan walaupun penulis bukan penghuni kontrakan tersebut.
6. Teman-teman Discord server Random RTO yang selalu menemani penulis secara online dan menjadi teman diskusi yang baik.

7. User TA IGS lainnya yang berjuang bersama penulis dalam menyelesaikan tugas akhir ini.
8. Mas Dias Adhi Pratama yang sudah berkenan memberikan referensi, penjelasan, dan bimbingan tentang pengerjaan tugas akhir ini.
9. Serta pihak-pihak lain yang mohon maaf tidak dapat disebutkan satu per satu, yang telah turut andil dalam membantu penulis selama perkuliahan.

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyelesaikan tugas akhir ini. Namun, penulis mohon maaf apabila terdapat kekurangan ataupun kesalahan yang penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan untuk ke depannya.

Surabaya, Juni 2020

Muh. Ivan Riansyah P.

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Aplikasi permainan merupakan salah satu media hiburan yang paling banyak diminati pada masyarakat modern. Permainan digital menjadi kegiatan yang dipilih untuk hiburan atau sekadar menghabiskan waktu luang. Dewasa ini sering sekali muncul perkembangan teknologi baru terkait aplikasi permainan. Perkembangan pesat teknologi pada bidang aplikasi permainan telah sampai pada tahap dimana pemain merasa apa yang sedang dimainkannya sangat mirip dengan dunia nyata [1].

Selain untuk hiburan, aplikasi permaianan yang memiliki grafis memukau selayaknya dunia nyata dapat digunakan untuk berbagai tujuan lainnya, salah satunya untuk media terapi fobia. Arachnophobia atau fobia laba-laba adalah gangguan kecemasan yang mengakibatkan seseorang mengalami rasa takut berlebih pada hewan laba-laba. Secara umum, mungkin banyak orang yang lebih memilih untuk menghindari laba-laba karena merasa jijik, takut digigit, hingga dianggap beracun. Berdasarkan data dari YouGov, Sebanyak 18% populasi dunia mengaku sangat takut, sedangkan 24% populasi menyatakan agak takut [2].

Dalam hal ini penulis ingin membuat sebuah aplikasi permainan yang dapat memberikan pengalaman bermain baru kepada para pemain sekaligus menjadi salah satu media untuk membantu proses terapi fobia laba-laba secara menyenangkan. Oleh karena itu dalam Tugas Akhir ini penulis akan membuat permainan digital yang berjudul Nightmare: Arachnophobia yang menerapkan Point Cloud Recognition sebagai sistem kontrol pemain. Permainan ini merupakan sebuah permainan *First Person Shooter* dengan genre *survival* game.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana merancang aturan main aplikasi permainan Nightmare: Arachnophobia dengan konsep 3D First Person Shooter (FPS)?
2. Bagaimana merancang tingkat kesulitan dan skenario untuk story mode pada permainan Nightmare: Arachnophobia?
3. Bagaimana merancang mekanisme penyerangan pemain dengan menerapkan Point Cloud Recognition untuk mengenali bentuk pola garis yang digambar oleh pemain?
4. Bagaimana implementasi dari rancangan di atas diselesaikan dengan menggunakan Game Engine Unity ?
5. Bagaimana metode pengujian dan evaluasi dari aplikasi permainan Nightmare: Arachnophobia ?

1.3. Batasan Permasalahan

Batasan masalah pada tugas akhir ini antara lain:

1. Aplikasi permainan yang dibuat merupakan permainan yang berjalan pada perangkat *Personal Computer* (PC) dengan sistem operasi Windows.
2. Lingkungan pengembangan yang digunakan adalah Visual Studio 2019 dengan Bahasa pemrograman yang digunakan adalah C#.
3. Point Cloud Recognition yang digunakan adalah $P-Recognizer.
4. Pola yang dikenali adalah hasil gambar dari pemain menggunakan *pointer mouse*.

1.4. Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk membuat aplikasi permainan Nightmare: Arachnophobia yang menerapkan Point Cloud Recognition sebagai kontrol pemain dalam melakukan serangan. Permainan Nightmare: Arachnophobia adalah permainan pada perangkat *Personal Computer* (PC) dengan sistem operasi Windows.

1.5. Manfaat

Manfaat dari tugas akhir ini adalah sebagai berikut:

1. Memberikan pengalaman bermain baru dengan menerapkan Point Cloud Recognition sebagai mekanisme penyerangan pada permainan *First Person Shooter*.
2. Mengasah kemampuan pengambilan keputusan pemain untuk memenangkan permainan.
3. Sebagai media hiburan untuk para pemain.
4. Membantu proses terapi fobia laba-laba dengan menggunakan laba-laba virtual sebagai media simulasi.

1.6. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Studi literatur

Pada studi literatur, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi permainan yaitu mengenai bahasa pemrograman C#, *game engine* Unity3D, *Survival Game*, *First Person Shooter* (FPS), Blender, dan Point Cloud Recognition.

2. Analisis dan desain sistem

Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap analisis dan perancangan mendefinisikan alur dari pembuatan aplikasi permainan ini serta langkah-langkah yang akan dikerjakan. Selain itu, dalam tahap ini juga akan dilakukan desain dari sistem dan proses-proses yang ada.

3. Implementasi sistem

Dalam menyelesaikan tugas akhir ini, aplikasi akan diimplementasikan menggunakan IDE Visual Studio dengan *game engine* Unity3D dan bahasa pemrograman C#.

4. Pengujian dan evaluasi

Tahap pengujian dan evaluasi berisi pengujian aplikasi dan evaluasi berdasarkan hasil pengujian. Pada tahap ini dilakukan pengujian dari fungsionalitas perangkat lunak, apakah aplikasi sudah berjalan sesuai yang diinginkan atau terdapapat masalah. Tahapan ini dimaksudkan untuk mengevaluasi aplikasi permainan dan melakukan perbaikan jika ditemukan kesalahan.

5. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
   a. Latar Belakang
   b. Rumusan Masalah
   c. Batasan Tugas Akhir
   d. Tujuan
   e. Metodologi
   f. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.7. Sistematika Penulisan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini:

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

Bab II **Tinjauan Pustaka**
Bab ini menjelaskan beberapa pustaka-pustaka yang dijadikan penunjang dan berhubungan dengan pokok pembahasan yang mendasari pembuatan tugas akhir.

Bab III **Analisis dan Perancangan Sistem**
Bab ini membahas mengenai analisis dan perancangan sistem yang akan dibangun.

Bab IV **Implementasi Sistem**

Bab ini membahas mengenai bagaimana implementasi sistem dari analisis dan desain yang sudah dirancang.

Bab V **Pengujian dan Evaluasi**

Bab ini membahas pengujian dari metode yang ditawarkan dalam tugas akhir untuk mengetahui kesesuaian metode dengan data yang ada.

Bab VI **Kesimpulan dan Saran**

Bab ini berisi kesimpulan dari hasil pengujian yang telah dilakukan. Bab ini juga membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi data atau daftar istilah yang penting pada tugas akhir ini.

BAB II
TINJAUAN PUSTAKA

Bab ini membahas pustaka/teori-teori yang menjadi dasar dalam pembuatan tugas akhir.

2.1. Aplikasi Permainan

Pengertian dari permainan sendiri menurut David Parlett, adalah “sesuatu yang memiliki akhir dan cara mencapainya”, artinya sesuatu yang memiliki tujuan, hasil, dan serangkaian peraturan untuk mencapai keduanya. Sedangkan menurut Sadiman, Permainan adalah setiap kontes antara pemain yang berinteraksi satu sama lain dengan mengikuti aturan-aturan tertentu untuk mencapai tujuan tertentu pula. Jadi permainan adalah cara bermain dengan mengikuti aturan-aturan tertentu yang dapat dilakukan secara individu maupun berkelompok guna mencapai tujuan tertentu [3].

Aplikasi permaian adalah permainan interaktif yang diproyeksikan dan divisualisasikan secara digital dalam bentuk aplikasi. Terdapat banyak media untuk menjalankan aplikasi permainan seperti *Personal Computer* (PC), *Game Console, Smartphone,* maupun perangkat lainnya yang dikhususkan sebagai media bermain aplikasi permainan. Aplikasi permainan atau bisa disebut sebagai permainan digital memiliki konsep dimana pemain memberikan masukkan (*Input*) terhadap perangkat, kemudian akan diproses oleh *Processor* sehingga menghasilkan suatu keluaran (*Output*) yang akan ditampilkan oleh perangkat.

Terdapat banyak jenis dari aplikasi permainan berdasarkan genrenya seperti *Action, Role Playing Game, Adventure, Puzzle, Simulation, Sports, Strategy,* dan masih banyak lainnya. Munculnya banyak jenis aplikasi permainan ini diakibatkan oleh beragamnya permintaan pasar. Permintaan yang beragam ini merupakan bukti bahwa penikmat aplikasi permainan tidak terbatas pada anak-anak saja, namun semua usia.

2.2. Unity 3D

Unity3D adalah salah satu *game engine* lintas *platform* yang dikembangkan oleh Unity Technologies dimana pertama kali diumumkan dan dirilis ke publik pada Juni 2005 di konferensi Apple Inc sebagai *game engine* eksklusif pada sistem operasi Mac OS X. Game engine ini dapat digunakan untuk membuat aplikasi permainan 2-dimensi, 3-dimensi, realitas virtual, realitas teraugmentasi dan juga simulasi. *Game engine* ini juga telah diadopsi diluar aplikasi permainan yaitu film, otomotif, arsitektur, rekayasa, dan konstruksi [4].

Untuk membuat suasana permainan terasa lebih nyata, digunakan fitur Post Processing Stack miliki Unity. Post Processing adalah sebuah proses yang mengaplikasikan *full-screen filters* dan efek-efek pada *buffer* gambar kamera sebelum akhirnya ditampilkan pada layar. Penggunaan Post Processing ini dapat meningkatkan secara drastis visual dari permainan dengan waktu pengaturan yang singkat [5].

2.3. Bahasa Pemrograman C#

C# (dibaca: c sharp) merupakan sebuah bahasa pemrograman berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari insiatif kerangkan .NET *framework*. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic dan lain-lain dengan beberapa penyederhanaan [6].

2.4. First Person Shooter

First Person Shooter (FPS) adalah permainan digital yang ciri utamanya adalah penggunaan sudut pandang orang pertama dengan tampilan layar yang mensimulasikan apa yang dilihat melalui mata karakter yang dimainkan. Ciri lain dari *game* dengan genre FPS adalah penggunaan senjata genggam jarak jauh.

Pengembangan permainan *First Person Shooter* (FPS) dimulai pada tahun 1973 dan 1974 yaitu *game* Spasim. Setelah itu, lebih banyak judul bermunculan seperti MIDI Maze tahun 1987, Wolfenstein 3D tahun 1992, dan Doom tahun 1993. Kemudian mulai berkembang lagi pada permainan Half Life (1998) dan Half Life 2 (2004) yang meningkatkan narasi dan elemen teka-teki [7].

2.5. Survival Game

Permainan bertahan hidup atau *Survival Game* salah satu sub-genre dari *Action Game* dengan pemain berada di dalam lingkungan dunia terbuka yang intens dimana pemain umumnya memulai permainan dengan peralatan seadanya dan diminta untuk mengumpulkan sumber daya, senjata, tempat berlindung, serta bertujuan bertahan selama mungkin. *Survival Game* sering kali berkaitan erat dengan tema horor, dimana pemain harus bertahan hidup dari bencana kiamat seperti serangan zombi. Contoh *Survival Game* yang juga menggunakan *First Person Shooter* adalah Left 4 Dead (2008) dan Left 4 Dead 2 (2009) [8].

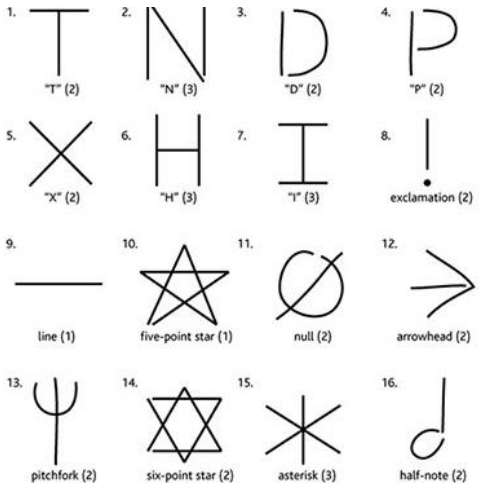
2.6. Blender

Blender adalah aplikasi gratis dan *open source* untuk pemodelan 3D. Blender mendukung seluruh fitur dari pemodelan 3D seperti pemodelan, pemasangan, animasi, simulasi, *rendering*, penggabungan, *motion tracking*, bahkan pengeditan video dan pembuatan game. Blender sangat cocok digunakan untuk pengguna pribadi dan studio kecil karena akan mendapatkan manfaat dari fitur-fitur Blender yang mendukung proses pengembangan aplikasi yang responsif [9].

2.7. $P Point-Cloud Recognizer

Point Cloud sendiri adalah kumpulan data dari titik-titik yang memiliki koordinat pada sebuah ruang. *Point Cloud* dapat berupa titik pada ruang dua dimensi maupun tiga dimensi. $P Point-Cloud Recognizer adalah sebuah *gesture recognizer* 2-D yang didesain sebagai *prototype* dari *user interface* gerakan pengguna. Pada konsep *machine learing* $P adalah sebuah instansi yang berdasarkan *nearest-neighbor classifier* dengan sebuah fungsi Euclidean sebagai nilainya. $P adalah *recognizer* paling baru dari keluarga algoritma dollar yang dimana terdapat $1 dan $N yang sudah ada lebih dahulu. Walapun sebagian besar dari kode $P diambil dari $1, namun $P dapat merepresentasikan bentuk pola garis sebagai *point-clouds* yang tidak berurutan. Dikarenakan hal ini, $P dapat menangani *unistrokes* dan *multistrokes* secara seimbang [10].

Cara kerja $P Point-Cloud Recognizer adalah menghubungkan antara hasil gambar dari *Line Renderer* yang dibuat lalu menyimpan titik-titik koordinatnya. Titik-titik koordinat tadi disimpan pada sebuah *Array List*. Kemudian hasil *Array List* tersebut dicocokkan dengan *Array List* milik Data set yang sudah tersedia. Penentuan bentuk dari pola yang dibuat pengguna didasarkan pada seberapa mirip *Array List* pola dengan yang ada pada data set. Contoh bentuk pola garis atau gestur yang sudah ada pada data set dan dapat dikenali oleh $P Point-Cloud Recognizer dapat dilihat pada gambar 2.1. Selain betuk pola yang ada pada data set, pengguna juga dapat menambahkan bentuk pola sendiri dengan cara meggunakan *interface* yang sudah disediakan pada *Demo Scene.*



Gambar 2.1 Contoh Bentuk Pola atau Gestur

Alasan dipilihnya algoritma $P Point-Cloud Recognizer pada permainan Nightmare: Arachnophobia adalah karena kecocokan algoritma ini dengan mekanisme penggambaran bentuk garis oleh pemain. Pada permainan ini mekanisme penggambaran bentuk garis dilakukan menggunakan perangkat *mouse* sehingga pola bentuk yang dapat dibuat pun terbatas, karenanya hanya bentuk-bentuk sederhana dua dimensi yang mudah digambar. $P Point-Cloud Recognizer adalah *recognizer* 2-D yang dapat menangani baik *unistrokes* maupun *multistroke*. Hal ini memungkinkan pemain untuk bebas menggambar pola dengan jumlah goresan yang banyak. Selain itu beban komputasi $P Point-Cloud Recognizer jauh lebih ringan daripada *recognizer* lainnya karena hanya menangani bentuk dua dimensi.

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tentang analisis dan perancangan Sistem aplikasi permainan Nightmare: Arachnophobia. Pembahasan yang dilakukan meliputi analisis sistem, perancangan sistem, skenario simulasi, dan perancangan antar muka sistem.

3.1. Analisis Sistem

Sub bab ini akan membahas tentang analisis kebutuhan sistem, meliputi spesifikasi kebutuhan sistem, baik itu kebutuhan fungsional sistem maupun kebutuhan non-fungsional sistem, dan identifikasi pengguna sistem.

3.1.1. Spesifikasi Kebutuhan Sistem

Pada sistem ini terdapat beberapa kebutuhan fungsional dan kebutuhan non-fungsional yang mendukung berjalannya sistem. Kebutuhan fungsional sistem dapat dilihat pada Tabel 3.1, sedangkan kebutuhan non-fungsional sistem dapat dilihat pada Tabel 3.2.

Tabel 3.1 Kebutuhan Fungsional Sistem

Kode	Deskripsi
F1	Pemain dapat melihat *Main Menu*
F2	Pemain dapat melihat *Quick Play Menu*
F3	Pemain dapat melihat *How to Play Menu*
F4	Pemain dapat keluar dari permainan
F5	Pemain dapat bergerak menggunakan kontrol *Mouse* dan *Keyboard*
F6	Pemain dapat menyerang musuh
F7	Pemain memiliki *Health Point* yang dapat berkurang
F8	Pemain memiliki *Mana Point* yang dapat digunakan dan juga dapat beregenerasi

F9	Pemain dapat meggambar pola menggunakan kontrol *Mouse*
F10	*Point-Cloud Gesture Recognizer* dapat mendeteksi bentuk garis yang dibuat oleh pemain
F11	Pemain dapat mengaktifkan *Weapon Skill* sesuai dengan bentuk pola garis yang digambar
F12	Musuh dapat mengejar dan menyerang pemain
F13	Musuh memiliki *Health Point* yang dapat berkurang
F14	Pemain dapat memulai kembali permainan jika pemain mati atau kalah
F15	Pemain dapat memenangkan permaian dengan mengalahkan *Boss* pada babak terakhir

Tabel 3.2 Kebutuhan Non-Fungsional Sistem

Kode	Deskripsi
NF1	Sistem dapat dijalankan pada perangkat komputer
NF2	Sistem dapat dijalankan pada Sistem Operasi Windows 10
NF3	Sistem memiliki tampilan antar muka yang mudah dipahami
NF4	Sistem menggunakan bahasa Inggris

3.1.2. Identifikasi Pengguna

Pengguna yang dapat menggunakan permainan Nightmare: Arachnophobia ini adalah siapa saja (umum). Sehingga, pengguna berhak menggunakan seluruh fungsionalitas yang terdapat pada sistem.

3.2. Perancangan Sistem

Sub bab ini membahas tentang bagaimana sistem ini dirancang, meliputi deskripsi umum sistem dan arsitektur sistem.

3.2.1. Deskripsi Umum Sistem

Nightmare: Arachnophobia merupakan aplikasi permainan berbasis *personal computer* (PC) dengan genre *First Person Shooter* (FPS) dan *Survival Game*. Nightmare: Arachnophobia menggunakan perangkat *Mouse* dan *Keyboard* sebagai kendalinya. *Point Cloud Recognition* diimplementasikan dalam permainan ini untuk mengenali bentuk pola garis yang digambar pemain sebagai aktivasi dari *Weapon Skill*. Sistem ini merupakan aplikasi permainan komputer yang menghadirkan atmosfer dan suasana fantasi seperti terjebak dalam mimpi buruk.

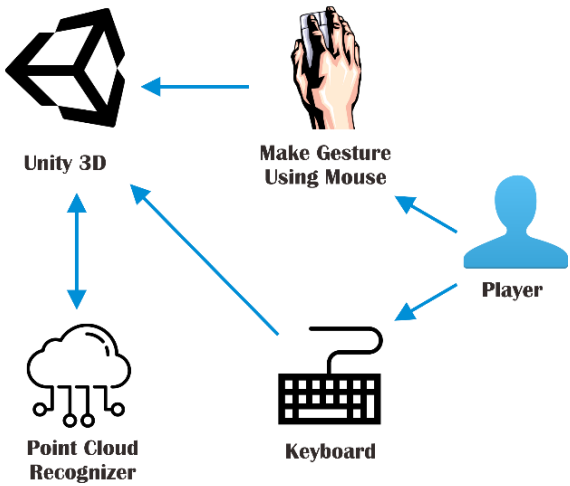
Pembangunan sistem ini dimulai dari membuat *terrain* yang bertema fantasi yang akan dijadikan sebagai lingkungan nyata di dalam permainan ini. Proses pembangunan sistem selanjutnya yaitu dengan merancang *gameplay* yang sesuai dengan tema dari permainan ini. Mencari aset-aset yang sesuai, dan melengkapi aset dengan mendesainnya secara manual.

3.2.2. Arsitektur Sistem

Nightmare: Arachnophobia berjalan dengan basis *Personal Computer* (PC), dengan *Mouse* dan *Keyboard* sebagai perangkat kendalinya. Perintah dari *Mouse* dan *Keyboard* diproses oleh *Processor* dan ditampilkan hasilnya di monitor komputer. Dengan *Point-Cloud Gesture Recognizer*, permainan Nightmare: Arachnophobia dapat mendeteksi bentuk pola garis yang digambar oleh pemain. Menggunakan *Pointer Mouse* sebagai media untuk menggambar, permainan ini dapat menentukan pola bentuk apa yang digambar oleh pemain. Fitur *Gesture Recognition* ini membuat permainan Nightmare: Arachnophobia menawarkan suatu pengalaman dan cara bermain baru pada pemain permainan *First Person Shooter* (FPS).

