TUGAS AKHIR – IF184802

RANCANG BANGUN APLIKASI BERBASIS REALITAS VIRTUAL PADA SIMULASI *WALL CLIMBING* MENGGUNAKAN OCULUS RIFT, OCULUS TOUCH DAN LEAP MOTION

GD WAHYU NUGRAHA SUBAGIA
NRP 05111510000016

Dosen Pembimbing
Dr. Eng. Darlis Herumurti, S.Kom., M.Kom.
Anny Yuniarti, S.Kom, M.Comp.Sc

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – IF184802

RANCANG BANGUN APLIKASI BERBASIS REALITAS VIRTUAL PADA SIMULASI *WALL CLIMBING* MENGGUNAKAN OCULUS RIFT, OCULUS TOUCH DAN LEAP MOTION

GD WAHYU NUGRAHA SUBAGIA
NRP 05111540000016

Dosen Pembimbing
Dr. Eng. Darlis Herumurti, S.Kom., M.Kom.
Anny Yuniarti, S.Kom, M.Comp.Sc

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – IF184802

RANCANG BANGUN APLIKASI BERBASIS REALITAS VIRTUAL PADA SIMULASI *WALL CLIMBING* MENGGUNAKAN OCULUS RIFT, OCULUS TOUCH DAN LEAP MOTION

GD WAHYU NUGRAHA SUBAGIA
NRP 05111540000016

Advisor
Dr. Eng. Darlis Herumurti, S.Kom., M.Kom.
Anny Yuniarti, S.Kom, M.Comp.Sc

INFORMATICS DEPARTMENT
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI BERBASIS REALITAS VIRTUAL PADA SIMULASI *WALL CLIMBING* MENGGUNAKAN OCULUS RIFT, OCULUS TOUCH DAN LEAP MOTION

TUGAS AKHIR

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

Oleh:
GD WAHYU NUGRAHA SUBAGIA
NRP. 05111540000016

Disetujui oleh Dosen Pembimbing Tugas Akhir:
Dr.Eng. Darlis Herumurti, S.Kom., M.Kom. .....
NIP: 19771217 200312 1 001 (pembimbing 1)

Anny Yuniarti, S.Kom, M.Comp.Sc .....
NIP: 19810622 200501 2 002 (pembimbing 2)

SURABAYA
JUNI, 2019

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN APLIKASI BERBASIS REALITAS VIRTUAL PADA SIMULASI *WALL CLIMBING* MENGGUNAKAN OCULUS RIFT, OCULUS TOUCH DAN LEAP MOTION

Nama Mahasiswa	: GD Wahyu Nugraha Subagia
NRP	: 05111540000016
Departemen	: Informatika FTIK-ITS
Dosen Pembimbing 1	: Dr. Eng. Darlis Herumurti, S.Kom., M.Kom.
Dosen Pembimbing 2	: Anny Yuniarti, S.Kom, M.Comp.Sc

ABSTRAK

Simulasi panjat dinding atau yang lebih dikenal dengan Wall Climbing merupakan kegiatan yang memiliki nilai-nilai simulasi sekaligus memiliki daya tarik tersendiri, seperti kesulitan yang beraneka ragam yang terdapat pada lekukan-lekukan yang dibuat sesuai dengan keinginan serta tingkat kesulitannya yang berbeda-beda. Simulasi panjat dinding telah berkembang pesat, terbukti dengan adanya banyak agenda kegiatan ekspedisi panjat dinding maupun kompetisi panjat dinding buatan. Simulasi Wall Climbing ini akan dibuat menjadi sebuah aplikasi yang berbasis realitas virtual yang menggunakan alat Oculus Rift dengan Oculus Touch dan Oculus Rift dengan Leap Motion Controller untuk mendeteksi gerakan tangan dan jari.

Secara umum proses ini dimulai dengan pemain membuka aplikasi dan akan menampilkan pilihan tombol play untuk memulai kegiatan memanjat dan pilihan tombol exit untuk keluar dari aplikasi. Pemain akan menuju ke halaman utama kegiatan yang akan terdapat objek wall climbing yang terdapat poin-poin untuk memanjat dengan cara menggengam dan menariknya kebawah

maka pemain akan berpindah posisi kebagian atas sedikit demi sedikit hingga berhasil mencapai atas tembok. Pada teknologi yang pertama yaitu Oculus Rift dengan menggunakan Oculus Touch sebagai controllernya pemain akan menggunakan Oculus Headset untuk menampilkan display dari aplikasi dan memegang kedua controller dari Oculus untuk menjadi sensor dengan cara menekan trigger dari controller dan akan menjadi tangan pemain pada saat memainkan aplikasi ini. Dengan teknologi yang kedua yaitu Oculus Rift dengan Leap Motion Controller, mula-mula pemain akan menggunakan Oculus Rift Headset untuk menjadi display dari pemain dan menggunakan Leap Motion Controller yang dipasang didepan Oculus Rift Headset untuk menjadi sensor dari tangan pada saat menggenggam dan tidak menggenggam sekaligus untuk menjadi kamera pengganti dari Oculus Rift Headset.

Hasil dari pengujian Tugas Akhir ini adalah sebuah aplikasi yang dapat dijalankan di perangkat computer high-end. Aplikasi ini dibangun dengan menggunakan bahasa pemrograman C# di Unity 3D versi 2018.3.5f1 dengan pengujian kepada responden, dengan pengujian beta dapat disimpulkan aplikasi telah mengimplementasikan perancangan perangkat yang baik.

Kata kunci: Leap Motion, Oculus Rift, Oculus Touch, Realitas Virtual, Unity 3D, Wall Climbing

VIRTUAL REALITY BASED APPLICATIONS DESIGN ON WALL CLIMBING SIMULATION USING THE OCULUS RIFT, OCULUS TOUCH AND LEAP MOTION

Name	: GD Wahyu Nugraha Subagia
NRP	: 05111540000016
Department	: Informatika FTIK-ITS
Supervisor 1	: Dr. Eng. Darlis Herumurti, S.Kom., M.Kom.
Supervisor 2	: Anny Yuniarti, S.Kom, M.Comp.Sc

ABSTRACT

Wall Climbing is an activity that has sports values while at the same time has its own charm, such as various difficulties which are found in indentations that are made according to their desires and different levels of difficulty. Wall climbing sports have grown rapidly, as evidenced by the many agendas of wall climbing expeditions and artificial wall climbing competitions. This wall climbing sport will be made into a virtual reality-based application that uses the Oculus Rift with Oculus Touch and Oculus Rift with Leap Motion Controller tools to detect hand and finger movements.

In general, this process begins with the player open the application and will display the play button option to start climbing activities and the exit button option to exit the application. The player will go to the main page of the activity where there will be a wall climbing object that has points to climb by grasping and pulling it down so the player will move to the top position little by little until he reaches the top of the wall. In the first technology, Oculus Rift with Oculus Touch by using its controller, the player will use the Oculus Headset to show the display from the

application and hold both controllers from Oculus to become sensors by pressing trigger from the controller and will be the player's hand when playing this application. With the second technology,Ouclus Rift with the Leap Motion Controller, first of all, player will use the Oculus Rift Headset to be a display of the player and use the Leap Motion Controller installed in front of the Oculus Rift Headset to become a sensor from the hand when holding and not holding at the same time to become a replacement camera for the Oculus Rift Headset.

The results of this Final Assignment are applications that can be run on high-end computer devices. This application was built using C# programming language in Unity 3D version 2018.3.5f1 with testing to respondents, with beta testing can be concluded that the application has implemented the design well.