Alur sistem permainan Nightmare: Arachnophobia adalah sistem menerima *input* dari pemain menggunakan *Mouse* dan *Keyboard*. *Input* dari *Keyboard* berupa gerakan pemain, sedangkan *input* dari *Mouse* berupa gerakan kamera, menyerang, dan juga menggambar pola untuk mengaktifkan *Weapon Skill*. *Input* gambar dari *Mouse* disimpan dalam bentuk garis yang tersusun atas titik-titik koordinat (x,y), lalu diproses oleh *Point-Cloud Gesutre Recognizer* untuk dijadikan data. Hasil dari pemrosesan data tersebut dibandingkan dengan *Dataset* yang sudah disediakan sebelumnya. *Dataset* juga berupa kumpulan titik-titik koordinat (x,y) yang membentuk gambar garis. Setelah dibandingkan, *Point-Cloud Gesutre Recognizer* lalu mengeluarkan hasil dalam bentuk *string* dari nama bentuk pola yang sesuai dan juga *score* hasil perbandigannya. Jika bentuk pola yang terdeteksi sesuai dengan pola yang dibutuhkan untuk aktivasi dari *Weapon Skill*, maka *skill* dari senjata akan aktif.

Pada Gambar 3.1 ditampilkan diagram dari arsitektur sistem permainan Nightmare: Arachnophobia.



Gambar 3.1 Arsitektur Nightmare: Arachnophobia

3.3. Perancangan *User Interface* dan Aset Permainan

Pada sub bab ini menjelaskan tentang user interface dan aset yang terdapat dalam permainan. Selain itu akan dibahas pula penggunaannya pada permainan.

3.3.1. *User Interface*

User interface pada permainan ini antara lain:

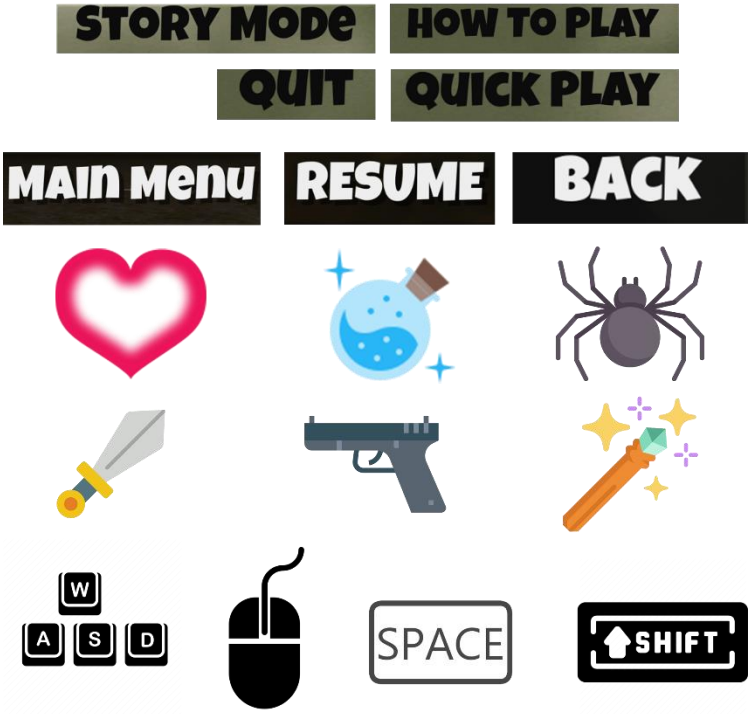
1. *Panel*

Panel digunakan pada *Main Menu*, *How To Play*, *Pause Menu*, *Next Level*, *Final Scene*, dan *Game Over*. Gambar 3.2 merupakan tampilan *panel* yang digunakan.

Gambar 3.2 Panel yang Diginakan pada Permainan

2. *Button* dan *Icon*

Button digunakan pada tiap halaman yang ada dalam permainan sebagai tombol navigasi antar *scene*. *Icon* digunakan sebagai estetika tampilan permainan dan juga sebagai petunjuk dalam permainan. Gambar 3.3 merupakan tampilan dari *Button* dan *Icon* yang digunakan.



Gambar 3.3 Button dan Icon yang Digunakan dalam Permainan

3.3.2. Aset Permainan

Aset yang digunakan pada permainan Nightmare: Arachnophobia ini antara lain:

1. *Environment*

Environment digunakan pada setiap *stage* permainan yang ada dalam permainan. Pada permainan ini terdapat *terrain* dengan fase siang dan malam hari, kecuali pada *stage* Castle1 dan Castle2. Pada *environment* ini digunakan aset gratis dari Unity Store yaitu "Medieval Buildings", "Flat Car Kit", dan "Cartoon Castle Building Kit". Gambar 3.4 merupakan tampilan dari *environment* yang digunakan.

Gambar 3.4 Aset Gaia Terrain Maker

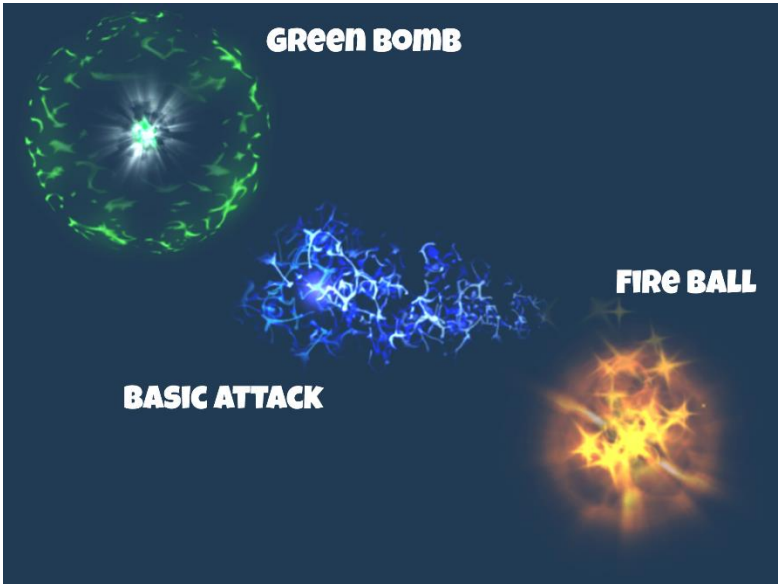
2. Karakter

Aset yang digunakan pada pembuatan permainan ini menggunakan aset gratis dari “free3d.com” yang dibuat oleh akun “3dhaupt”. Namun karakter laba-laba yang disediakan oleh aset gratis ini hanya satu. Untuk memenuhi rancangan jumlah karakter musuh, penulis memodifikasi *material* yang ada pada *3D model* laba-laba ini. Gambar 3.5 menampilkan karakter musuh yang digunakan pada permainan ini.

Gambar 3.5 Aset Character (Laba-laba)

3. *Particle Effect*

Particle Effect digunakan sebagai efek visual dari sihir atau *Spell* pada senjata *Scepter*. Visual efek ini bertujuan untuk menambah estetika dari permainan Nightmare: Arachnophobia. Pada pembuatan permainan ini digunakan aset gratis dari Unity Store yaitu "KY Magic Effects". Sihir dalam yang dapat digunakan adalah *Lightning ball* (*Basic Attack*), *Green Bomb*, dan *Fire Ball*. Gambar 3.6 merupakan tampilan dari *particle effects* yang digunakan.



Gambar 3.6 Aset Particle Effect

4. Senjata

Pada permainan Nightmare: Arachnophobia terdapat tiga senjata berbeda yang dapat digunakan oleh pemain. Senjata-senjata tersbut adalah pedang, pistol, dan tongkat sihir. Aset senjata ini merupakan aset gratis dari “Mixamo.com”. Gambar 3.7 menampilkan aset senjata yang digunakan.

Gambar 3.7 Aset Senjata

3.4. Perancangan Skenario Permainan

Pada sub bab ini menjelaskan tentang skenario permainan untuk menentukan kondisi menang atau kalah. Selain itu akan dibahas pula aturan permainan dan mekanisme permainan.

3.4.1. Alur Permainan

Alur permainan dari permainan Nightmare: Arachnophobia antara lain:

1. Saat permainan dijalankan maka pemain akan melihat Menu Utama yang memiliki 4 tombol interaksi yaitu, tombol *Story Mode*, *Quick Play*, *How to Play*, dan *Quit*.
2. Jika pemain menekan tombol *Quick Play*, maka akan muncul menu untuk memilih *stage* mana yang akan dimainkan. Pemain bisa langsung bermain pada *stage* tersebut tanpa harus memainkan *stage* sebelumnya.
3. Jika pemain menekan tombol *How to Play*, maka akan muncul petunjuk cara bermain dan informasi mengenai permainan.
4. Jika pemain menekan tombol *Quit*, maka pemain akan keluar dari permainan dan permainan akan tertutup.
5. Untuk mulai bermain, pemain menekan tombol *Story Mode.*
6. Pemain akan ditempatkan pada titik *spawn* awal di tengah *map* dalam keadaan *Health Point* pada kondisi penuh dan *Mana Point* kosong.
7. *Health Point* akan berkurang jika pemain diserang oleh musuh, sedangkan *Mana Point* beregenerasi setiap detiknya.
8. Pemain dapat bergerak bebas di dalam ruang lingkup *map.*
9. Saat permainan berlangsung, musuh akan muncul dan mengejar pemain.
10. Musuh akan menyerang pemain jika bersentuhan dengan pemain.
11. Pemain diharuskan untuk mengalahkan musuh sebelum musuh dapat menyentuh pemain. Pemain dapat menyerang dengan *basic attack* maupun *weapon skills.*
12. Untuk menggunakan *weapon skills* pemain harus memiliki *Mana Point* yang penuh. *Weapon Skills* dapat diaktifkan dengan cara menggambar pola yang tepat untuk jenis senjata yang sedang digunakan.
13. Saat permainan berlangsung, pemain dapat menekan tombol *escape* pada *keyboard* untuk menjeda permainan dan menampilkan *Pause Menu*.

14. Pada *Pause Menu* pemain dapat memilih tombol *Resume, Main Menu,* dan *Quit.* Tombol *Resume* untuk melanjutkan permainan, tombol *Main Menu* untuk kembali ke halaman utama, dan tombol *Quit* untuk keluar dari permaian.
15. Pemain dikatakan kalah jika *Health Point* mencapai nol.
16. Pemain dapat memilih tombol *Restart* untuk mengulang permainan pada *stage* saat ini atau tombol *Quit* untuk keluar dari permainan.
17. Pemain dikatakan menang jika pemain berhasil mengalahkan semua laba-laba dan bertahan hingga babak terakhir.

3.4.2. Aturan Main

Dalam memainkan permainan Nightmare: Arachnophobia ada beberapa aturan main sebagai berikut:

1. Tujuan permainan adalah untuk bertahan hidup hingga akhir pada seluruh babak dengan cara mengalahkan semua laba-laba yang ada.
2. Setiap babak memiliki dua fase yaitu siang dan malam.
3. Pada fase siang pemain harus membunuh semua *creep* laba-laba.
4. Pada fase malam, pemain akan dihadapkan dengan *Boss* pada babak tersebut. Pemain dapat melanjutkan ke babak berikutnya setelah mengalahkan *Boss.*
5. Pemain mendapatkan senjata yang berbeda pada setiap babak. Terdapat tiga senjata yaitu Pedang, Pistol, dan Sihir.
6. Setiap senjata memiliki *Weapon Skills* masing-masing yang diaktifkan dengan cara menggambar pola. Pemain dapat menggambar garis dengan menekan tombol *Mouse* sebelah kanan.
7. Pemain dapat bergerak ke segala arah, namun terbatas pada *map* pada babak itu saja.
8. Permainan akan berakhir jika *Health Point* pemain mencapai nol.

3.4.3. Mekanisme Permainan

1. Pemain

- Pemain memiliki *Health Point* yang apabila terkena serangan maka akan berkurang. Jumlah *Health* pemain adalah 100.
- Pemain memiliki *Mana Point* yang akan terisi 5 poin setiap detiknya dengan nilai maksimal adalah 100 poin.
- Pemain bergerak dengan tombol *keyboard* W, A, S, dan D. Serta tombol *Shift* untuk lari.
- Kecepatan gerak pemain adalah 4. Sedangkan saat berlari adalah 6.
- Pemain dapat menyerang dengan menggunakan tombol *mouse* sebelah kiri (LMB).
- Pemain dapat mengaktifkan *Weapon Skill* jika *Mana Point* telah penuh dan pemain menekan tombol *mouse* sebelah kanan (RMB).
- Pemain mengkonfirmasi gambar dengan menekan tombol *Space* pada *keyboard*. Kemudian bentuk pola garis akan dikenali oleh *Point-Cloud Gesture Recognizer*.

2. Senjata

a. Pedang

- Memiliki *damage* dasar 100 point.
- Memiliki jarak serangan 10.
- Memiliki kecepatan serangan 2 detik untuk sekali serang.
- Memiliki dua *Weapon Skills* yaitu *Damage Up* yang meningkatkan *damage* menjadi 200 poin dan *Rapid Slash* yang meningkatkan kecepatan serang menjadi 1 detik.
- Pola yang mengaktifkan *Damage Up* adalah gestur ‘Z’.
- Pola yang mengaktifkan *Rapid Slash* adalah gestur ‘V’.
- *Weapon skills* berdurasi selama 15 detik, lalu kembali kepada kondisi serangan dasar.

b. Pistol
- Memiliki *damage* dasar 50 poin.
- Memiliki jarak serangan 200.
- Memiliki kecepatan serangan 1 detik untuk sekali serang.
- Memiliki dua *Weapon Skills* yaitu *Damage Up* yang meningkatkan *damage* menjadi 100 poin dan *Rapid Fire* yang meningkatkan kecepatan serang menjadi 0,5 detik.
- Pola yang mengaktifkan *Damage Up* adalah gestur 'S'.
- Pola yang mengaktifkan *Rapid Fire* adalah gestur 'M'.
- *Weapon skills* berdurasi selama 15 detik, lalu kembali kepada kondisi serangan dasar.

c. Sihir
- Memiliki *damage* dasar 100 poin.
- Memiliki jarak serangan 200.
- Memiliki kecepatan serangan 1 detik untuk sekali serang.
- Memiliki kecepatan gerak *projectile* 20.
- Memiliki dua *Weapon Skills* yaitu *Green Bomb* (kecepatan *projectile* : 40 dan *damage* 100) dan *Fire Ball* (kecepatan *projectile* : 10 dan *damage* 200)
- *Projectile* setiap serangan memiliki efek visual yang berbeda.
- Pola yang mengaktifkan *Green Bomb* adalah gestur 'X'.
- Pola yang mengaktifkan *Fire Ball* adalah gestur '-' atau garis lurus.
- *Weapon skills* berdurasi selama 15 detik, lalu kembali kepada kondisi serangan dasar.

3. **Musuh**
- Musuh terdiri dari *Creep* dan *Boss*.
- *Creep* terbagi menjadi :
  - *Normal Spider* (*Damage* : 10, *Health* : 100, *Speed* : 3)
  - *Small Spider* (*Damage* : 5, *Health* : 50, *Speed* : 3.5)
  - *Agile Spider* (*Damage* : 15, *Health* : 50, *Speed* : 5)
  - *Big Spider* (*Damage* : 20, *Health* : 200, *Speed* : 3.5)

- *Boss* terbagi menjadi :
    - *Normal boss* (*Damage*, *Health*, dan *Speed* berbeda-beda untuk setiap *stage*)
    - *Agile boss* (*Damage*, *Health*, dan *Speed* berbeda-beda untuk setiap *stage*)
- Kondisi musuh akan terbatas pada mengejar dan menyerang pemain.
- Musuh akan menyerang pemain jika sudah menyentuh pemain.
- Musuh bergerak dengan mencari jalur terdekat dengan pemain, namun akan menghindari objek yang tidak bisa ia lewati.

4. **Babak/Stage**

a. *Village Day Phase*
- Target membunuh 30 laba-laba.
- Musuh terdiri dari 2 jenis *creep*, yaitu *Normal Spider* dan *Small Spider*.
- Terdapat 2 lokasi muncul *Normal Spider* dan 1 lokasi untuk *Small Spider*.
- *Normal Spider* akan muncul setiap 15 detik sekali pada kedua lokasi, dan *Small Spider* akan muncul setiap 20 detik sekali.

b. *Village Night Phase*
- Terdapat 1 *Normal Boss*.
- Status untuk *Normal Boss* pada babak ini adalah *Damage* : 25, *Speed* : 4, dan *Health* : 1000.

c. *Railway Day Phase*
- Target membunuh 40 laba-laba.
- Musuh terdiri dari 2 jenis *creep*, yaitu *Agile Spider* dan *Big Spider*.
- Terdapat 2 lokasi muncul *Agile Spider* dan 1 lokasi untuk *Big Spider*.

- *Agile Spider* akan muncul setiap 10 detik sekali pada kedua lokasi, dan *Big Spider* akan muncul setiap 20 detik sekali.

d. Railway Night Phase
- Terdapat 3 *Normal Boss*.
- Status untuk *Normal Boss* pada babak ini adalah *Damage* : 25, *Speed* : 4, dan *Health* : 1000.

e. Forest Day Phase
- Target membunuh 40 laba-laba.
- Musuh terdiri dari 1 jenis *creep* yaitu *Big Spider*.
- Terdapat 4 lokasi muncul *Big Spider*.
- *Big Spider* akan muncul setiap 15 detik sekali pada seluruh lokasi.

f. Forest Night Phase
- Terdapat 5 *Agile Boss.*
- Status untuk *Agile Boss* pada babak ini adalah *Damage* : 20, *Speed* : 5, dan *Health* : 500.

g. Castle 1 Day Phase
- Target membunuh 50 laba-laba.
- Musuh terdiri dari 3 jenis *creep* yaitu *Normal Spider*, *Small Spider*, dan *Agile Spider*.
- Terdapat 1 lokasi muncul *Normal Spider*, 1 lokasi muncul *Small Spider* dan 1 lokasi muncul *Agile Spider*.
- Seluruh Jenis Laba-laba akan muncul setiap 10 detik sekali pada lokasi masing-masing.

h. Castle 1 Night Phase
- Terdapat 1 *Normal Boss* dan 1 *Agile Boss*.
- Status untuk *Normal Boss* pada babak ini adalah *Damage* : 25, *Speed* : 4, dan *Health* : 2500.
- Status untuk *Agile Boss* pada babak ini adalah *Damage* : 20, *Speed* : 5, dan *Health* : 1500.

i. Castle 2 Day Phase

- Target membunuh 50 laba-laba.
- Musuh terdiri dari 4 jenis *creep* yaitu *Normal Spider*, *Small Spider*, *Agile Spider*, dan *Big Spider*.
- Terdapat masing-masing 1 lokasi muncul untuk setiap jenis *creep*.
- *Normal Spider* muncul 15 detik sekali, *Small Spider* muncul 10 detik sekali, *Agile Spider* muncul 15 detik sekali, dan *Big Spider* muncul 20 detik sekali.

j. Castle 1 Night Phase

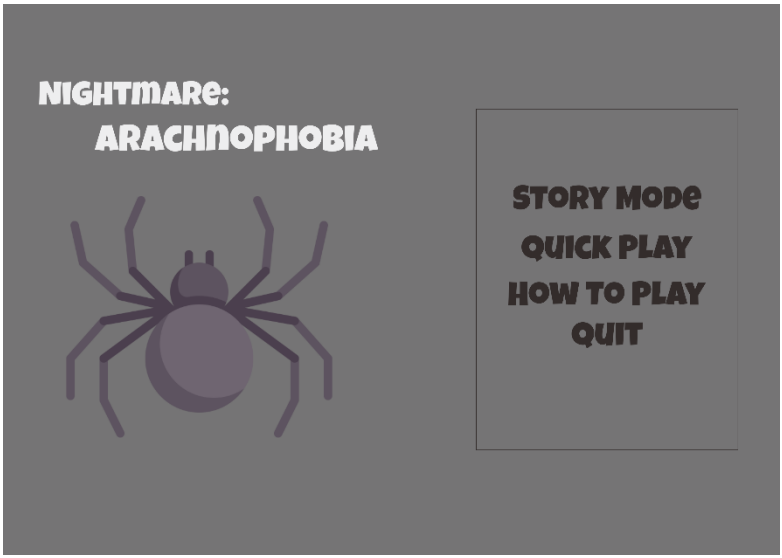
- Terdapat 2 *Normal Boss* dan 2 *Agile Boss*.
- Status untuk *Normal Boss* pada babak ini adalah *Damage* : 25, *Speed* : 4, dan *Health* : 2500.
- Status untuk *Agile Boss* pada babak ini adalah *Damage* : 20, *Speed* : 5, dan *Health* : 1500.

3.5. Perancangan Tampilan Antarmuka

Sub bab ini membahas bagaimana rancangan antarmuka pengguna yang akan digunakan untuk tugas akhir ini. Rancangan antarmuka yang dibahas meliputi ketentuan masukan dan rancangan halaman tampilan. Pada aplikasi permainan ini terdapat beberapa tampilan, yaitu tampilan menu utama, tampilan cara bermain, tampilan permainan (inti permainan), menu *pause*, dan tampilan akhir permainan.

3.5.1. Tampilan Menu Utama

Tampilan menu utama permainan merupakan tampilan yang pertama kali muncul ketika aplikasi dijalankan. Pada tampilan awal terdapat empat tombol, yaitu tombol *Story Mode*, *Quick Play*, *How to Play*, dan *Quit*. Tampilan rancangan antarmuka dari menu utama ditampilkan pada Gambar 3.8.



Gambar 3.8 Rancangan Antarmuka Halaman Menu Utama

1. Tombol **Story Mode,** berfungsi untuk memulai permainan pertama kali sesuai urutan alur cerita.
2. Tombol **Quick Play**, berfungsi untuk memilih s*tage* mana yang ingin dimainkan.
3. Tombol **How to Play**, berfungsi untuk menampilkan panel berupa tampilan cara bermain dan informasi mengenai permainan.
4. Tombol **Quit**, berfungsi untuk keluar dari permainan dan permainan akan tertutup.

3.5.2. Tampilan Cara Bermain

Tampilan cara bermain merupakan halaman yang akan muncul setelah pemain menekan tombol *How to Play*. Halaman ini berisikan informasi cara bermain yang perlu diketahui untuk dapat memainkan permainan. Gambar 3.9 menampilkan rancangan antarmuka cara bermain.

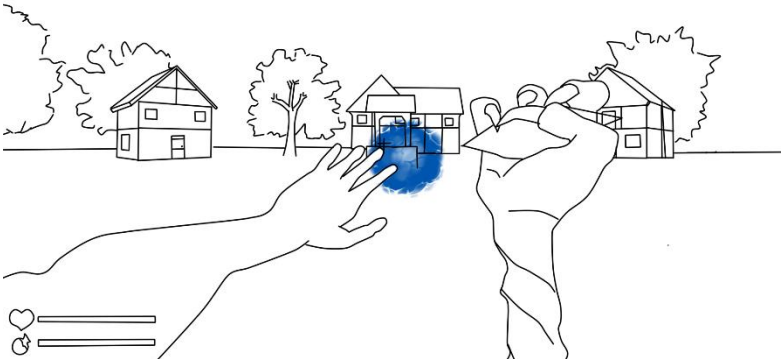
Gambar 3.9 Rancangan Antarmuka Halaman Cara Bermain

3.5.3. Tampilan Permainan

Tampilan permainan merupakan halaman yang muncul setelah pemain menekan tombol *Story Mode* pada menu utama. Tampilan permainan ini merupakan halaman utama dari keseluruhan permainan. Pada halaman inilah pemain melakukan interaksi dengan objek-objek di dalam permainan.

Pada rancangan tampilan ini menggunakan contoh senjata sihir sebagai tampilan permainan. Rancangan tampilan permainan terdiri dari posisi diam, menyerang dengan serangan dasar, menggambar pola, dan aktivasi *weapon skill.* Berikut ini rancangan tampilan permaian ditampilkan pada Gambar 3.10 – 3.13.

Gambar 3.10 Rancangan Antarmuka Permainan dengan Sihir



Gambar 3.11 Rancangan Serangan Dasar Sihir

Gambar 3.12 Rancangan Menggambar Pola

Gambar 3.13 Rancangan Weapon Skill Sihir

3.5.4. Tampilan Pause Menu

Tampilan *Pause Menu* akan muncul jika pemain menekan tombol *escape* pada *keyboard*. Rancangan tampilan *Pause Menu* dapat dilihat pada Gambar 3.14.

Gambar 3.14 Rancangan Antarmuka Pause Menu

1. Tombol **Resume,** berfungsi untuk melanjutkan permainan kembali.
2. Tombol **Main Menu**, berfungsi untuk kembali ke halaman menu utama.
3. Tombol **Quit**, berfungsi untuk keluar dari permainan dan permainan akan tertutup.

3.5.5. Tampilan Next Level

Tampilan *Next Level* ini adalah tampilan yang muncul saat pemain berhasil mengalahkan laba-laba sebanyak yang ditargetkan pada *stage* tersebut. Terdapat tombol *continue* yang mengarahkan pemain ke *stage* berikutnya. Rancangan tampilan *Next Level* dapat dilihat pada Gambar 3.15.

Gambar 3.15 Rancangan Antarmuka Next Level

3.5.6. Tampilan Akhir Permainan

Tampilan akhir permainan adalah tampilan yang muncul ketika *Health Point* pemain mencapai nol yang artinya pemain kalah. Pada gambar 3.16 ditampilkan rancangan tampilan akhir permainan.

Gambar 3.16 Rancangan Antarmuka Akhir Permainan

1. Tombol **Restart,** berfungsi untuk memulai permainan kembali pada *stage* saat ini.
2. Tombol **Quit**, berfungsi untuk keluar dari permainan dan permainan akan tertutup.

Saat permainan berakhir, pemain dapat memilih untuk memulai kembali permainan atau mengakhiri bermain dengan menutup permainan.

BAB IV
IMPLEMENTASI SISTEM

Bab ini membahas implementasi dari perancangan sistem sesuai dengan perancangan yang telah dibuat. Bahasa pemrograman yang digunakan untuk implementasi sistem adalah bahasa pemrograman C# dengan *game engine* Unity3D.

4.1. Lingkungan Implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan tugas akhir ini memiliki spesifikasi perangkat keras dan perangkat lunak yang ditunjukkan oleh Tabel 4.1.

Tabel 4.1 Spesifikasi Lingkungan Implementasi

Perangkat	Spesifikasi
Perangkat Keras	• Perangkat Pengembangan Sistem: AMD FX-9830P RADEON R7 3.0 GHz, AMD Radeon™ RX 460 4GB, RAM 16GB.
Perangkat Lunak	• Sistem Operasi: Microsoft Windows 10 64-bit • Perangkat Pengembang: Unity 2019.2.6f1 (64-bit) dan Visual Studio Community 2019. • Perangkat Pembantu: Microsoft Word 2019, Corel Draw X8, Adobe Photoshop CC 2015, dan Blender.

4.2. Implementasi Permainan

Implementasi dari masing-masing fungsi utama dituliskan menggunakan kode berbahasa C#. Implementasi fungsi diurut berdasarkan antarmuka dan *game object* yang ada pada permainan.

4.2.1. Implementasi Halaman Menu Utama

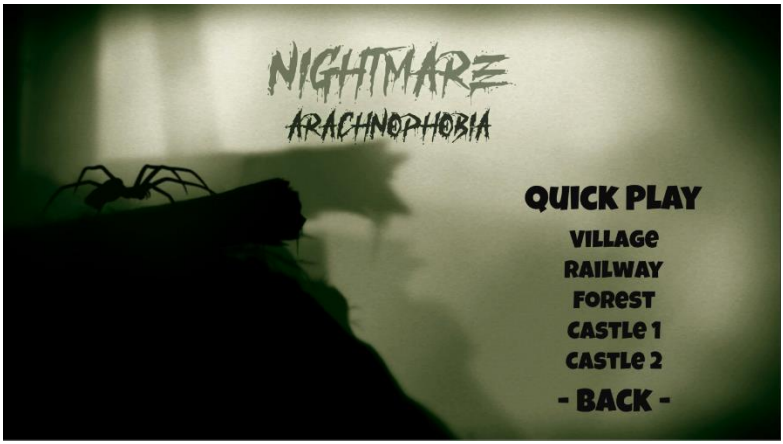
Gambar 4.1 merupakan hasil implementasi dari rancangan Menu Utama.

Gambar 4.1 Implementasi Halaman Menu Utama

Berikut djelaskan penjelasan dari konten Menu Utama sebagai berikut:

1. Tombol **Story Mode,** berfungsi untuk memulai permainan pertama kali sesuai urutan alur cerita.
2. Tombol **Quick Play**, berfungsi untuk memilih s*tage* mana yang ingin dimainkan.
3. Tombol **How to Play**, berfungsi untuk menampilkan panel berupa tampilan cara bermain dan informasi mengenai permainan.
4. Tombol **Quit**, berfungsi untuk keluar dari permainan dan permainan akan tertutup.

Pada Menu Utama ini terdapat tombol Quick Play yang digunakan untuk langsung memilih *stage* pada permainan. Menu Quick Play ini masih berada pada halaman yang sama dengan Menu utama. Pada Gambar 4.2 ditampilkan Menu Quick Play.



Gambar 4.2 Implementasi Menu Quick Play

Pada Kode Sumber 4.1 dijelaskan beberapa fungsi yang dibuat untuk menjalankan sistem Menu Utama.

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.  using UnityEngine.SceneManagement;
5.
6.  public class MainMenu : MonoBehaviour
7.  {
8.      public void PlayGame()
9.      {
10.         SceneManager.LoadScene("Village_day");
11.     }
12.
13.     public void Village()
14.     {
15.         SceneManager.LoadScene("Village_day");
16.     }
17.
18.     public void Railway()
19.     {
20.         SceneManager.LoadScene("Railway_day");
21.     }
```

```
22.
23.     public void Forest()
24.     {
25.         SceneManager.LoadScene("Forest_day");
26.     }
27.
28.     public void CastleOne()
29.     {
30.         SceneManager.LoadScene("Castle_1");
31.     }
32.
33.     public void CastleTwo()
34.     {
35.         SceneManager.LoadScene("Castle_2");
36.     }
37.
38.     public void HowToPlay()
39.     {
40.         SceneManager.LoadScene("HowToPlayScene");

41.     }
42.
43.     public void QuitGame()
44.     {
45.         Debug.Log("Quit Game");
46.         Application.Quit();
47.     }
48. }
```

Kode Sumber 4.1 Implementasi Menu Utama

4.2.2. Implementasi Halaman Cara Bermain

Pada Gambar 4.3 dapat dilihat hasil Implementasi tampilan halaman Cara Bermain.

Gambar 4.3 Implementasi Halaman Cara Bermain

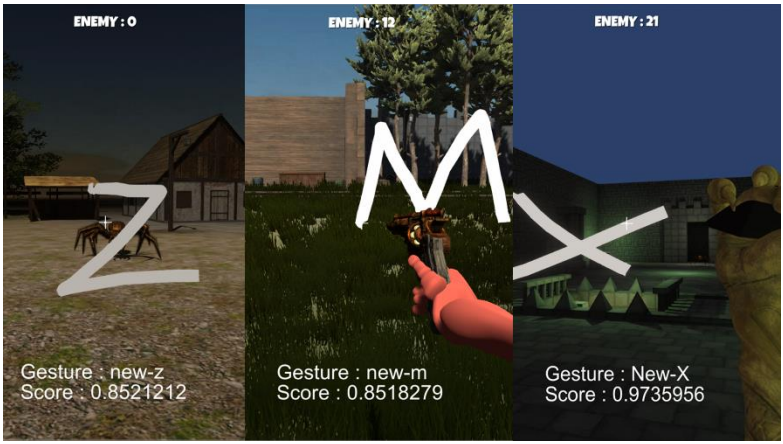
Pada Kode Sumber 4.2 dijelaskan beberapa fungsi yang dipakai pada implementasi halaman Cara Bermain. Fungsi yang dibuat hanya digunakan sebagai *trigger* untuk tombol *Back* yang digunakan untuk menutup halaman Cara Bermain dan kembali ke halaman Menu Utama.

```
1.  using UnityEngine.SceneManagement;
2.  using UnityEngine;
3.
4.  public class HowToPlay : MonoBehaviour
5.  {
6.      public void BackToMainMenu()
7.      {
8.          SceneManager.LoadScene("MainMenu");
9.      }
10. }
```

Kode Sumber 4.2 Implementasi Halaman Cara Bermain

4.2.3. Implementasi Pengenalan Bentuk Pola

Pada Gambar 4.4 dapat dilihat hasil implementasi dari pengenalan pola yang digambar pemain. Pemain menggambar pola menggunakan tombol *mouse* sebelah kanan (RMB) jika *Mana Point* telah penuh dan mengkonfirmasi pola dengan menggunakan tombol *space* pada *keyboard*. Hasil dari pola yang dikenali akan dikirim dalam bentuk *string* ke masing-masing *script* senjata. Apabila hasil yang dikenali benar, maka *Weapon Skills* akan aktif.



Gambar 4.4 Implementasi Pengenalan Pola

Pada Kode Sumber 4.3 dibuat *script* untuk implementasi sistem menggambar pola pada permainan dan menyambungkannya dengan sistem *Point-Cloud Gesture Recognizer*.

```
1.  using UnityEngine;
2.  using UnityEngine.UI;
3.  using System;
4.  using System.Collections;
5.  using System.Collections.Generic;
6.  using System.IO;
7.  using PDollarGestureRecognizer;
8.
9.  public class Demo : MonoBehaviour {
10.  Transform player;
11.  FirstPersonAIO playerMovement;
12.
13.  public Transform gestureOnScreenPrefab;
14.  public String resultSkill;
15.  public SpawnSpell spawnSpell;
16.  public Sword swordSkill;
17.  public Shoot gunSkill;
18.  public Slider manaSlider;
19.  public float currentMana = 0f;
```

```
20.
21. private List<Gesture> trainingSet = new List<Gest
    ure>();
22.  private List<Point> points = new List<Point>();
23.  private int strokeId = -1;
24.  private Vector3 virtualKeyPosition = Vector2.zer
    o;
25.  private Rect drawArea;
26.  private RuntimePlatform platform;
27.  private int vertexCount = 0;
28.  private List<LineRenderer> gestureLinesRenderer
    = new List<LineRenderer>();
29.  private LineRenderer currentGestureLineRenderer;
30.
31.  private float thresholdX = 0.875f;
32.  private float thresholdLine = 0.85f;
33.  private float thresholdZ = 0.7f;
34.  private float thresholdV = 0.7f;
35.  private float thresholdS = 0.7f;
36.  private float thresholdM = 0.7f;
37.
38.  private string messageGesture;
39.  private string messageScore;
40.  private bool recognized;
41.  private GUIStyle guiStyle = new GUIStyle();
42.
43.  void Start () {
44.      player = GameObject.FindWithTag("Player").tr
    ansform;
45.      playerMovement = player.GetComponent<FirstPe
    rsonAIO>();
46.
47.      drawArea = new Rect(0, 0, Screen.width, Scre
    en.height);
48.      platform = Application.platform;
49.
50.      TextAsset[] gesturesXml = Resources.LoadAll<
    TextAsset>("GestureSet/10-stylus-MEDIUM/");
51.      foreach (TextAsset gestureXml in gesturesXml
    )
52.          trainingSet.Add(GestureIO.ReadGestureFro
    mXML(gestureXml.text));
```

```
53.
54.         string[] filePaths = Directory.GetFiles(Appl
    ication.persistentDataPath, "*.xml");
55.         foreach (string filePath in filePaths)
56.             trainingSet.Add(GestureIO.ReadGestureFro
    mFile(filePath));
57.         }
58.
59.  void Update(){
60.         if(currentMana < 100){
61.            currentMana += 5 * Time.deltaTime;
62.         }
63.
64.         else{
65.             if(Input.GetMouseButton(1)){
66.                playerMovement.enableCameraMovement =
     false;
67.                playerMovement.playerCanMove = false;
68.                Cursor.lockState = CursorLockMode.Non
    e;
69.                Cursor.lockState = CursorLockMode.Con
    fined;
70.                Cursor.visible = true;
71.                virtualKeyPosition = new Vector3(Inpu
    t.mousePosition.x, Input.mousePosition.y);
72.          }
73.
74.             if(drawArea.Contains(virtualKeyPosition)
    ){
75.                 if(Input.GetMouseButtonDown(1)){
76.                     if(recognized){
77.                        recognized = false;
78.                        strokeId = -1;
79.                        points.Clear();
80.
81.                        foreach(LineRenderer lineRend
    erer in gestureLinesRenderer){
82.                            lineRenderer.SetVertexCo
    unt(0);
83.                            Destroy(lineRenderer.gam
    eObject);
84.                         }
```

```
85.
86.                     gestureLinesRenderer.Clear()
    ;
87.                 }
88.
89.                 ++strokeId;
90.                 Transform tmpGesture = Instantia
    te(gestureOnScreenPrefab, transform.position, tra
    nsform.rotation) as Transform;
91.                 currentGestureLineRenderer = tmp
    Gesture.GetComponent<LineRenderer>();
92.                 gestureLinesRenderer.Add(current
    GestureLineRenderer);
93.                 vertexCount = 0;
94.             }
95.
96.             if(Input.GetMouseButton(1)) {
97.                 points.Add(new Point(virtualKeyPo
    sition.x, -virtualKeyPosition.y, strokeId));
98.                 currentGestureLineRenderer.SetVer
    texCount(++vertexCount);
99.         currentGestureLineRenderer.SetPosition(v
    ertexCount - 1, Camera.main.ScreenToWorldPoint(ne
    w Vector3(virtualKeyPosition.x, virtualKeyPositio
    n.y, 3)));
100.                    }
101.                }
102.            }
103.            manaSlider.value = currentMana;
104.        }
105.
106.        void OnGUI() {
107.            guiStyle.fontSize = 50;
108.            guiStyle.normal.textColor = Color.whi
    te;
109.
110.            GUI.Label(new Rect(750, Screen.height
     - 200, 0, 0), messageGesture, guiStyle);
111.            GUI.Label(new Rect(750, Screen.height
     - 150, 0, 0), messageScore, guiStyle);
112.
```

```
113.            if (Input.GetKeyDown("space") && curr
    entMana > 99) {
114.                recognized = true;
115.                Gesture candidate = new Gesture(p
    oints.ToArray());
116.                int n = points.ToArray().Length;
117.
118.                if (n == 1){
119.                    return;}
120.
121.                else{
122.                    Result gestureResult = PointC
    loudRecognizer.Classify(candidate, trainingSet.To
    Array());
123.                    resultSkill = gestureResult.G
    estureClass;
124.
125.                    switch (resultSkill){
126.                        case "New-X":
127.                            if(gestureResult.Scor
    e < thresholdX){
128.                                resultSkill = "Ges
    ture Not Found";
129.                                messageGesture = r
    esultSkill;
130.                                messageScore = "";
131.                            }
132.                            else{
133.                                messageGesture = "
    Gesture : " + resultSkill;
134.                                messageScore = "Sc
    ore : " + gestureResult.Score;
135.                                currentMana = 0;
136.                            }
137.                            break;
138.
139.                        case "line":
140.                            if(gestureResult.Scor
    e < thresholdLine){
141.                                resultSkill = "Ges
    ture Not Found";
```

```
142.                                    messageGesture = r
    esultSkill;
143.                                    messageScore = "";

144.                                }
145.                                else{
146.                                    messageGesture = "
    Gesture : " + resultSkill;
147.                                    messageScore = "Sc
    ore : " + gestureResult.Score;
148.                                    currentMana = 0;
149.                                }
150.                                break;
151.
152.                            case "new-z":
153.                                if(gestureResult.Scor
    e < thresholdZ){
154.                                    resultSkill = "Ges
    ture Not Found";
155.                                    messageGesture = r
    esultSkill;
156.                                    messageScore = "";
157.                                }
158.                                else{
159.                                    messageGesture = "
    Gesture : " + resultSkill;
160.                                    messageScore = "Sc
    ore : " + gestureResult.Score;
161.                                    currentMana = 0;
162.                                }
163.                                break;
164.
165.                            case "new-v":
166.                                if(gestureResult.Scor
    e < thresholdV){
167.                                    resultSkill = "Ges
    ture Not Found";
168.                                    messageGesture = r
    esultSkill;
169.                                    messageScore = "";
170.                                }
171.                                else{
```

```
172.                                    messageGesture = "
    Gesture : " + resultSkill;
173.                                    messageScore = "Sc
    ore : " + gestureResult.Score;
174.                                    currentMana = 0;
175.                                }
176.                                break;
177.
178.                            case "new-s":
179.                                if(gestureResult.Scor
    e < thresholdS){
180.                                    resultSkill = "Ges
    ture Not Found";
181.                                    messageGesture = r
    esultSkill;
182.                                    messageScore = "";
183.                                }
184.                                else{
185.                                    messageGesture = "
    Gesture : " + resultSkill;
186.                                    messageScore = "Sc
    ore : " + gestureResult.Score;
187.                                    currentMana = 0;
188.                                }
189.                                break;
190.
191.                            case "new-m":
192.                                if(gestureResult.Scor
    e < thresholdM){
193.                                    resultSkill = "Ges
    ture Not Found";
194.                                    messageGesture = r
    esultSkill;
195.                                    messageScore = "";
196.                                }
197.                                else{
198.                                    messageGesture = "
    Gesture : " + resultSkill;
199.                                    messageScore = "Sc
    ore : " + gestureResult.Score;
200.                                    currentMana = 0;
201.                                }
```

```
202.                                break;
203.                        }
204.
205.                        spawnSpell.flag = true;
206.                        swordSkill.flag = true;
207.                        gunSkill.flag = true;
208.                        playerMovement.enableCameraMo
    vement = true;
209.                        playerMovement.playerCanMove
    = true;
210.                        Cursor.lockState = CursorLock
    Mode.Locked;
211.                        Cursor.visible = false;
212.                       }
213.                  }
214.              }
215.          }
```

Kode Sumber 4.3 Implementasi Menggambar Pola

Cara kerja kode sumber diatas adalah menhubungkan antara hasil gambar dari *Line Renderer* yang dibuat oleh pemain menggunakan gerakan *mouse* lalu menyimpan titik-titik koordinat dari *Line Renderer* tersebut. Titik-titik koordinat tadi disimpan pada sebuah *Array List*. Kemudian hasil *Array List* tersebut dikirimkan ke Kode Sumber 4.4 dan dicocokkan dengan *Dataset* yang sudah tersedia. Kode Sumber 4.3 akan memanggil fungsi **Classify()** pada Kode Sumber 4.4 yang berfungsi untuk membandingkan antara kedua *Array List*. Berikut dijelaskan Kode Sumber 4.4.

```
1.  using System;
2.  using System.Collections.Generic;
3.
4.  using UnityEngine;
5.
6.  namespace PDollarGestureRecognizer
7.  {
8.      public class PointCloudRecognizer
```

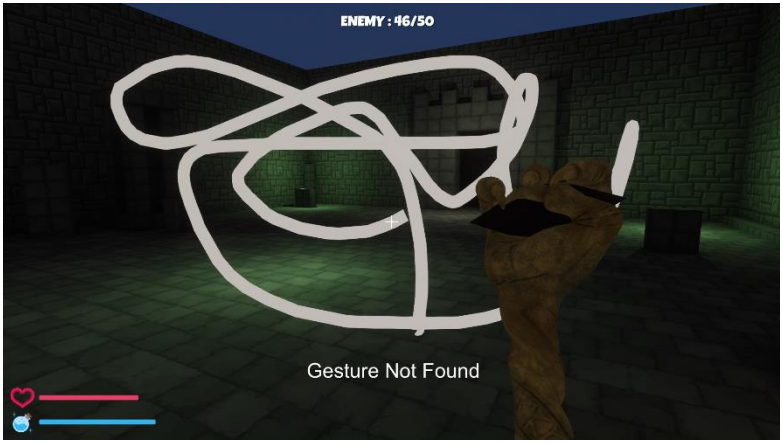
```
9.      {
10.         public static Result Classify(Gesture can
    didate, Gesture[] trainingSet)
11.         {
12.             float minDistance = float.MaxValue;
13.             string gestureClass = "";
14.             foreach (Gesture template in training
    Set)
15.             {
16.                 float dist = GreedyCloudMatch(can
    didate.Points, template.Points);
17.                 if (dist < minDistance)
18.                 {
19.                     minDistance = dist;
20.                     gestureClass = template.Name;

21.                 }
22.             }
23.
24.             return gestureClass == "" ? new Resul
    t() { GestureClass = "No match", Score = 0.0f } :
     new Result() { GestureClass = gestureClass, Scor
    e = Mathf.Max((minDistance - 2.0f) / -
    2.0f, 0.0f) };
25.         }
26.
27.         private static float GreedyCloudMatch(Poi
    nt[] points1, Point[] points2)
28.         {
29.             int n = points1.Length;
30.             float eps = 0.5f;
31.             int step = (int)Math.Floor(Math.Pow(n
    , 1.0f - eps));
32.             float minDistance = float.MaxValue;
33.             for (int i = 0; i < n; i += step)
34.             {
35.                 float dist1 = CloudDistance(point
    s1, points2, i);
36.                 float dist2 = CloudDistance(point
    s2, points1, i);
37.                 minDistance = Math.Min(minDistanc
    e, Math.Min(dist1, dist2));
```

```
38.             }
39.             return minDistance;
40.         }
41.
42.         private static float CloudDistance(Point[
    ] points1, Point[] points2, int startIndex)
43.         {
44.             int n = points1.Length;
45.             bool[] matched = new bool[n];
46.             Array.Clear(matched, 0, n);
47.
48.             float sum = 0;
49.             int i = startIndex;
50.             do
51.             {
52.                 int index = -1;
53.                 float minDistance = float.MaxValu
    e;
54.                 for (int j = 0; j < n; j++)
55.                     if (!matched[j])
56.                     {
57.                         float dist = Geometry.Sqr
    EuclideanDistance(points1[i], points2[j]);
58.                         if (dist < minDistance)
59.                         {
60.                             minDistance = dist;
61.                             index = j;
62.                         }
63.                     }
64.                 matched[index] = true;
65.                 float weight = 1.0f - ((i - start
    Index + n) % n) / (1.0f * n);
66.                 sum += weight * minDistance;
67.                 i = (i + 1) % n;
68.             } while (i != startIndex);
69.             return sum;
70.         }
71.     }
72. }
```

Kode Sumber 4.4 Implementasi Point-Cloud Gesture Recognizer

Kode Sumber 4.4 akan mengembalikan keluaran berupa tipe data *string* dan *float* yang akan digunakan pada Kode Sumber 4.3 sebagai nama bentuk pola garis dan *score* kecocokan dengan *dataset.* Dari Kode Sumber 4.3 akan dikirim nama dari pola ke masing-masing *script* senjata yang akan dicocokkan dengan menggunakan *Switch Case* dengan pola untuk aktivasi *Weapon Skills.*

Namun jika bentuk pola garis yang digambar oleh pemain terlalu abstrak atau tidak cocok dengan bentuk manapun pada *dataset*, maka akan ditampilkan pemberitahuan bahwa bentuk tidak ditemukan seperti pada Gambar 4.5. Hal ini karena setiap bentuk pola memiliki *threshold* masing-masing untuk mencegah pemain untuk menggambar pola garis dengan sembarangan. *Threshold* untuk setiap bentuk pola dapat dilihat pada Kode Sumber 4.3.



Gambar 4.5 Bentuk Pola Tidak Ditemukan

4.2.4. Implementasi Senjata Pedang

Senjata pedang merupakan senjata pertama yang digunakan oleh pemain untuk menyerang. Pedang digunakan pemain selama *stage* 1 yaitu Village Day dan Village Night.

Senjata pedang ini memiliki serangan dasar seperti yang ditampilkan pada Gambar 4.5 dan juga *Weapon Skills : Damage Up* dan *Rapid Slash. Weapon Skills* ini dapat diaktifkan dengan menggambar pola yang tepat. Pada Gambar 4.6 ditampilkan saat pemain menggambar pola untuk senjata pedang.

Gambar 4.6 Implementasi Serangan Dasar Senjata Pedang

Gambar 4.7 Implementasi Weapon Skill Senjata Pedang

Pada Kode Sumber 4.5 dijelaskan bagaimana senjata pedang dapat menyerang dengan serangan dasar dan juga *Weapon Skills* untuk memberikan *damage* ke musuh.

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.
5.  public class Sword : MonoBehaviour
6.  {
7.      public float damage = 100f;
8.      public Camera cam;
9.      public float range = 3f;
10.     public float fireRate = 0.5f;
11.     public bool flag = false;
12.     public Demo resultDemo;
13.     public GameObject impactEffect;
14.     public GameObject damageUpUI;
15.     public GameObject rapidSlashUI;
16.
17.     protected string skillWord = "";
18.
19.     AudioSource swordAudio;
20.     Animator anim;
21.     GameObject player;
22.     PlayerHealth playerHealth;
23.
24.     private float nextTimeToFire = 0f;
25.     private float skillDura = 0f;
26.
27.     void Awake()
28.     {
29.         swordAudio = GetComponent<AudioSource>();

30.         anim = GetComponent<Animator>();
31.         player = GameObject.FindGameObjectWithTag
    ("Player");
32.         playerHealth = player.GetComponent<Player
    Health>();
33.     }
```

```
34.
35.     private void Update()
36.     {
37.         skillWord = resultDemo.resultSkill;
38.
39.         if (skillWord != null && flag)
40.         {
41.             switch (skillWord)
42.             {
43.                 case "new-z":
44.                     damage = 200f;
45.                     skillDura = 15f;
46.                     flag = false;
47.                     damageUpUI.SetActive(true);
48.                     Debug.Log("ini masuk new-
    z");
49.                     break;
50.
51.                 case "new-v":
52.                     fireRate = 1f;
53.                     skillDura = 15f;
54.                     flag = false;
55.                     rapidSlashUI.SetActive(true);

56.                     Debug.Log("ini masuk new-
    v");
57.                     break;
58.
59.                 default:
60.                     damage = 100f;
61.                     fireRate = 0.5f;
62.                     break;
63.             }
64.         }
65.
66.         skillDura -= 1 * Time.deltaTime;
67.         if (skillDura <= 0)
68.         {
69.             damage = 100f;
70.             fireRate = 0.5f;
71.             damageUpUI.SetActive(false);
72.             rapidSlashUI.SetActive(false);
```

```
73.         }
74.
75.         if (Input.GetButtonDown("Fire1") && Time.
    time >= nextTimeToFire && playerHealth.currentHea
    lth > 0)
76.         {
77.             anim.Play("swordSlash");
78.             anim.SetTrigger("slashed");
79.             nextTimeToFire = Time.time + 1f / fir
    eRate;
80.             Slashing();
81.         }
82.     }
83.
84.     void Slashing()
85.     {
86.         Ray ray = cam.ScreenPointToRay(Input.mous
    ePosition);
87.         RaycastHit hit;
88.         swordAudio.Play();
89.         if(Physics.Raycast(ray, out hit, range))

90.         {
91.             Debug.Log(hit.transform.name);
92.             EnemyHealth enemyhealth = hit.transfo
    rm.GetComponent<EnemyHealth>();
93.             if (enemyhealth != null)
94.             {
95.                 StartCoroutine(ShowImpact(hit, en
    emyhealth));
96.             }
97.         }
98.     }
99.
100.            IEnumerator ShowImpact(RaycastHit hit,
     EnemyHealth enemyhealth)
101.            {
102.                yield return new WaitForSeconds(1)
    ;
103.                enemyhealth.TakeDamage(damage);
```

```
104.                GameObject impGO = Instantiate(imp
    actEffect, hit.point, Quaternion.LookRotation(hit
    .normal));
105.                Destroy(impGO, 0.5f);
106.            }
107.        }
```

Kode Sumber 4.5 Implementasi Senjata Pedang

Cara kerja Kode Sumber 4.5 adalah dengan sistem *Raycasting*. *Ray* pada *Raycasting* diposisikan pada *pointer mouse* yang sudah terkunci pada layar kamera. Jika pada saat pemain menyerang dan terdapat *game object* yang terkena *Raycast* dengan jarak yang sudah ditentukan, maka akan tercatat pada *log game.* Apabila *game object* yang terkena *raycast* adalah objek musuh, maka akan Kode Sumber 4.5 akan memanggil fungsi **TakeDamage()** pada *script* objek musuh. Besarnya *damage* telah dideklarasikan sebelumnya. Alasan mengapa digunakan sistem *Raycasting* pada senjata pedang daripada sistem *Collider* adalah agar pemain tidak hanya menyerang dengan memposisikan objek pedang dibagian depan untuk ditabrakan ke musuh. Dengan menggunakan sistem ini, pemain harus menyerang pada jarak serang agar bisa memberikan *damage* ke objek musuh.

4.2.5. Implementasi Senjata Pistol

Senjata pistol merupakan senjata kedua yang digunakan oleh pemain untuk menyerang. Pistol digunakan pemain selama *stage* 2 yaitu Railway Day dan Railway Night. Senjata pistol ini memiliki serangan dasar seperti yang ditampilkan pada Gambar 4.7 dan juga *Weapon Skills : Damage Up* dan *Rapid Fire. Weapon Skills* ini dapat diaktifkan dengan menggambar pola yang tepat. Pada Gambar 4.8 ditampilkan saat pemain menggambar pola untuk senjata pistol.

Gambar 4.8 Implementasi Serangan Dasar Senjata Pedang

Gambar 4.9 Implementasi *Weapon Skill Senjata Pistol*

Pada Kode Sumber 4.6 dijelaskan bagaimana senjata pistol dapat menyerang dengan serangan dasar dan juga *Weapon Skills* untuk memberikan *damage* ke musuh.

```
1.  using UnityEngine;
2.
3.  public class Shoot : MonoBehaviour
4.  {
5.      public float damage = 50;
6.      public float fireRate = 1f;
7.      public float range = 100f;
8.      public Camera fpsCam;
9.      public ParticleSystem muzzleFlash;
10.     public GameObject impactEffect;
11.     public GameObject damageUpUI;
12.     public GameObject rapidFireUI;
13.     public bool flag = false;
14.     public Demo resultDemo;
15.
16.     protected string skillWord = "";
17.     private float nextTimeToFire = 0f;
18.     float skillDura = 0f;
19.
20.     Animator anim;
21.     AudioSource gunAudio;
22.     GameObject player;
23.     PlayerHealth playerHealth;
24.
25.     void Awake()
26.     {
27.         anim = GetComponent<Animator>();
28.         gunAudio = GetComponent<AudioSource>();
29.         player = GameObject.FindGameObjectWithTag
    ("Player");
30.         playerHealth = player.GetComponent<Player
    Health>();
31.     }
32.
33.     private void Update()
34.     {
35.         skillWord = resultDemo.resultSkill;
36.
37.         if (skillWord != null && flag)
38.         {
39.             switch (skillWord)
40.             {
```

```
41.                 case "new-s":
42.                     damage = 100f;
43.                     skillDura = 15f;
44.                     flag = false;
45.                     damageUpUI.SetActive(true);
46.                     Debug.Log("ini masuk new-
    s");
47.                     break;
48.
49.                 case "new-m":
50.                     fireRate = 2f;
51.                     skillDura = 15f;
52.                     flag = false;
53.                     rapidFireUI.SetActive(true);

54.                     Debug.Log("ini masuk new-
    m");
55.                     break;
56.
57.                 default:
58.                     damage = 50f;
59.                     fireRate = 1f;
60.                     break;
61.             }
62.         }
63.
64.         skillDura -= 1 * Time.deltaTime;
65.
66.         if (skillDura <= 0)
67.         {
68.             damage = 50f;
69.             fireRate = 1f;
70.             damageUpUI.SetActive(false);
71.             rapidFireUI.SetActive(false);
72.         }
73.
74.         if (Input.GetButtonDown("Fire1") && Time.
    time >= nextTimeToFire && playerHealth.currentHea
    lth > 0)
75.         {
76.             anim.Play("shootGun");
77.             anim.SetTrigger("Shooted");
```

```
78.             nextTimeToFire = Time.time + 1f / fir
    eRate;
79.             Shooting();
80.         }
81.     }
82.
83.     void Shooting()
84.     {
85.         muzzleFlash.Play();
86.         gunAudio.Play();
87.         RaycastHit hit;
88.
89.         if (Physics.Raycast(fpsCam.transform.posi
    tion, fpsCam.transform.forward, out hit, range))

90.         {
91.             Debug.Log(hit.transform.name);
92.             EnemyHealth enemyhealth = hit.transfo
    rm.GetComponent <EnemyHealth>();
93.
94.             if (enemyhealth != null)
95.             {
96.                 enemyhealth.TakeDamage(damage);
97.             }
98.
99.             GameObject impGO = Instantiate(impact
    Effect, hit.point, Quaternion.LookRotation(hit.no
    rmal));
100.                    Destroy(impGO, 0.5f);
101.                }
102.            }
103.        }
```

Kode Sumber 4.6 Implementasi *Senjata Pistol*

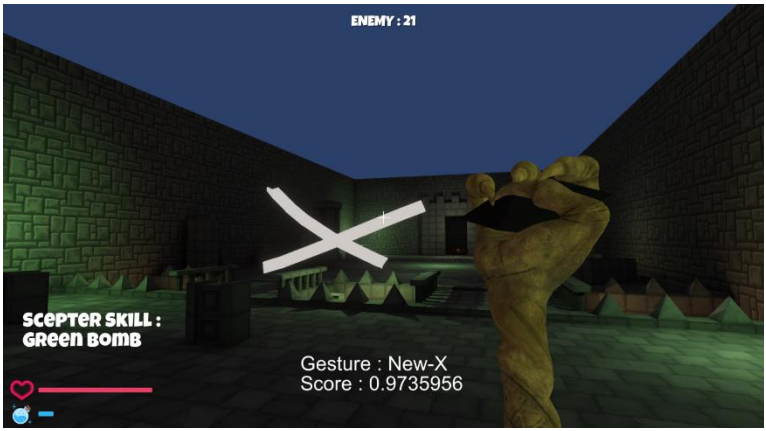
Cara kerja Kode Sumber 4.6 adalah dengan sistem *Raycasting*. *Ray* pada *Raycasting* diposisikan pada *pointer mouse* yang sudah terkunci pada layar kamera. Jika pada saat pemain menyerang dan terdapat *game object* yang terkena *Raycast* dengan jarak yang sudah ditentukan, maka akan tercatat pada *log game.*

Apabila *game object* yang terkena *raycast* adalah objek musuh, maka akan Kode Sumber 4.6 akan memanggil fungsi **TakeDamage()** pada *script* objek musuh. Besarnya *damage* telah dideklarasikan sebelumnya.

4.2.6. Implementasi Senjata Sihir

Senjata sihir merupakan senjata ketiga yang digunakan oleh pemain untuk menyerang. Sihir digunakan pemain selama pada tiga *stage* terakhir yaitu Forest Day & Night, Castle 1, dan Caslte 2. Senjata sihir ini memiliki serangan dasar seperti yang ditampilkan pada Gambar 4.9 dan juga *Weapon Skills : Green Bomb* dan *Fire Bal. Weapon Skills* ini dapat diaktifkan dengan menggambar pola yang tepat. Pada Gambar 4.10 ditampilkan saat pemain menggambar pola untuk senjata sihir.

Gambar 4.10 Implementasi Serangan Dasar Senjata Sihir



Gambar 4.11 Implementasi Weapon Skill Senjata Sihir

Pada Kode Sumber 4.7 dijelaskan bagaimana senjata sihir dapat menyerang dengan serangan dasar dan juga *Weapon Skills*. Dijelaskan juga bagiamana senjata sihir dapat menembakkan *projectile* sihir.

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.
5.  public class SpawnSpell : MonoBehaviour
6.  {
7.      public Camera cam;
8.      public float range;
9.      public float fireRate = 1f;
10.     public GameObject firePoint;
11.     public List<GameObject> vfx = new List<GameOb
    ject>();
12.     public Demo resultDemo;
13.     public bool flag = false;
14.     public GameObject greenBombUI;
15.     public GameObject fireBallUI;
16.
17.     protected string skillWord = "";
18.
```

```
19.     private Ray rayMouse;
20.     private Quaternion rotation;
21.     private Vector3 pos;
22.     private Vector3 direction;
23.     private GameObject effectToSpawn;
24.     private float nextTimeToFire = 0f;
25.     private float skillDura = 0f;
26.
27.     Animator anim;
28.     AudioSource spellAudio;
29.     GameObject player;
30.     PlayerHealth playerHealth;
31.
32.     void Start()
33.     {
34.         effectToSpawn = vfx[0];
35.         anim = GetComponent<Animator>();
36.         spellAudio = GetComponent<AudioSource>();

37.         player = GameObject.FindGameObjectWithTag
    ("Player");
38.         playerHealth = player.GetComponent<Player
    Health>();
39.     }
40.
41.     void Update()
42.     {
43.         skillWord = resultDemo.resultSkill;
44.
45.         if (skillWord != null && flag)
46.         {
47.             switch (skillWord)
48.             {
49.                 case "New-X":
50.                     effectToSpawn = vfx[1];
51.                     skillDura = 15f;
52.                     flag = false;
53.                     greenBombUI.SetActive(true);

54.                     Debug.Log("ini masuk new x");

55.                     break;
```

```
56.
57.                 case "line":
58.                     effectToSpawn = vfx[2];
59.                     skillDura = 15f;
60.                     flag = false;
61.                     fireBallUI.SetActive(true);
62.                     Debug.Log("ini masuk line");

63.                     break;
64.
65.                 default:
66.                     effectToSpawn = vfx[0];
67.                     break;
68.             }
69.         }
70.
71.         skillDura -= 1 * Time.deltaTime;
72.
73.         if (skillDura <= 0)
74.         {
75.             effectToSpawn = vfx[0];
76.             greenBombUI.SetActive(false);
77.             fireBallUI.SetActive(false);
78.         }
79.
80.         if (Input.GetButtonDown("Fire1") && Time.
    time >= nextTimeToFire && playerHealth.currentHea
    lth > 0)
81.         {
82.             anim.Play("castSpell");
83.             anim.SetTrigger("casted");
84.             nextTimeToFire = Time.time + 1f / fir
    eRate;
85.             spawnVfx();
86.         }
87.     }
88.
89.     void spawnVfx()
90.     {
91.         RaycastHit hit;
92.         GameObject vfx;
93.         spellAudio.Play();
```

```
94.
95.         if (Physics.Raycast(cam.transform.positio
    n, cam.transform.forward, out hit, range))
96.         {
97.             RotateToDirection(gameObject, hit.poi
    nt);
98.             vfx = Instantiate(effectToSpawn, fire
    Point.transform.position, Quaternion.identity);
99.             vfx.transform.localRotation = rotatio
    n;
100.                }
101.
102.                else
103.                {
104.                    return;
105.                }
106.            }
107.
108.            void RotateToDirection(GameObject obj,
     Vector3 destination)
109.            {
110.                direction = destination - obj.tran
    sform.position;
111.                rotation = Quaternion.LookRotation
    (direction);
112.            }
113.
114.        }
```

Kode Sumber 4.7 Implementasi Senjata Sihir (1)

Cara kerja Kode Sumber 4.7 adalah dengan cara menembakkan *game object* berupa *projectile* sihir lurus dari titik tembak. *Projectile* sihir sendiri memiliki *script* yang ditunjukkan oleh Kode Sumber 4.8 yang mengatur gerakan *projectile* dan memberikan *damage* jika mengenai musuh.

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
```

```
4.
5.  public class SpellMove : MonoBehaviour
6.  {
7.      public float speed = 20f;
8.      public float damage = 100f;
9.      public GameObject impactEffect;
10.
11.     void Update()
12.     {
13.         if (speed != 0)
14.         {
15.             transform.position += transform.forwa
    rd * (speed * Time.deltaTime);
16.         }
17.
18.         else
19.         {
20.             Debug.Log("no speed");
21.         }
22.     }
23.
24.     void OnCollisionEnter(Collision collision)
25.     {
26.         Debug.Log(this.gameObject.name + " hit "
    + collision.gameObject.name);
27.         speed = 0f;
28.         Destroy(gameObject);
29.         EnemyHealth enemyhealth = collision.trans
    form.GetComponent<EnemyHealth>();
30.
31.         if (enemyhealth != null)
32.         {
33.             enemyhealth.TakeDamage(damage);
34.         }
35.
36.         ContactPoint contact = collision.contacts
    [0];
37.         Quaternion rot = Quaternion.FromToRotatio
    n(Vector3.up, contact.normal);
38.         Vector3 pos = contact.point;
39.         var hitVFX = Instantiate(impactEffect, po
    s, rot);
```

```
        Destroy(hitVFX, 1f);
    }
}
```

Kode Sumber 4.8 Implementasi Senjata Sihir (2)

Pada Kode Sumber 4.8 dijelaskan *projectile* sihir yang telah dimunculkan pada Kode Sumber 4.7 akan digerakkan lurus ke depan selama objek tersebut memiliki kecepatan. Jika *projectile* sihir menabrak *game object* lain, maka *projectile* sihir akan berhenti dan meledak dengan cara memanggil *impactEffect.* Jika *game object* yang ditabrak adalah objek musuh, maka Kode Sumber 4.8 akan memanggil fungsi **TakeDamage()** yang ada pada *script* objek musuh. Besarnya *damage* yang diakibatkan telah dideklarasikan sebelmnya.

4.2.7. Implementasi Objek Player

Game object player atau pemain pada permainan Nightmare: Arachnophobia ini memiliki komponen berupa *player movement* dan juga *player health.*

Player movement menangani gerakan pemain dalam sudut pandang orang pertama (*First Person*). Untuk implementasi dari *player movement* sendiri, penulis menggunakan aset yang telah disediakan pada Unity Store yaitu “First Person All in One” yang berupa *game object* dengan kamera dan *script* untuk bergerak. Aset ini disediakan oleh Unity Store karena banyaknya permainan dengan konsep sudut pandang orang perama atau *first person* yang dikembangkan dengan Unity3D.

Player health menangani darah atau *Health Point* dari pemain. Implementasi dari *player health* dijelaskan pada Kode Sumber 4.9.

```
using UnityEngine;
using UnityEngine.UI;
```

```
3.  using System.Collections;
4.
5.  public class PlayerHealth : MonoBehaviour
6.  {
7.      public int startingHealth = 100;
8.      public int currentHealth;
9.      public Slider healthSlider;
10.     public Image damageImage;
11.     public float flashSpeed = 5f;
12.     public Color flashColour = new Color(1f, 0f,
    0f, 0.1f);
13.
14.     FirstPersonAIO playerMovement;
15.     bool isDead;
16.     bool damaged;
17.
18.     void Start()
19.     {
20.         playerMovement = GetComponent<FirstPerson
    AIO>();
21.         currentHealth = startingHealth;
22.     }
23.
24.     void Update()
25.     {
26.         if (damaged)
27.         {
28.             damageImage.color = flashColour;
29.         }
30.
31.         else
32.         {
33.             damageImage.color = Color.Lerp(damage
    Image.color, Color.clear, flashSpeed * Time.delta
    Time);
34.         }
35.         damaged = false;
36.     }
37.
38.     public void TakeDamage(int amount)
39.     {
40.         damaged = true;
```

```
41.         currentHealth -= amount;
42.         healthSlider.value = currentHealth;
43.
44.         if (currentHealth <= 0 && !isDead)
45.         {
46.             Death();
47.         }
48.     }
49.
50.     void Death()
51.     {
52.         isDead = true;
53.         playerMovement.enabled = false;
54.         GetComponent<Rigidbody>().isKinematic = true;
55.     }
56. }
```

Kode Sumber 4.9 Implementasi Player Health

Kode Sumber 4.9 berfungsi mengatur *Health Point* awal pemain, mekanisme jika terserang musuh, dan juga saat pemain mati kehabisan *Health Point*. Jumlah *Health Point* awal telah dideklarasikan sebelumnya. Jika pemain terkena serangan musuh, maka *Health Point* pemain akan berkurang sejumlah *damage* yang dihasilkan oleh objek musuh tersebut. Apabila *Health Point* pemain mencapai nol, maka pemain dinyatakan mati dan kalah. Semua fungsi pada *game obeject player* akan dinonaktifkan jika pemain mati.

4.2.8. Implementasi Objek Musuh

Game object enemy atau musuh pada permainan Nightmare: Arachnophobia ini memiliki komponen berupa *enemy health*, *enemy movement*, *enemy attack*, *dan enemy manager*.

Enemy health menangani darah atau *Health Point* dari objek musuh. Implementasi dari *enemy health* dijelaskan pada Kode Sumber 4.10.

```
1.  using UnityEngine;
2.
3.  public class EnemyHealth : MonoBehaviour
4.  {
5.      public float startingHealth = 50;
6.      public float currentHealth;
7.      public float sinkSpeed = 2.5f;
8.      public int scoreValue = 1;
9.      public AudioClip deathClip;
10.
11.     Animator anim;
12.     CapsuleCollider capsuleCollider;
13.     AudioSource enemyAudio;
14.
15.     bool isDead;
16.     bool isSinking;
17.
18.     void Awake()
19.     {
20.         anim = GetComponent<Animator>();
21.         capsuleCollider = GetComponent<CapsuleCol
    lider>();
22.         enemyAudio = GetComponent<AudioSource>();

23.         currentHealth = startingHealth;
24.     }
25.
26.     void Update()
27.     {
28.         if (isSinking)
29.         {
30.             transform.Translate(-
    Vector3.up * sinkSpeed * Time.deltaTime);
31.         }
32.     }
33.
34.     public void TakeDamage(float amount)
35.     {
```

```
36.         if (isDead)
37.             return;
38.
39.         enemyAudio.Play();
40.         currentHealth -= amount;
41.
42.         if (currentHealth <= 0)
43.         {
44.             Death();
45.         }
46.     }
47.
48.     void Death()
49.     {
50.         isDead = true;
51.         capsuleCollider.isTrigger = true;
52.         anim.SetTrigger("Dead");
53.         enemyAudio.clip = deathClip;
54.         enemyAudio.Play();
55.         TotalKill.totalKill += scoreValue;
56.
57.     }
58.
59.     public void StartSinking()
60.     {
61.         GetComponent<UnityEngine.AI.NavMeshAgent>
    ().enabled = false;
62.         GetComponent<Rigidbody>().isKinematic = t
    rue;
63.         isSinking = true;
64.         Destroy(gameObject, 2f);
65.     }
66. }
```

Kode Sumber 4.10 Implementasi Enemy Health

Kode Sumber 4.10 berfungsi mengatur *Health Point* awal setiap musuh. Jumlah *Health Point* awal telah dideklarasikan sebelumnya. Jika objek musuh terkena serangan pemain, maka *Health Point* musuh akan berkurang sejumlah *damage* yang

dihasilkan oleh pemain. Apabila *Health Point* mencapai nol, maka objek musuh akan mati dan dihancurkan.

Enemy movement menangani pergerakan objek musuh. Implementasi dari *enemy movement* dijelaskan pada Kode Sumber 4.11.

```
1.  using UnityEngine.AI;
2.  using UnityEngine;
3.
4.  public class Spider_movement : MonoBehaviour
5.  {
6.      Transform player;
7.      NavMeshAgent nav;
8.      PlayerHealth playerHealth;
9.      EnemyHealth enemyHealth;
10.
11.     void Start()
12.     {
13.         player = GameObject.FindWithTag ("Player"
    ).transform;
14.         playerHealth = player.GetComponent <Playe
    rHealth> ();
15.         enemyHealth = GetComponent <EnemyHealth>
    ();
16.         nav = GetComponent<NavMeshAgent> ();
17.     }
18.
19.     void Update()
20.     {
21.         if(enemyHealth.currentHealth > 0 && playe
    rHealth.currentHealth > 0)
22.         {
23.             nav.SetDestination(player.position);

24.         }
25.
26.         else
27.         {
28.             nav.enabled = false;
29.         }
```

```
30.     }
31. }
```

Kode Sumber 4.11 Implementasi Enemy Movement

Pada Kode Sumber 4.11 dijelaskan objek musuh akan mengejar *game object* dengan tag "*Player*" selama pemain memiliki *Health Point.* Musuh akan berhenti mengejar jika pemain mati. Musuh bergerak berdasarkan AI dari Unity menggunakan *Navigation Destination* yang sudah diatur terlebih dahulu. Hal ini membuat objek musuh mengejar pemain menggunakan rute terpendek. Namun akan memutari jalan yang tidak bisa dilewati karena perbedaan ketinggian yang terlalu jauh atau ada objek lain yang menghalangi.

Enemy attack menangani objek musuh dalam menyerang pemain. Implementasi dari *enemy attack* dijelaskan pada Kode Sumber 4.12.

```
1.  using UnityEngine;
2.
3.  public class EnemyAttack : MonoBehaviour
4.  {
5.      public float timeBetweenAttacks = 0.5f;
6.      public int attackDamage = 10;
7.
8.      Animator anim;
9.      GameObject player;
10.     PlayerHealth playerHealth;
11.     EnemyHealth enemyHealth;
12.
13.     bool playerInRange;
14.     float timer;
15.
16.     void Awake ()
17.     {
18.         player = GameObject.FindGameObjectWithTag
    ("Player");
```

```
19.         playerHealth = player.GetComponent <Playe
    rHealth> ();
20.         enemyHealth = GetComponent<EnemyHealth>()
    ;
21.         anim = GetComponent <Animator> ();
22.     }
23.
24.     void OnTriggerEnter (Collider other)
25.     {
26.         if(other.gameObject == player)
27.         {
28.             playerInRange = true;
29.         }
30.     }
31.
32.     void OnTriggerExit (Collider other)
33.     {
34.         if(other.gameObject == player)
35.         {
36.             playerInRange = false;
37.         }
38.     }
39.
40.     void Update ()
41.     {
42.         timer += Time.deltaTime;
43.
44.         if(timer >= timeBetweenAttacks && playerI
    nRange && enemyHealth.currentHealth > 0)
45.         {
46.             Attack ();
47.         }
48.     }
49.
50.     void Attack ()
51.     {
52.         timer = 0f;
53.
54.         if(playerHealth.currentHealth > 0)
55.         {
56.             playerHealth.TakeDamage (attackDamage
    );
```

```
57.         }
58.     }
59. }
```

Kode Sumber 4.12 Implementasi Enemy Attack

Cara kerja Kode Sumber 4.12 adalah objek musuh akan menyerang pemain jika *collider* musuh bersentuhan dengan objek pemain. *Collider* musuh ini diatur sebagai *trigger* yang memanggil fungsi **OnTriggerEnter().** Jika pemain dalam jarak serang dan masih memiliki *Health Point* yang tersisa, maka musuh akan menyerang dan memberikan *damage* ke pemain.

Enemy manager menangani kapan dan dimana objek musuh muncul. Implementasi dari *enemy manager* dijelaskan pada Kode Sumber 4.13.

```
1.  using UnityEngine;
2.
3.  public class EnemyManager : MonoBehaviour
4.  {
5.      public PlayerHealth playerHealth;
6.      public GameObject enemy;
7.      public float spawnTime = 15f;
8.      public Transform[] spawnPoints;
9.
10.     void Start ()
11.     {
12.         InvokeRepeating ("Spawn", spawnTime, spaw
    nTime);
13.     }
14.
15.     void Spawn ()
16.     {
17.         if(playerHealth.currentHealth <= 0f)
```

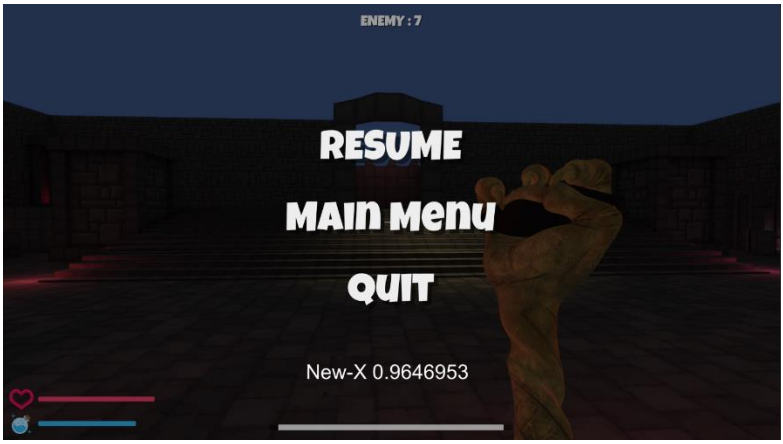
```
18.         {
19.             return;
20.         }
21.
22.         int spawnPointIndex = Random.Range (0, sp
    awnPoints.Length);
23.         Instantiate (enemy, spawnPoints[spawnPoin
    tIndex].position, spawnPoints[spawnPointIndex].ro
    tation);
24.     }
25. }
```

Kode Sumber 4.13 Implementasi Enemy Manager

Pada Kode Sumber 4.13 dijelaskan bahwa objek musuh akan dimuculkan setiap sekian detik dan pada lokasi yang ditentukan. Kemunculan objek musuh ini dilakukan berulang kali hingga permainan selesai.

4.2.9. Implementasi Tampilan Pause Menu

Pada Gambar 4.11 ditampilkan hasil implementasi dari tampilan *Pause Menu*.



Gambar 4.12 Implementasi Tampilan Pause Menu

Tampilan *Pause Menu* akan muncul jika pemain menekan tombol *escape* pada *keyboard.* Berikut penjelasan dari konten *Pause Menu* :

1. Tombol **Resume,** berfungsi untuk melanjutkan permainan kembali.
2. Tombol **Main Menu**, berfungsi untuk kembali ke halaman menu utama.
3. Tombol **Quit**, berfungsi untuk keluar dari permainan dan permainan akan tertutup.

Pada Kode Sumber 4.14 dijelaskan mengenai fungsi yang digunakan pada implementasi tampilan *Pause Menu.*

```
1.  using UnityEngine;
2.  using UnityEngine.SceneManagement;
3.
4.
5.  public class PauseMenu : MonoBehaviour
6.  {
7.      public static bool GameIsPaused = false;
8.      public GameObject pauseMenuUI;
9.
```

```
10.     void Update()
11.     {
12.         if (Input.GetKeyDown(KeyCode.Escape))
13.         {
14.             if (GameIsPaused)
15.             {
16.                 Resume();
17.             }
18.             else
19.             {
20.                 Pause();
21.             }
22.         }
23.     }
24.
25.     public void Resume()
26.     {
27.         pauseMenuUI.SetActive(false);
28.         Cursor.lockState = CursorLockMode.Locked;

29.         Cursor.visible = false;
30.         Time.timeScale = 1f;
31.         GameIsPaused = false;
32.     }
33.
34.     void Pause()
35.     {
36.         pauseMenuUI.SetActive(true);
37.         Cursor.lockState = CursorLockMode.None;
38.         Cursor.lockState = CursorLockMode.Confine
    d;
39.         Cursor.visible = true;
40.         Time.timeScale = 0f;
41.         GameIsPaused = true;
42.     }
43.
44.     public void MainMenu()
45.     {
46.         Time.timeScale = 1f;
47.         SceneManager.LoadScene("MainMenu");
48.     }
49.
```

```
50.     public void QuitGame()
51.     {
52.         Debug.Log("Quit Game");
53.         Application.Quit();
54.     }
55. }
```

Kode Sumber 4.14 Implementasi Tampilan Pause Menu

4.2.10. Implementasi Tampilan Next Level

Pada Gambar 4.12 ditampilkan hasil implementasi dari tampilan *Next Level*.

Gambar 4.13 Implementasi Tampilan Next Level

Tampilan *Next Level* ini adalah tampilan yang muncul saat pemain berhasil mengalahkan laba-laba sebanyak yang ditargetkan pada *stage* tersebut. Terdapat tombol *continue* yang mengarahkan pemain ke *stage* berikutnya. Pada Kode Sumber 4.15 dijelaskan mengenai fungsi pada implementasi tampilan *Next Level.*

```
1.  using System.Collections;
```

```
2.  using UnityEngine;
3.  using UnityEngine.SceneManagement;
4.
5.  public class NextLevelManager : MonoBehaviour
6.  {
7.      public int killTarget = 10;
8.      public GameObject NextLevelUI;
9.      Scene currentScene;
10.
11.     void Start()
12.     {
13.         currentScene = SceneManager.GetActiveScen
    e();
14.     }
15.
16.     void Update()
17.     {
18.         if (TotalKill.totalKill == killTarget)
19.         {
20.             StartCoroutine(NextLevel());
21.         }
22.     }
23.
24.     public IEnumerator NextLevel()
25.     {
26.         yield return new WaitForSeconds(2);
27.         Cursor.lockState = CursorLockMode.None;
28.         Cursor.lockState = CursorLockMode.Confine
    d;
29.         Cursor.visible = true;
30.         NextLevelUI.SetActive(true);
31.         Time.timeScale = 0f;
32.     }
33.
34.     public void NextLvl()
35.     {
36.         SceneManager.LoadScene(currentScene.build
    Index + 1);
37.         Time.timeScale = 1f;
38.     }
39. }
```

Kode Sumber 4.15 Implementasi Tampilan Next Level

4.2.11. Implementasi Tampilan Akhir Permainan

Pada Gambar 4.13 ditampilkan hasil implementasi dari tampilan *Next Level*.

Gambar 4.14 Implementasi Tampilan Akhir Permainan

Tampilan akhir permainan adalah tampilan yang muncul ketika *Health Point* pemain mencapai nol yang artinya pemain kalah. Terdapat dua tombol pada tampilan ini yaitu :

1. Tombol **Restart,** berfungsi untuk memulai permainan kembali pada *stage* saat ini.
2. Tombol **Quit**, berfungsi untuk keluar dari permainan dan permainan akan tertutup.

Pada Kode Sumber 4.16 dijelaskan mengenai fungsi pada implementasi tampilan akhir permainan.

```
1.  using UnityEngine;
2.  using UnityEngine.SceneManagement;
3.
```

```
4.  public class GameOverManager : MonoBehaviour
5.  {
6.      public PlayerHealth playerHealth;
7.      public GameObject GameOverUI;
8.
9.      void Update()
10.     {
11.         if (playerHealth.currentHealth <= 0)
12.         {
13.             Cursor.lockState = CursorLockMode.Non
    e;
14.             Cursor.lockState = CursorLockMode.Con
    fined;
15.             Cursor.visible = true;
16.             GameOverUI.SetActive(true);
17.         }
18.     }
19.
20.     public void Restart()
21.     {
22.         SceneManager.LoadScene(SceneManager.GetAc
    tiveScene().name);
23.     }
24.
25.     public void QuitGame()
26.     {
27.         Debug.Log("Quit Game");
28.         Application.Quit();
29.     }
30. }
```

Kode Sumber 4.16 Implementasi Tampilan Akhir Permainan

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas mengenai rangkaian uji coba dan evaluasi yang dilakukan terhadap permainan Nightmare: Arachnophobia. Proses pengujian dilakukan dengan menggunakan metode *blackbox* berdasarkan skenario yang telah ditentukan.

5.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan tugas akhir ini dilakukan pada lingkungan dan alat kakas dapat dilihat di Tabel 5.1. di bawah ini.

Tabel 5.1 Tabel Lingkungan Pengujian Sistem

Perangkat	Spesifikasi
Perangkat Keras	• Perangkat Pengembangan Sistem: AMD FX-9830P RADEON R7 3.0 GHz, AMD Radeon™ RX 460 4GB, RAM 16GB.
Perangkat Lunak	• Sistem Operasi: Microsoft Windows 10 64-bit • Perangkat Pengembang: Unity 2019.2.6f1 (64-bit) dan Visual Studio Community 2019. • Perangkat Pembantu: Microsoft Word 2019, Corel Draw X8, Adobe Photoshop CC 2015, dan Blender.

5.2. Pengujian Sistem

Pengujian fungsionalitas sistem dilakukan dengan menyiapkan sejumlah skenario sebagai tolok ukur keberhasilan pengujian. Pengujian fungsionalitas dilakukan untuk menguji apakah fungsionalitas yang diidentifikasi benar-benar diimplementasikan dan bekerja sebagaimana seharusnya.

Pengujian fungsionalitas yang terdapat pada permainan dijabarkan sebagai berikut.

5.2.1. Uji Coba Menu Utama

Pada sub bab ini dijelaskan secara detil mengenai skenario yang dilakukan dan hasil yang didapatkan dari pengujian fungsionalitas perangkat lunak yang dibangun pada halaman awal. Penjelasan disajikan dengan menampilkan kondisi awal, masukan, keluaran, hasil yang dicapai, dan kondisi akhir.

Pada menu permainan yang akan diuji adalah fungsionalitas tombol yang terdapat di menu utama, yaitu tombol *Story Mode*, *Quick Play*, *How to Play*, dan *Quit*. Tampilan menu permainan dapat dilihat pada Gambar 4.1.

Pengujian dimulai ketika pengguna membuka permainan yang kemudian dihadapkan dengan halaman menu awal permainan. Skenario pengujian yang dilakukan dapat dilihat pada Tabel 5.2.

Tabel 5.2 Pengujian Halaman Menu Permainan

Kode Uji	UF-001
Deskripsi	Menguji antarmuka yang terdapat pada halaman *Main Menu*
Kondisi Awal	Pengguna berada pada posisi awal permainan yaitu halaman utama
Prosedur Pengujian	• Pemain memilih tombol **Story Mode** • Pemain memilih tombol **Quick Play** • Pemain memilih tombol **How To Play** • Pemain memilih tombol **Quit**
Skenario 1 *– Memilih tombol* ***Story Mode***	
Masukan	Menekan tombol **Story Mode** pada *Main Menu* menggunakan *mouse*
Hasil yang Diharapkan	Sistem memulai permainan pada posisi awal bermain di *stage* village day

Hasil yang Diperoleh	Sistem memulai permainan pada posisi awal bermain di *stage* village day
Kesimpulan	Berhasil
Skenario 2 – *Memilih tombol* ***Quick Play***	
Masukan	Menekan tombol **Quick Play** pada *Main Menu* menggunakan *mouse*
Hasil yang Diharapkan	Sistem menampilkan daftar *stage*. Setelahnya sistem memulai permainan pada posisi awal bermain di *stage* yang dipilih.
Hasil yang Diperoleh	Sistem menampilkan daftar *stage*. Kemudian sistem memulai permainan pada posisi awal di *stage* yang dipilih.
Kesimpulan	Berhasil
Skenario 3 – *Memilih tombol* ***How to Play***	
Masukan	Menekan tombol **How to Play** pada *Main Menu* menggunakan *mouse*
Hasil yang Diharapkan	Sistem menampilkan halaman *How to Play*
Hasil yang Diperoleh	Sistem menampilkan halaman *How to Play*
Kesimpulan	Berhasil
Skenario 4 – *Memilih tombol* ***Quit***	
Masukan	Menekan tombol **Quit** pada *Main Menu* menggunakan *mouse*
Hasil yang Diharapkan	Sistem menutup permainan dan pemain keluar dari permainan
Hasil yang Diperoleh	Sistem menutup permainan dan pemain keluar dari permainan
Kesimpulan	Berhasil

5.2.2. Uji Coba Mekanisme Permainan

Pada sub bab ini akan dijelaskan mengenai skenario pengujian mekanisme permainan yang meliputi kontrol bergerak, serangan dasar, menggambar pola, aktivasi *Weapon Skills*, dan terkena serangan musuh.

Pengujian dimulai setelah pemain menekan tombol Story Mode pada halaman menu utama. Skenario pengujian yang dilakukan dapat dilihat pada Tabel 5.3.

Tabel 5.3 Pengujian Mekanisme Permainan

Kode Uji	UF-002
Deskripsi	Menguji mekanisme permainan
Kondisi Awal	Pengguna berada pada posisi awal bermain setelah menekan tombol Story Mode
Prosedur Pengujian	• Pemain begerak menggunakan *keyboard* • Pemain melakukan serangan dasar dengan menggunakan *mouse* • Pemain menggambar pola pada layar permainan menggunakan *mouse* • Pemain mengaktifkan *Weapon Skills* • Pemain terkena serangan musuh
Skenario 1** – Bergerak menggunakan **Keyboard	
Masukan	Menekan tombol W, A, S, dan D pada *keyboard*. Menekan tombol *Shift* pada *keyboard*
Hasil yang Diharapkan	• Pemain dapat bergerak maju, mundur, ke kiri, dan ke kanan • Pemain dapat berlari
Hasil yang Diperoleh	• Pemain dapat bergerak maju, mundur, ke kiri, dan ke kanan • Pemain dapat berlari
Kesimpulan	Berhasil

Skenario 2** – Pemain melakukan serangan dasar dengan menggunakan **Mouse	
Masukan	Pemain menekan tombol *mouse* sebelah kiri (LMB)
Hasil yang Diharapkan	Pemain melakukan serangan dasar sesuai dengan senjata yang aktif dan memberikan *damage* kepada musuh.
Hasil yang Diperoleh	Pemain melakukan serangan dasar sesuai dengan senjata yang aktif dan memberikan *damage* kepada musuh.
Kesimpulan	Berhasil
Skenario 3** – Pemain menggambar pola pada layar permainan menggunakan **Mouse	
Masukan	Pemain menekan tombol *mouse* sebelah kanan (RMB) untuk menggabar pola pada layar permainan
Hasil yang Diharapkan	Muncul *Line Renderer* sebagai bentuk yang digambar oleh pemain
Hasil yang Diperoleh	Muncul *Line Renderer* sebagai bentuk yang digambar oleh pemain
Kesimpulan	Berhasil
Skenario 4** – Pemain mengaktifkan **Weapon Skills	
Masukan	Pemain menekan tombol *space* pada *keyboard* untuk mengkonfirmasi pola yang telah digambar
Hasil yang Diharapkan	Bentuk pola garis berhasil dikenali dan *Weapon Skill* tertentu aktif sesuai dengan gambar pola yang dibuat pemain
Hasil yang Diperoleh	Bentuk pola garis berhasil dikenali dan *Weapon Skill* tertentu aktif sesuai dengan gambar pola yang dibuat pemain
Kesimpulan	Berhasil

Skenario 5 *– Pemain terkena serangan musuh*	
Masukan	Pemain bergerak menyentuh laba-laba
Hasil yang Diharapkan	*Health Point* pemain berkurang sebanyak *damage* dari serangan laba-laba
Hasil yang Diperoleh	*Health Point* pemain berkurang sebanyak *damage* dari serangan laba-laba
Kesimpulan	Berhasil

5.2.3. Uji Coba Pause Menu

Pada sub bab ini akan dijelaskan mengenai skenario pengujian pada tampilan *Pause Menu*. Pemain dapat menjeda permainan dengan menekan tombol *escape* pada *keyboard*.

Pengujian dilakukan saat pemain sedang berada di tengah permainan. Skenario pengujian dapat dilihat pada Tabel 5.4.

Tabel 5.4 Pengujian Tampilan Pause Menu

Kode Uji	UF-003
Deskripsi	Menguji tampilan *Pause Menu*
Kondisi Awal	Pengguna sedang berada di tengah bermain
Prosedur Pengujian	• Pemain menjeda permainan • Pemain menekan tombol **Resume** • Pemain menekan tombol **Main Menu** • Pemain menekan tombol **Quit**
Skenario 1 *– Pemain menjeda permainan*	
Masukan	Menekan tombol **Escape** pada *keyboard*
Hasil yang Diharapkan	• Permainan dijeda • Pemain tidak bisa bergerak, menyerang, ataupun menggambar pola. • Musuh tidak bisa bergerak atau menyerang pemain

Hasil yang Diperoleh	• Permainan dijeda • Pemain tidak bisa bergerak, menyerang, ataupun menggambar pola • Musuh tidak bisa bergerak atau menyerang pemain
Kesimpulan	Berhasil
Skenario 2 *– Menekan tombol* ***Resume***	
Masukan	Menekan tombol **Resume** pada *Pause Menu* menggunakan *mouse*
Hasil yang Diharapkan	Sistem menutup tampilan *Pause Menu* dan permainan dilanjutkan
Hasil yang Diperoleh	Sistem menutup tampilan *Pause Menu* dan permainan dilanjutkan
Kesimpulan	Berhasil
Skenario 3 *– Menekan tombol* ***Main Menu***	
Masukan	Menekan tombol **Main Menu** pada *Pause Menu* menggunakan *mouse*
Hasil yang Diharapkan	Sistem menampilkan halaman menu utama
Hasil yang Diperoleh	Sistem menampilkan halaman menu utama
Kesimpulan	Berhasil
Skenario 4 *– Memilih tombol* ***Quit***	
Masukan	Menekan tombol **Quit** pada *Pause Menu* menggunakan *mouse*
Hasil yang Diharapkan	Sistem menutup permainan dan pemain keluar dari permainan
Hasil yang Diperoleh	Sistem menutup permainan dan pemain keluar dari permainan
Kesimpulan	Berhasil

5.2.4. Uji Coba Akhir Permainan

Pada sub bab ini dijelaskan mengenai skenario penjelasan pada tampilan akhir permainan atau *Game Over*. Tampilan *Game Over* akan muncul ketika *Health Point* dari pemain telah habis. Pada tampilan ini tersedia dua buah tombol berupa tombol *Restart* dan tombol *Quit*.

Pemain dapat memulai kembali permainan dengan menekan tombol *Restart* atau pemain dapat keluar dan menutup permainan dengan menekan tombol *Quit*. Skenario pengujian dapat dilihat pada Tabel 5.5.

Tabel 5.5 Pengujian Tampilan Akhir Permainan

Kode Uji	UF-004
Deskripsi	Menguji antarmuka yang terdapat pada halaman akhir permainan atau *Game Over*
Kondisi Awal	Pemain berada pada kondisi *Health Point* habis karena serangan musuh.
Prosedur Pengujian	• Pemain menekan tombol **Restart** • Pemain menekan tombol **Quit**
Skenario 1 *– Menekan tombol* ***Restart***	
Masukan	Menekan tombol **Restart** pada *Game Over Menu* menggunakan *mouse*
Hasil yang Diharapkan	Permainan akan dimulai kembali pada posisi awal bermain di *stage* saat ini
Hasil yang Diperoleh	Permainan akan dimulai kembali pada posisi awal bermain di *stage* saat ini
Kesimpulan	Berhasil
Skenario 2 *– Memilih tombol* ***Quit***	
Masukan	Menekan tombol **Quit** pada *Game Over Menu* menggunakan *mouse*
Hasil yang Diharapkan	Sistem menutup permainan dan pemain keluar dari permainan

Hasil yang Diperoleh	Sistem menutup permainan dan pemain keluar dari permainan
Kesimpulan	Berhasil

5.2.5. Hasil Uji Coba

Pada bagian ini dijelaskan mengenai hasil evaluasi dari pengujian yang dilakukan pada permainan. Hasil evaluasi dapat dilihat pada Tabel 5.6.

Tabel 5.6 Hasil Evaluasi Uji Coba

Kode	Deskripsi	Kemungkinan / Skenario	Perilaku Terlaksana
UF-001	Uji Coba Menu Permainan	Skenario 1	Ya
		Skenario 2	Ya
		Skenario 3	Ya
		Skenario 4	Ya
UF-002	Uji Coba Mekanisme Permainan	Skenario 1	Ya
		Skenario 2	Ya
		Skenario 3	Ya
		Skenario 4	Ya
		Skenario 5	Ya
UF-003	Uji Coba Tampilan Pause Menu	Skenario 1	Ya
		Skenario 2	Ya
		Skenario 3	Ya
		Skenario 4	Ya
UF-004	Uji Coba Tampilan Akhir Permainan	Skenario 1	Ya
		Skenario 2	Ya

5.3. Pengujian Subjektivitas Sistem

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

5.3.1. Skenario Pengujian Pengguna

Dalam melakukan pengujian permainan, pengguna diminta mencoba memainkan permainan untuk mencoba semua fungsionalitas dan fitur yang ada. Pengujian permainan oleh pengguna dilakukan dengan sebelumnya memberikan informasi seputar permainan, kegunaan, dan fitur-fitur yang dimiliki. Setelah informasi tersampaikan, pengguna kemudian diarahkan untuk langsung mencoba permainan. Jumlah pengguna yang terlibat dalam pengujian perangkat sebanyak 30 orang. Perangkat yang digunakan dalam pengujian adalah perangkat yang dimiliki oleh masing-masing pengguna, namun ada juga penguji yang meminjam perangkat yang dimiliki penulis.

Dalam memberikan penilaian dan tanggapan, pengguna diberikan kuesioner pengujian permainan. Kuesioner pengujian ini dilakukan dengan mengisi form kuesioner yang telah diberikan. Kuesioner pengujian ini memiliki beberapa aspek penilaian seputar permainan. Nilai yang diberikan rentang nilai 1 hingga 5 dengan rincian pada Tabel 5.7. Pada bagian akhir kuisioner, terdapat saran dari pengguna untuk perbaikan aspek dan fitur dalam permainan. Kuisioner mengenai karakteristik pengguna dapat dilihat pada Tabel 5.8, sedangkan kuisioner penilaian permainan dapat dilihat pada Tabel 5.9.

Tabel 5.7 Rentang Nilai

No	Keterangan	Nilai
1	Sangat Tidak Setuju (STS)	1
2	Tidak Setuju (TS)	2
3	Netral (N)	3
4	Setuju (S)	4
5	Sangat Setuju (SS)	5

Tabel 5.8 Kuesioner Mengenai Karakteristik Pengguna

No	Pertanyaan Karakteristik Pengguna
1	Pernahkah anda memainkan sebuah permainan digital atau *game* di Komputer ?
2	Pernahkah anda memainkan permainan dengan genre Survival ?
3	Pernahkah anda memainkan permainan *First Person Shooter* ?
4	Apakah anda memiliki fobia laba-laba atau Arachnophobia ?

Tabel 5.9 Kuisioner Penilaian Permainan

No	Parameter	STS	TS	N	S	SS
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.					
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.					

3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.					
4	Saya merasakan ketegangan seperti dikejar laba-laba raksasa.					
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.					
6	Saya merasa belum pernah memainkan permainan *genre survival* dengan *gameplay* yang seperti ini.					
7	Aplikasi dapat berjalan dengan lancar tanpa adanya *lag* dan/*atau crash.*					
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.					
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.					
10	Saya merasa mudah melakukan serangan dasar.					

11	Saya merasa mudah menggambar pola dan mengaktifkan *Weapon Skill*.					
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.					

5.3.2. Tujuan Pertanyaan Kuisioner

Kuisioner ini dibuat untuk mengetahui tanggapan dari pengguna dan untuk mengukur apakah permainan Nightmare: Arachnophobia sudah berjalan sesuai dengan rancangan. Tujuan dari pertanyaan kuisioner dapat dilihat pada Tabel 5.10.

Tabel 5.10 Tujuan Pertanyaan Kuisioner

No	Isi Pertanyaan	Tujuan Pertanyaan
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.	Untuk mengetahui apakah permainan memiliki tampilan antarmuka yang baik
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.	Untuk mengetahui apakah permainan memiliki tampilan antarmuka yang baik
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.	Untuk mengetahui apakah permainan memiliki grafik visual yang baik

4	Saya merasakan ketegangan dikejar laba-laba raksasa.	Untuk mengetahui apakah permainan memiliki grafik visual yang baik
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.	Untuk mengetahui apakah permainan telah memenuhi manfaat yang ingin dicapai
6	Saya merasa belum pernah memainkan permainan *genre survival* dengan *gameplay* yang seperti ini.	Untuk mengetahui apakah permainan memberikan pengalaman bermain baru kepada pengguna
7	Aplikasi dapat berjalan dengan lancar tanpa adanya *lag* dan/*atau crash*.	Untuk mengetahui apakah permainan berjalan dengan lancar
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.	Untuk mengetahui apakah petunjuk permainan dapat mempermudah pengguna dalam bermain
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.	Untuk mengetahui apakah pengguna merasa nyaman ketika bermain
10	Saya merasa mudah melakukan serangan dasar.	Untuk mengetahui apakah pengguna

		merasa nyaman ketika bermain
11	Saya merasa mudah menggambar pola dan mengaktifkan *Weapon Skill.*	Untuk mengetahui apakah pengguna merasa nyaman ketika bermain
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.	Untuk mengetahui tingkat ketertarikan pengguna dalam bermain permainan ini

5.3.3. Daftar Penguji Permainan

Sub bab ini menunjukkan daftar pengguna yang bertindak sebagai penguji coba Nightmare: Arachnophobia. Dalam pengujian ini tidak terdapat kriteria atau batas usia yang harus dimiliki pengguna, karena aplikasi ini ditujukan kepada berbagai kalangan tanpa batasan umur. Daftar nama penguji aplikasi ini dapat dilihat pada Tabel 5.11.

Tabel 5.11 Daftar Penguji

No	Nama	Pekerjaan	Usia	Perangkat Pengujian
1	Yura Anugrah Mursyd	Mahasiswa	23	AMD FX-9830P R7, RX460, RAM 16 GB
2	Mundzir Al-khotiby	Mahasiswa	22	AMD FX-9830P R7, RX460, RAM 16 GB
3	Jojo I. Mangkulla	Mahasiswa	21	AMD FX-9830P R7, RX460, RAM 16 GB

4	Yogi Fajdwani	Mahasiswa	21	AMD FX-9830P R7, RX460, RAM 16 GB
5	Indri Tiffani	Pelajar	15	AMD FX-9830P R7, RX460, RAM 16 GB
6	M. Napissatrya	Wiraswasta	26	AMD FX-9830P R7, RX460, RAM 16 GB
7	Tri Kurniawaty	Guru	28	AMD FX-9830P R7, RX460, RAM 16 GB
8	Muhammad Raihan A.	Mahasiswa	21	Core i7 4720HQ, GT 940M, 8GB RAM
9	Ariq Sudibyo	Mahasiswa	21	Core i7-7700K, GTX 1060, RAM 16GB
10	Nada Nibrassalbila Rosadi	Mahasiswi	22	Intel Core i7 6700HQ, GTX 950M, RAM 8GB
11	Bernardino Adam Wijaya	Pelajar	18	Intel Pentium 4417U, Intel HD 610, RAM 4 GB
12	Aisyah Muswar	Mahasiswi	21	Core i7-4720HQ, NVIDIA GT 940M, RAM 4GB
13	Fariz Ardin Adhiyaksa	Mahasiswa	22	Intel Core i7 6700HQ, GTX 950M, RAM 8GB
14	Mohammad Nafis Naufally	Mahasiswa	21	AMD Ryzen 3550H, GTX 1650, 8GB RAM

15	Muhammad Fadhlan Min Robby	Mahasiswa	21	Intel I7-7660U, Intel Iris Plus 640, 16GB RAM
16	Falah Nurli Filano	Mahasiswa	21	i5-7200U, Geforce 920MX, 8GB RAM
17	Haidar Sakti Oktafiansyah	Mahasiswa	22	core i7-9750H, RTX2060 6GB, 16GB RAM
18	Muhammad Farhan Agus Trisasti	Mahasiswa	21	AMD FX-9830P R7, RX460, RAM 16 GB
19	Oktavian Rahman Koko	Mahasiswa	22	AMD FX-9830P R7, RX460, RAM 16 GB
20	Muh. Renaldi Aryaputra Wibowo	Mahasiswa	21	Core i7-6700HQ 3.60GHz, Nvidia 960M, 16GB
21	Ahmad Imam Fadhila	Mahasiswa	21	Intel i5-7200U, NVIDIA 940MX, 8GB RAM
22	Muhammad Zirkul Fahmi	Mahasiswa	22	Core i7-8750H, NVIDIA GTX 1060, Ram 12 GB
23	Steven Theofilus Sukertha	Mahasiswa	19	Core i5 6400, GTX 1050ti, 8GB
24	Adrian Hassan	Mahasiswa	23	AMD FX-9830P R7, RX460, RAM 16 GB
25	Sonia Titisan Ramdhani	Pelajar	16	AMD FX-9830P R7, RX460, RAM 16 GB

26	Kinantti rizkya	Pelajar	16	AMD FX-9830P R7, RX460, RAM 16 GB
27	Keke Aulia Asmiey Mangkulla	Mahasiswi	19	AMD A10-8700P, AMD R7, RAM 4 GB
28	Rifda Shofiyya Ardini	Mahasiswi	18	Core i7 6700HQ, GTX 950M, 8GB RAM
29	Yordan Wiguna	Mahasiswa	22	Core i7-8750H, GTX 1060, 12GB RAM
30	Akhmad Farkhan Khasyim	Mahasiswa	22	Ryzen 5 3550H, GTX 1050, 8GB RAM

5.3.4. Hasil Pengujian Pengguna

Berdasarkan hasil kuesioner yang sudah diisi oleh penguji, maka yang pertama didapatkan ialah karakteristik pengguna, yaitu sebagai berikut:

- Semua penguji pernah bermain permainan digital pada komputer (30/30).
- Semua penguji pernah memainkan permainan dengan genre Survival (28/30).
- Mayoritas penguji pernah memainkan permainan *First Person Shooter* (27/30).
- Terdapat 9 penguji yang memiliki fobia terhadap laba-laba atau Arachnophobia.

Kemudian pengujian terhadap aspek antarmuka, visual, *gameplay*, dan tingkat kenyamanan pengguna mendapatkan hasil seperti yang ditunjukkan pada Tabel 5.12. Sistem penilaian yang digunakan yaitu dengan menjumlahkan seluruh nilai dari seluruh responden dengan kemudian dirata-rata dan dibagi dengan nilai

maksimum (5), sehingga didapatkan persentase nilai dari setiap parameter yang diujikan. Hasil akhir dari pengujian setiap aspek dapat dilihat pada Tabel 5.13.

Tabel 5.12 Hasil Pengujian Pengguna

No	Pernyataan	Penilaian					Rata-rata
		1	2	3	4	5	
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.	0	0	1	18	11	4,33
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.	0	0	6	13	11	4,16
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.	0	0	6	13	11	4,16
4	Saya merasakan ketegangan seperti dikejar laba-laba raksasa.	0	0	0	12	18	4,6
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.	0	0	4	22	4	4
6	Saya merasa belum pernah memainkan permainan *genre survival* dengan *gameplay* yang seperti ini.	0	0	9	11	10	4,03
7	Aplikasi dapat berjalan dengan lancar tanpa	0	0	1	14	15	4,46

	adanya *lag* dan/*atau crash*.						
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.	0	0	3	12	15	4,4
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.	0	0	6	18	6	4
10	Saya merasa mudah melakukan serangan dasar.	0	0	0	11	19	4,63
11	Saya merasa mudah menggambar pola dan mengaktifkan *Weapon Skill*.	0	0	8	16	6	3,93
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.	0	0	7	15	8	4,03

Tabel 5.13 Hasil Akhir Pengujian Pengguna

No	Pernyataan	Rata-rata	Total	Total (%)
Parameter Antarmuka				
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.	4,33	4,245	84,9%
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.	4,16		

Parameter Visual				
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.	4,16	4,38	87,6%
4	Saya merasakan ketegangan seperti dikejar laba-laba raksasa.	4,6		
Parameter GamePlay				
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.	4	4,015	80,3%
6	Saya merasa belum pernah memainkan permainan *genre survival* dengan *gameplay* yang seperti ini.	4,03		
Parameter Kenyamanan				
7	Aplikasi dapat berjalan dengan lancar tanpa adanya *lag* dan/*atau crash.*	4,46	4,24	84,8%
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.	4,4		
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.	4		
10	Saya merasa mudah melakukan serangan dasar.	4,63		
11	Saya merasa mudah menggambar pola dan mengaktifkan *Weapon Skill.*	3,93		
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.	4,03		

5.3.5. Kritik dan Saran Pengguna

Selain memberikan penilaian terhadap fungsionalitas aplikasi, pengguna juga memberikan tanggapan berupa kritik dan saran untuk pengembangan aplikasi selanjutnya. Kritik dan saran pengguna dapat dilihat pada Tabel 5.14.

Tabel 5.14 Kritik dan Saran Pengguna

No	Nama	Kritik dan Saran
1	Yura Anugrah Mursyd	Pola yang dibuat kadang tidak terbaca oleh *game*. Sebaiknya apabila salah pola, pengurangan *mana* sebesar 50%
2	Mundzir Al-khotiby	*Game* yang mudah cuman ada beberapa hal yang harus disempurnakan seperti sensitivitas
3	Jojo I. Mangkulla	Mungkin untuk sensitivitas *mouse*-nya terlalu tinggi ketika digunakan, sebaiknya bisa diturunkan sedikit agar lebih mudah dimainkan
4	Yogi Fajdwani	Permainan menarik dan tingkat kesulitan bisa disesuaikan
5	Indri Tiffani	Tingkat kesulitan dapat disesuaikan. Jadi bisa dipilih sesuai kemampuan
6	M. Napissatrya	Kualitas gambar kurang halus
7	Tri Kurniawaty	Kendali saat menggambar pola sedikit membingungkan jika tidak dijelaskan dahulu
8	Muhammad Raihan A.	Karena menggunakan *mouse*, penggambaran pola sedikit kaku dan tidak luwes
9	Ariq Sudibyo	*Movement speed* pada *stage* Railway terlalu pelan sebaiknya ditingkatkan

10	Nada Nibrassalbila Rosadi	*Stage* Forest laba-labanya tidak kelihatan karena rumputnya terlalu tinggi
11	Bernardino Adam Wijaya	Menggambar bentuk nya susah jika sambil dikejar laba-laba. Tapi seru gamenya
12	Aisyah Muswar	Ada beberapa mekanisme yang sulit dipahami jika tidak dijelaskan secara langsung
13	Fariz Ardin Adhiyaksa	Ditambah countdown bar setelah menggambar pola sebagai batas konfirmasi pattern. jika melebihi batas waktu, maka pola yang telah digambar sudah tidak valid lagi
14	Mohammad Nafis Naufally	Saran untuk laba-labanya ditambahkan dalam satu waktu sekaligus agar menambah tingkat kesulitan
15	Muhammad Fadhlan Min Robby	Ada stage yang laba-labanya terlalu lama spawn. Sebaiknya dipercepat agar pemain tidak kebingungan harus ngapain
16	Falah Nurli Filano	Laba-laba ada yang nyangkut di flatcar pada stage railway. Mungkin diatur lagi collider flatcar nya agar terdeteksi dengan baik oleh AI nya
17	Haidar Sakti Oktafiansyah	Cerita pada game nya terasa kurang menarik, seperti hanya pelengkap saja. Mungkin cerita akan lebih menari jika ada cutscene
18	Muhammad Farhan Agus Trisasti	Disarankan agar dapat menambah keanekaragaman warna laba-laba
19	Oktavian Rahman Koko	Mungkin sensitifitas mouse bisa lebih baik

20	Muh. Renaldi Aryaputra Wibowo	Bagus game nya, sangat realistis
21	Ahmad Imam Fadhila	Tampilan agak terlalu gelap sehingga sedikit sulit untuk melihat
22	Muhammad Zirkul Fahmi	tingkat kesulitan dibuat lebih menantang lagi
23	Steven Theofilus Sukertha	nteraksi pemain dan game dapat dikembangkan lagi. Jadi menambah konten atau gameplay dalam gamenya
24	Adrian Hassan	Menggambar menggunakan mouse susah, lebih enak kalau pakai controller
25	Sonia Titisan Ramdhani	Laba-laba nya besar-besar bikin takut, tapi keren gamenya
26	Kinantti rizkya	susah nginget bentuk gestur setiap senjatanya
27	Keke Aulia Asmiey Mangkulla	Sangat bagus. tapi terkadang ketika membuat huruf untuk special skill, sering tidak terbaca. Mungkin bisa diperbaiki
28	Rifda Shofiyya Ardini	Untuk kedepannya mungkin tambahkan item drop hasil membunuh laba-labanya
29	Yordan Wiguna	Kadang saat menyerang menggunakan sihir, projectile-nya naik ke atas tiba-tiba. sebaiknya diperbaiki agar tidak melenceng
30	Akhmad Farkhan Khasyim	Laba-labanya terlalu besar, bikin geli. Untuk saran sendiri agar GUI dibuat lebih menarik lagi

5.4. Evaluasi Pengujian

Sub bab ini membahas mengenai evaluasi terhadap pengujian-pengujian yang telah dilakukan. Dalam hal ini, evaluasi menunjukkan data rekapitulasi dari hasil pengujian fungsionalitas. Rekapitulasi disusun dalam bentuk tabel yang dapat dilihat pada Tabel 5.6. Dari data yang terdapat pada tabel tersebut, diketahui bahwa aplikasi yang dibuat telah berjalan sesuai dengan skenario yang diharapkan dan memenuhi semua fungsionalitasnya.

Dari hasil pengujian oleh pengguna ditemukan kendala dalam fungsionalitas permainan, dimana pemain kesulitan untuk menggambar pola garis dikarenakan beberapa bentuk pola sulit untuk digambar menggunakan *mouse*. Hal ini disebabkan karena keterbatasan gerakan yang dapat dibuat dengan *mouse* yang membuat pemain merasa kaku saat menggambar pola. Untuk bisa mendapatkan hasil yang optimal, pemilihan pola garis sebaiknya dipilih bentuk yang mudah digambar dengan *mouse*.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Bab ini akan membahas mengenai kesimpulan yang diperoleh selama pengerjaan tugas akhir dan saran mengenai 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 yang dilakukan, dapat diambil kesimpulan sebagai berikut.

1. Permainan berhasil menampilkan tampilan *First Person Shooter* (FPS) dalam lingkungan yang telah dibuat.
2. Permainan dapat mendeteksi bentuk pola garis yang dibuat oleh pemain menggunakan Point Cloud Recognition.
3. Permainan dapat mengaktivasi *Weapon Skills* berdasarkan bentuk pola garis yang terdeteksi dengan baik.
4. Permainan berhasil dibuat dengan *Game Engine* Unity 3D.
5. Berdasarkan hasil dari pengujian beta dan pengujian oleh pengguna dapat disimpulkan permainan Nightmare: Arachnophobia telah mengimplementasikan rancangan dengan baik.

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. Di antaranya adalah sebagai berikut:

1. Ditambahkan variasi model musuh yang ada pada permainan.
2. Cerita dalam permainan dipersiapkan dengan lebih baik lagi dengan menambahkan cerita pada setiap *stage*-nya agar alur cerita terasa lebih menarik.

3. Pergerakan kamera sebagai tampilan permainan FPS dibuat lebih halus agar mudah dikendalikan oleh pemain.
4. Mekanisme dalam menggambar pola untuk mengaktifkan *Weapon Skills* dibuat lebih *smooth* lagi agar pemain tidak kaku dalam menggambar. Untuk bisa mendapatkan hasil yang optimal, pemilihan pola garis sebaiknya dipilih bentuk yang mudah digambar dengan *mouse.*

DAFTAR PUSTAKA

[1] D. A. Pratama, Permainan Realitas Virtual Grand Wizard Menggunakan Oculus Rift dan Leap Motion Dengan Kendali Point Cloud Recognition, Surabaya: Institut Teknologi Sepuluh Nopember (ITS), 2020.

[2] B. D. Wicaksono, "Ini 10 Phobia Paling Umum di Masyarakat Berdasarkan Penelitian Ilmiah," www.idntimes.com, 25 November 2017. [Online]. Available: https://www.idntimes.com/science/discovery/bayu/phobia-yang-umum/full. [Accessed 11 November 2019].

[3] A. S. Sadiman, Media Pendidikan: Pengertian, Pengembangan, dan Pemanfaatannya, Jakarta: RajaGrafindo Persada, 1993.

[4] Unity, "Unity," Unity, [Online]. Available: https://unity3d.com/unity. [Accessed 14 November 2019].

[5] Unity, "Post-processing overview," Unity, 07 Mei 2019. [Online]. Available: https://docs.unity3d.com/Manual/PostProcessingOverview.html. [Accessed 22 November 2019].

[6] Wikipedia, "C sharp," Wikipedia, [Online]. Available: https://id.wikipedia.org/wiki/C_sharp. [Accessed 14 November 2019].

[7] Wikipedia, "First-person shooter," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/First-person_shooter. [Accessed 16 November 2019].

[8] Wikipedia, "Survival Game," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Survival_game. [Accessed 2020 Juni 16].

[9] Blender, "About Blender," Blender, [Online]. Available: https://www.blender.org/about/. [Accessed 14 November 2019].

[10] A. Lamboley, "PDollar Point-Cloud Gesture Recognizer," Da Viking Code, 02 Februari 2017. [Online]. Available: http://www.aymericlamboley.fr/blog/unity-p-dollar-gesture-recognizer/. [Accessed 29 November 2019].

[11] A. Praharsana, "Penerapan Teknologi Virtual Reality pada Perangkat Bergerak berbasis Android untuk Mendukung Terapi Fobia Laba-laba (Arachnophobia)," *JURNAL TEKNIK ITS*, vol. 4, no. 1, pp. A-129, 2015.

LAMPIRAN

Lampiran 1 Penguji Coba (1)

Lampiran 2 Penguji Coba (2)

Lampiran 3 Penguji Coba (3)

Lampiran 4 Penguji Coba (4)

Lampiran 5 Penguji Coba (5)

Lampiran 6 Penguji Coba (6)

Lampiran 7 Penguji Coba (7)

Lampiran 8 Penguji Coba (8)

Lampiran 9 Penguji Coba (9)

Lampiran 10 Penguji Coba (10)

KUISIONER TUGAS AKHIR

05111640000093 – *Muhammad Ivan Nianyah Putra*

PENERAPAN POINT-CLOUD GESTURE RECOGNIZER PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

Identitas Responden

Surabaya, 10 Juni 2020

Nama Lengkap : Yara Aminatul Muniyah

Pekerjaan : Mahasiswa

Usia : 23 tahun

A. KARAKTERISTIK RESPONDEN

Jawablah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

1. Pernahkah anda memainkan sebuah permainan digital atau *game* di Komputer?
   - ☑ Pernah
   - ○ Tidak Pernah
2. Pernahkah anda memainkan *game* dengan *genre Survival*?
   - ☑ Pernah
   - ○ Tidak Pernah
3. Pernahkah anda memainkan permainan *First Person Shooter*?
   - ☑ Pernah
   - ○ Tidak Pernah
4. Apakah anda memiliki fobia laba-laba atau Arachnophobia?
   - ☑ Punya
   - ○ Tidak Punya

B. PENILAIAN TERHADAP PERMAINAN

Isilah tabel dibawah ini dengan menggunakan tanda centang (v)

SS = Sangat Setuju *S = Setuju* *N = Netral*

TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter	STS	TS	N	S	SS
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.					✓
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.					✓
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.					✓
4	Saya merasakan ketegangan seperti dikejar laba-laba raksasa.					✓
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.				✓	
6	Saya merasa belum pernah memainkan permainan genre survival dengan gameplay yang seperti ini.			✓		
7	Aplikasi dapat berjalan dengan lancar tanpa adanya lag dan/atau crash.					✓
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.					✓
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.					✓
10	Saya merasa mudah melakukan serangan dasar.					✓
11	Saya merasa mudah menggambar gestur dan mengaktifkan Weapon Skill.				✓	
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.				✓	

C. KRITIK DAN SARAN

tekstur e yang dibuat kadang kadang kebaca oleh game, sebaiknya objek salah gestuge pengurangan mana sekitar 50%

KUISIONER TUGAS AKHIR

05111640000093 – *Muhammad Ivan Nianyah Putra*

PENERAPAN POINT-CLOUD GESTURE RECOGNIZER PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

Identitas Responden

Surabaya, 11 Juni 2020

Nama Lengkap : Munzir Al-Hadiqy

Pekerjaan : Mahasiswa

Usia : 22

Munzir Al-Hadiqy

A. KARAKTERISTIK RESPONDEN

Jawablah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

1. Pernahkah anda memainkan sebuah permainan digital atau *game* di Komputer?
   - ☑ Pernah
   - ○ Tidak Pernah
2. Pernahkah anda memainkan *game* dengan *genre Survival*?
   - ☑ Pernah
   - ○ Tidak Pernah
3. Pernahkah anda memainkan permainan *First Person Shooter*?
   - ☑ Pernah
   - ○ Tidak Pernah
4. Apakah anda memiliki fobia laba-laba atau Arachnophobia?
   - ○ Punya
   - ☑ Tidak Punya

B. PENILAIAN TERHADAP PERMAINAN

Isilah tabel dibawah ini dengan menggunakan tanda centang (v)

SS = Sangat Setuju *S = Setuju* *N = Netral*

TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter	STS	TS	N	S	SS
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.				✓	
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.				✓	
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.				✓	
4	Saya merasakan ketegangan seperti dikejar laba-laba raksasa.					✓
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.				✓	
6	Saya merasa belum pernah memainkan permainan genre survival dengan gameplay yang seperti ini.					✓
7	Aplikasi dapat berjalan dengan lancar tanpa adanya lag dan/atau crash.					✓
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.					✓
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.				✓	
10	Saya merasa mudah melakukan serangan dasar.					✓
11	Saya merasa mudah menggambar gestur dan mengaktifkan Weapon Skill.				✓	
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.				✓	

C. KRITIK DAN SARAN

~~[illegible]~~ Game yang mudah cuman ada beberapa hal yang harus di sempurnakan, seperti sensitivitas.

KUISIONER TUGAS AKHIR

05111640000093 – *Muhammad Ivan Riansyah Putra*

PENERAPAN POINT-CLOUD GESTURE RECOGNIZER PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

Identitas Responden

Surabaya, 10 Juni 2020

Nama Lengkap : Jojo I. Mangkuta

Pekerjaan : Mahasiswa

Usia : 21

Jojo

A. KARAKTERISTIK RESPONDEN

Jawablah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

1. Pernahkah anda memainkan sebuah permainan digital atau *game* di Komputer?
   ☑ Pernah ○ Tidak Pernah
2. Pernahkah anda memainkan *game* dengan *genre Survival*?
   ☑ Pernah ○ Tidak Pernah
3. Pernahkah anda memainkan permainan *First Person Shooter*?
   ☑ Pernah ○ Tidak Pernah
4. Apakah anda memiliki fobia laba-laba atau Arachnophobia?
   ○ Punya ☑ Tidak Punya

B. PENILAIAN TERHADAP PERMAINAN

Isilah tabel dibawah ini dengan menggunakan tanda centang (v)

SS = Sangat Setuju *S = Setuju* *N = Netral*

TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter	STS	TS	N	S	SS
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.				✓	
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.					✓
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.					✓
4	Saya merasakan ketegangan seperti dikejar laba-laba raksasa.				✓	
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.				✓	
6	Saya merasa belum pernah memainkan permainan genre survival dengan gameplay yang seperti ini.				✓	
7	Aplikasi dapat berjalan dengan lancar tanpa adanya lag dan/atau crash.					✓
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.				✓	
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.			✓		
10	Saya merasa mudah melakukan serangan dasar.				✓	
11	Saya merasa mudah menggambar gestur dan mengaktifkan Weapon Skill.				✓	
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.					✓

C. KRITIK DAN SARAN

Bagus untuk sementara mouse-nya cepat tinggi, kalau digunakan, saya kira bisa dikurangi sedikit agar lebih mudah dimainkan.

KUISIONER TUGAS AKHIR

05111640000093 – *Muhammad Ivan Riansyah Putra*

PENERAPAN POINT-CLOUD GESTURE RECOGNIZER PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

Identitas Responden

Surabaya, 11 Juni 2020

Nama Lengkap : Yogi Fajriwani

Pekerjaan : Mahasiswa

Usia : 21

Yogi Fajriwani

A. KARAKTERISTIK RESPONDEN

Jawablah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

1. Pernahkah anda memainkan sebuah permainan digital atau *game* di Komputer?
   ☑ Pernah ○ Tidak Pernah
2. Pernahkah anda memainkan *game* dengan *genre Survival*?
   ☑ Pernah ○ Tidak Pernah
3. Pernahkah anda memainkan permainan *First Person Shooter*?
   ○ Pernah ☑ Tidak Pernah
4. Apakah anda memiliki fobia laba-laba atau Arachnophobia?
   ○ Punya ☑ Tidak Punya

B. PENILAIAN TERHADAP PERMAINAN

Isilah tabel dibawah ini dengan menggunakan tanda centang (v)

SS = Sangat Setuju *S = Setuju* *N = Netral*

TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter	STS	TS	N	S	SS
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.					✓
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.					✓
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.					✓
4	Saya merasakan ketegangan seperti dikejar laba-laba raksasa.					✓
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.					✓
6	Saya merasa belum pernah memainkan permainan genre survival dengan gameplay yang seperti ini.				✓	
7	Aplikasi dapat berjalan dengan lancar tanpa adanya lag dan/atau crash.					✓
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.					✓
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.				✓	
10	Saya merasa mudah melakukan serangan dasar.					✓
11	Saya merasa mudah menggambar gestur dan mengaktifkan Weapon Skill.					✓
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.					✓

C. KRITIK DAN SARAN

Pertanyaannya menarik dan tingkat kesulitan bisa ditingkatkan

KUISIONER TUGAS AKHIR

05111640000093 – *Muhammad Ivan Nizzayah Putra*

PENERAPAN POINT-CLOUD GESTURE RECOGNIZER PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

Identitas Responden

Nama Lengkap : Indri Tiffani

Pekerjaan : Pelajar

Usia : 15

Surabaya, 11 Juni 2020

Indri Tiffani

A. KARAKTERISTIK RESPONDEN

Jawablah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

1. Pernahkah anda memainkan sebuah permainan digital atau *game* di Komputer?
   ☑ Pernah ○ Tidak Pernah
2. Pernahkah anda memainkan *game* dengan *genre Survival*?
   ☑ Pernah ○ Tidak Pernah
3. Pernahkah anda memainkan permainan *First Person Shooter*?
   ○ Pernah ☑ Tidak Pernah
4. Apakah anda memiliki fobia laba-laba atau Arachnophobia?
   ☑ Punya ○ Tidak Punya

B. PENILAIAN TERHADAP PERMAINAN

Isilah tabel dibawah ini dengan menggunakan tanda centang (v)

SS = Sangat Setuju *S = Setuju* *N = Netral*

TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter	STS	TS	N	S	SS
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.				✓	
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.					✓
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.					✓
4	Saya merasakan ketegangan seperti dikejar laba-laba raksasa.				✓	
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.				✓	
6	Saya merasa belum pernah memainkan permainan genre survival dengan gameplay yang seperti ini.					✓
7	Aplikasi dapat berjalan dengan lancar tanpa adanya lag dan/atau crash.					✓
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.				✓	
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.					✓
10	Saya merasa mudah melakukan serangan dasar.					✓
11	Saya merasa mudah menggambar gestur dan mengaktifkan Weapon Skill.				✓	
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.					✓

C. KRITIK DAN SARAN

Tingkat kesulitan dapat disesuaikan. Jadi, bisa dipilih sesuai kemampuan

KUISIONER TUGAS AKHIR

05111640000093 – *Muhammad Ivan Nizzayah Putra*

PENERAPAN POINT-CLOUD GESTURE RECOGNIZER PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

Identitas Responden

Nama Lengkap : M. Nanssanya

Pekerjaan : Wiraswasta

Usia : 26

Surabaya, 11 Juni 2020

A. KARAKTERISTIK RESPONDEN

Jawablah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

1. Pernahkah anda memainkan sebuah permainan digital atau *game* di Komputer?
   ☑ Pernah ○ Tidak Pernah
2. Pernahkah anda memainkan *game* dengan *genre Survival*?
   ☑ Pernah ○ Tidak Pernah
3. Pernahkah anda memainkan permainan *First Person Shooter*?
   ☑ Pernah ○ Tidak Pernah
4. Apakah anda memiliki fobia laba-laba atau Arachnophobia?
   ○ Punya ☑ Tidak Punya

B. PENILAIAN TERHADAP PERMAINAN

Isilah tabel dibawah ini dengan menggunakan tanda centang (v)

SS = Sangat Setuju *S = Setuju* *N = Netral*

TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter	STS	TS	N	S	SS
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.				S	
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.				✓	
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.				✓	
4	Saya merasakan ketegangan seperti dikejar laba-laba raksasa.				✓	
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.					✓
6	Saya merasa belum pernah memainkan permainan genre survival dengan gameplay yang seperti ini.					✓
7	Aplikasi dapat berjalan dengan lancar tanpa adanya lag dan/atau crash.				✓	
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.					✓
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.				✓	
10	Saya merasa mudah melakukan serangan dasar.					✓
11	Saya merasa mudah menggambar gestur dan mengaktifkan Weapon Skill.					✓
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.					✓

C. KRITIK DAN SARAN

KUALITAS GAMBAR KURANG HALUS

KUISIONER TUGAS AKHIR

05111640000093 – *Muhammad Ivan Rosiyah Putra*

PENERAPAN POINT-CLOUD GESTURE RECOGNIZER PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

Identitas Responden Surabaya, 11 Juni 2020

Nama Lengkap : Pri Kurniawaty

Pekerjaan : Guru

Usia : 28

A. KARAKTERISTIK RESPONDEN

Jawablah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

1. Pernahkah anda memainkan sebuah permainan digital atau *game* di Komputer?
   ☑ Pernah ○ Tidak Pernah
2. Pernahkah anda memainkan *game* dengan *genre Survival*?
   ☑ Pernah ○ Tidak Pernah
3. Pernahkah anda memainkan permainan *First Person Shooter*?
   ☑ Pernah ○ Tidak Pernah
4. Apakah anda memiliki fobia laba-laba atau Arachnophobia?
   ○ Punya ☑ Tidak Punya

B. PENILAIAN TERHADAP PERMAIAN

Isilah tabel dibawah ini dengan menggunakan tanda centang (v)

SS = Sangat Setuju *S = Setuju* *N = Netral*

TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter	STS	TS	N	S	SS
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.					✓
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.					✓
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.				✓	
4	Saya merasakan ketegangan seperti dikejar laba-laba raksasa.				✓	
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.				✓	
6	Saya merasa belum pernah memainkan permainan genre survival dengan gameplay yang seperti ini.					✓
7	Aplikasi dapat berjalan dengan lancar tanpa adanya lag dan/atau crash.					✓
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.					✓
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.				✓	
10	Saya merasa mudah melakukan serangan dasar.				✓	
11	Saya merasa mudah menggambar gestur dan mengaktifkan Weapon Skill.				✓	
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.					✓

C. KRITIK DAN SARAN

Kendali sangat mengakselkan gesture sedikit membingungkan jika tidak di jelaskan dahulu.

KUISIONER TUGAS AKHIR

05111640000093 – *Muhammad Ivan Rosiyah Putra*

PENERAPAN POINT-CLOUD GESTURE RECOGNIZER PADA PERMAIAN NIGHTMARE: ARACHNOPHOBIA

Identitas Responden Surabaya, 11 Juni 2020

Nama Lengkap : muhammad rahman a.

Pekerjaan : mahasiswa

Usia : 21

A. KARAKTERISTIK RESPONDEN

Jawablah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

1. Pernahkah anda memainkan sebuah permainan digital atau *game* di Komputer?
   ☑ Pernah ○ Tidak Pernah
2. Pernahkah anda memainkan *game* dengan *genre Survival*?
   ☑ Pernah ○ Tidak Pernah
3. Pernahkah anda memainkan permainan *First Person Shooter*?
   ☑ Pernah ○ Tidak Pernah
4. Apakah anda memiliki fobia laba-laba atau Arachnophobia?
   ○ Punya ☑ Tidak Punya

B. PENILAIAN TERHADAP PERMAIAN

Isilah tabel dibawah ini dengan menggunakan tanda centang (v)

SS = Sangat Setuju *S = Setuju* *N = Netral*

TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter	STS	TS	N	S	SS
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.				✓	
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.					✓
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.					✓
4	Saya merasakan ketegangan seperti dikejar laba-laba raksasa.					✓
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.				✓	
6	Saya merasa belum pernah memainkan permainan genre survival dengan gameplay yang seperti ini.					✓
7	Aplikasi dapat berjalan dengan lancar tanpa adanya lag dan/atau crash.				✓	
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.					✓
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.				✓	
10	Saya merasa mudah melakukan serangan dasar.				✓	
11	Saya merasa mudah menggambar gestur dan mengaktifkan Weapon Skill.			✓		
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.			✓		

C. KRITIK DAN SARAN

karena menggunakan mouse. Rembuatan gesture sedikit kaku dan tidak luwes.

KUISIONER TUGAS AKHIR

05111640000093 – *Muhammad Ivan Riansyah Putra*

PENERAPAN POINT-CLOUD GESTURE RECOGNIZER PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

Identitas Responden Surabaya, 10 Juni 2020

Nama Lengkap : Arif Sudibyo

Pekerjaan : Mahasiswa

Usia : 21 tahun

Arif Arif Sudibyo

A. KARAKTERISTIK RESPONDEN

Jawablah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

1. Pernahkah anda memainkan sebuah permainan digital atau *game* di Komputer?
   ☑ Pernah ○ Tidak Pernah
2. Pernahkah anda memainkan *game* dengan *genre Survival*?
   ☑ Pernah ○ Tidak Pernah
3. Pernahkah anda memainkan permainan *First Person Shooter*?
   ☑ Pernah ○ Tidak Pernah
4. Apakah anda memiliki fobia laba-laba atau Arachnophobia?
   ☑ Punya ○ Tidak Punya

B. PENILAIAN TERHADAP PERMAINAN

Isilah tabel dibawah ini dengan menggunakan tanda centang (v)

SS = Sangat Setuju *S = Setuju* *N = Netral*

TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter	STS	TS	N	S	SS
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.					✓
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.				✓	
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.				✓	
4	Saya merasakan kelegangan seperti dikejar laba-laba raksasa.					✓
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.				✓	
6	Saya merasa belum pernah memainkan permainan genre survival dengan gameplay yang seperti ini.					✓
7	Aplikasi dapat berjalan dengan lancar tanpa adanya lag dan/atau crash.					✓
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.					✓
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.				✓	
10	Saya merasa mudah melakukan serangan dasar.					✓
11	Saya merasa mudah menggambar gestur dan mengaktifkan Weapon Skill.			✓		
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.				✓	

C. KRITIK DAN SARAN

- Pada stage selanjutnya terdiri lebih sebaiknya ditingkatkan.

KUISIONER TUGAS AKHIR

05111640000093 – *Muhammad Ivan Riansyah Putra*

PENERAPAN POINT-CLOUD GESTURE RECOGNIZER PADA PERMAINAN NIGHTMARE: ARACHNOPHOBIA

Identitas Responden Surabaya, 11 Juni 2020

Nama Lengkap : Nada Nibras Salsabila Rosadi

Pekerjaan : Mahasiswi

Usia : 22

A. KARAKTERISTIK RESPONDEN

Jawablah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

1. Pernahkah anda memainkan sebuah permainan digital atau *game* di Komputer?
   ☑ Pernah ○ Tidak Pernah
2. Pernahkah anda memainkan *game* dengan *genre Survival*?
   ☑ Pernah ○ Tidak Pernah
3. Pernahkah anda memainkan permainan *First Person Shooter*?
   ☑ Pernah ○ Tidak Pernah
4. Apakah anda memiliki fobia laba-laba atau Arachnophobia?
   ○ Punya ☑ Tidak Punya

B. PENILAIAN TERHADAP PERMAINAN

Isilah tabel dibawah ini dengan menggunakan tanda centang (v)

SS = Sangat Setuju *S = Setuju* *N = Netral*

TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter	STS	TS	N	S	SS
1	Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik.				✓	
2	Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali.				✓	
3	Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk.					✓
4	Saya merasakan kelegangan seperti dikejar laba-laba raksasa.				✓	
5	Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini.				✓	
6	Saya merasa belum pernah memainkan permainan genre survival dengan gameplay yang seperti ini.					✓
7	Aplikasi dapat berjalan dengan lancar tanpa adanya lag dan/atau crash.				✓	
8	Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan.					✓
9	Saya merasa mudah mengontrol gerakan pemain dan arah kamera.				✓	
10	Saya merasa mudah melakukan serangan dasar.				✓	
11	Saya merasa mudah menggambar gestur dan mengaktifkan Weapon Skill.					✓
12	Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya.				✓	

C. KRITIK DAN SARAN

- Stage kosong lebih-lebihnya tidak karena rumputnya terlalu tinggi

Nama Lengkap

20 tanggapan

Bernardino Adam Wijaya

Aisyah Muswar

Fariz Ardin Adhiyaksa

Mohammad Nafis Naufally

Muhammad Fadhlan Min Robby

Falah Nurli Filano

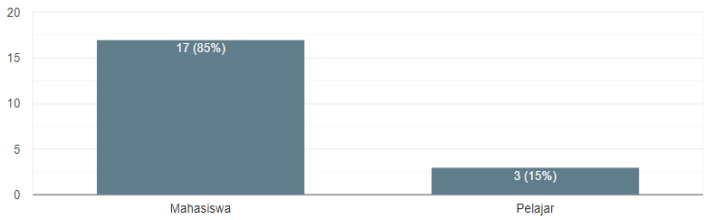
Haidar Sakti Oktafiansyah

Muhammad Farhan Agus Trisasti

Oktavian Rahman Koko

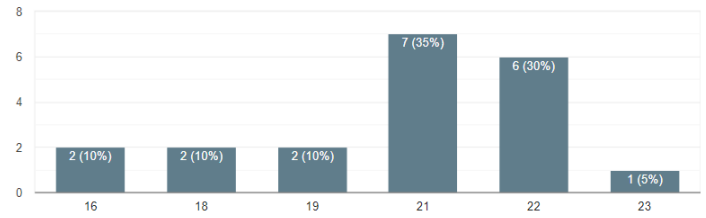
Pekerjaan

20 tanggapan



Usia

20 tanggapan



Spesifikasi Perangkat Yang Digunakan Untuk Uji Coba (CPU, GPU, dan RAM)

20 tanggapan

AMD FX-9830P R7, Radeon Rx460, 16 GB

Intel Pentium 4417U 2.3 GHz, Intel HD Graphic 610, Ram 4 GB

Core i7-4720HQ, NVIDIA GeForce GT 940M, Ram 4GB

CPU Intel Core i7 6700HQ, GPU GTX 950M, RAM 8GB

Processor AMD Ryzen 3550H, GTX 1650, 8GB RAM

Intel I7-7660U, Intel Iris Plus Graphics 640, 16gb

i5-7200U, GEFORCE 920MX, 8GB

core i7-9750H, RTX2060 6GB, 16GB RAM ddr4 2666Mhz

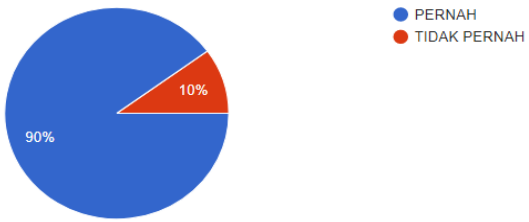
AMD FX-9830P R7, RX 460, 16 GB

Pernahkah anda memainkan sebuah permainan digital atau game di Komputer?

20 tanggapan

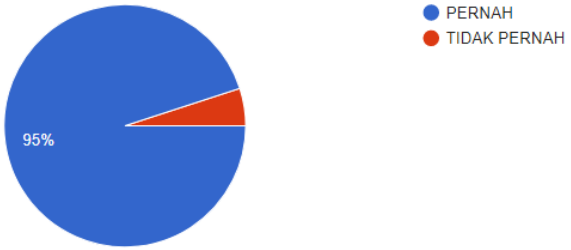
Pernahkah anda memainkan game dengan genre Survival?

20 tanggapan



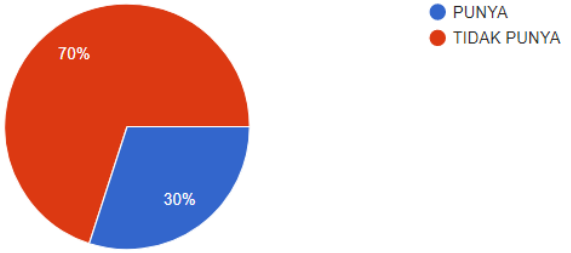
Pernahkah anda memainkan permainan First Person Shooter?

20 tanggapan



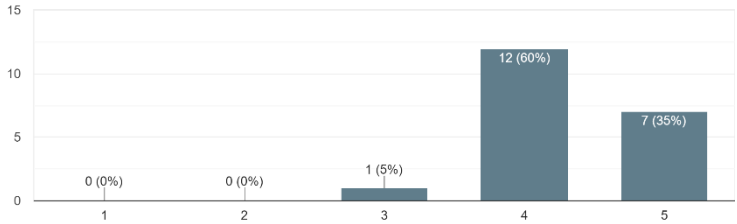
Apakah anda memiliki fobia laba-laba atau Arachnophobia?

20 tanggapan



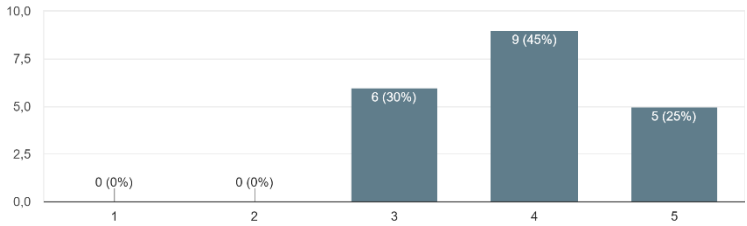
Permainan Nightmare: Arachnophobia memiliki tampilan, warna, dan desain antarmuka yang menarik

20 tanggapan



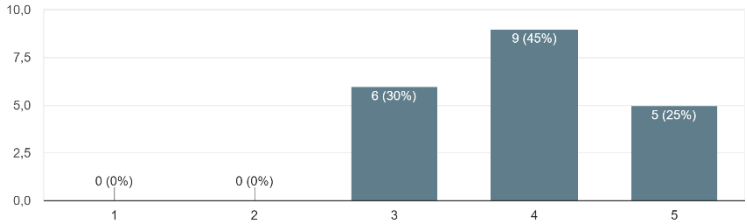
Permainan Nightmare: Arachnophobia memiliki tata letak tombol, instruksi, dan informasi lainnya yang mudah dilihat / dikenali

20 tanggapan



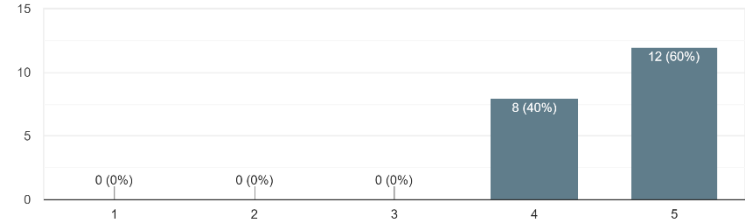
Saya merasakan sensasi layaknya sedang berada dalam mimpi buruk

20 tanggapan



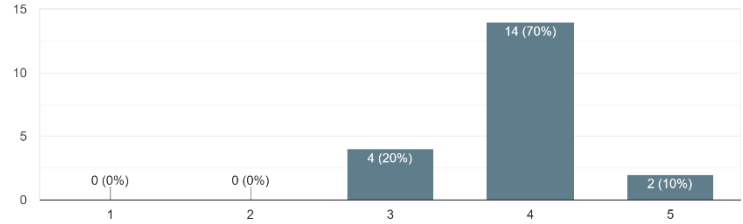
Saya merasakan ketegangan seperti dikejar laba-laba raksasa

20 tanggapan



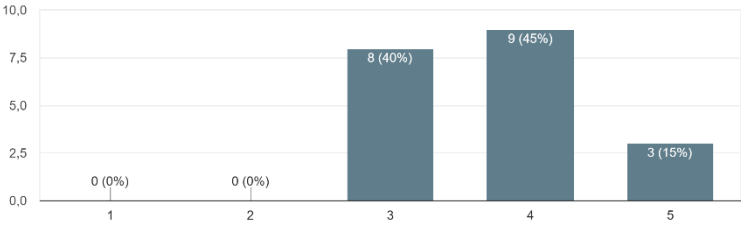
Permainan ini dapat melatih kemampuan berpikir saya dalam mengambil keputusan dalam permainan ini

20 tanggapan



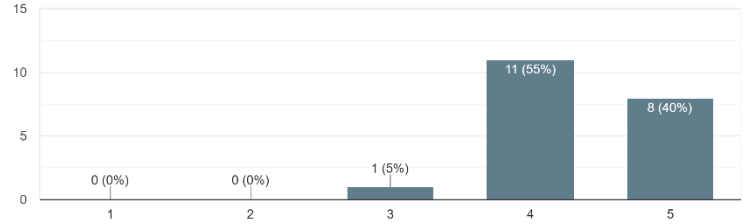
Saya merasa belum pernah memainkan permainan genre survival dengan gameplay yang seperti ini

20 tanggapan

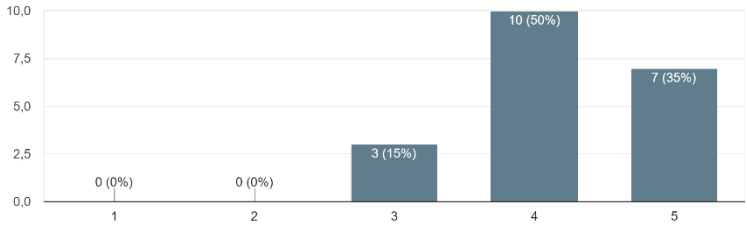


Aplikasi dapat berjalan dengan lancar tanpa adanya lag dan/atau crash

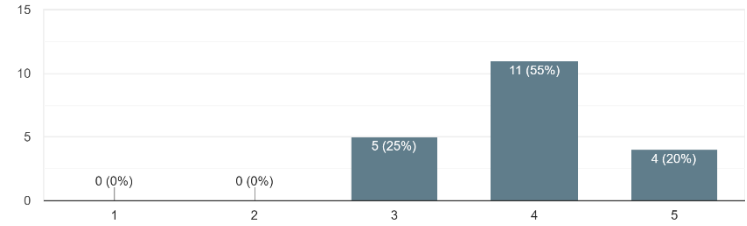
20 tanggapan



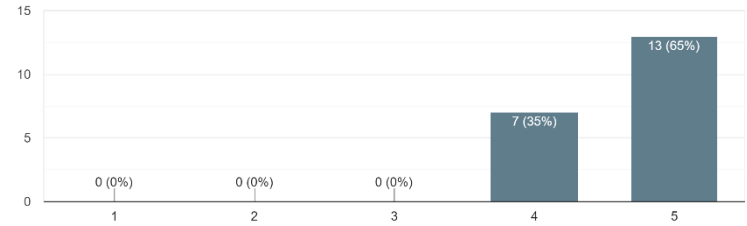
Saya merasa terbantu dengan adanya petunjuk yang disediakan permainan
20 tanggapan



Saya merasa mudah mengontrol gerakan pemain dan arah kamera
20 tanggapan

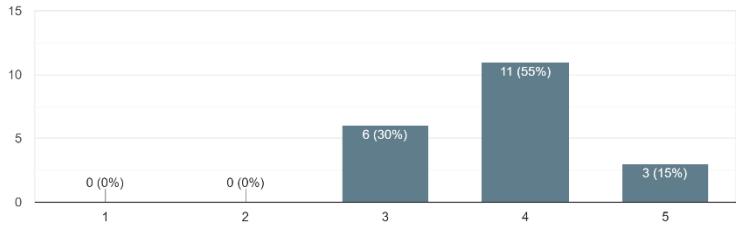


Saya merasa mudah melakukan serangan dasar
20 tanggapan



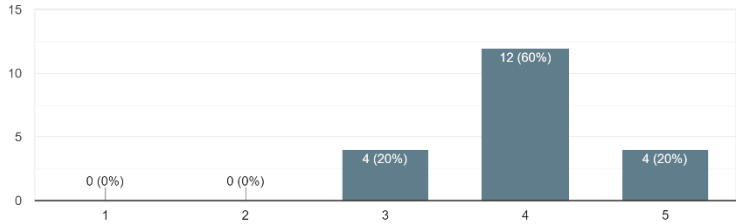
Saya merasa mudah menggambar gestur dan mengaktifkan Weapon Skill

20 tanggapan



Saya merasa tertarik untuk memainkan permainan ini untuk selanjutnya

20 tanggapan



Kritik dan Saran

20 tanggapan

Menggambar bentuk nya susah jika sambil dikejar laba-laba. Tapi seru gamenya.

Ada beberapa mekanisme yang sulit dipahami jika tidak dijelaskan secara langsung

Ditambah countdown bar setelah menggambar pola sebagai batas konfirmasi pattern. jika melebihi batas waktu, maka pola yang telah digambar sudah tidak valid lagi.

Saran untuk laba-labanya ditambahkan dalam satu waktu sekaligus agar menambah tingkat kesulitan.

Ada stage yang laba-labanya terlalu lama spawn. sebaiknya dipercepat agar pemain tidak kebingungan harus ngapain

Laba-laba ada yang nyangkut di flat car pada stage railway. Mungkin bisa diatur lagi collider flat car nya agar terdeteksi dengan baik oleh AI nya.

Cerita pada game nya terasa kurang menarik, seperti hanya pelengkap saja. Mungkin cerita akan lebih menari jika ada cutscene

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Muhammad Ivan Riansyah Putra, lahir di Denpasar pada tanggal 15 Oktober 1998. Penulis menempuh pendidikan mulai dari TK Sandhy Putra, SDN 1 Sumbawa Besar, SMPN 1 Sumbawa Besar, SMAN 1 Bantul, dan saat ini melanjutkan studinya di Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember di Surabaya.

Selama menempuh dunia perkuliahan, penulis aktif mengikuti organisasi dan kepanitiaan di kampus antara lain sebagai anggota staff Departemen Teknologi HMTC Kabinet Kreasi, staff ahli Information Media BEMF ASASI, staff 3D Schematics 2017, staff ahli 3D Schematics 2018, Staff publikasi dan dokumentasi pada K-Fest 2017 & 2018, Staff publikasi dan dokumentasi pada Inochi 2017 & 2018.

Dalam menyelesaikan pendidikan sarjananya, penulis mengambil bidang minat Interaksi, Grafika, dan Seni (IGS). Komunikasi pada penulis dapat dihubungi melalui alamat *e-mail*: shakell.zero@gmail.com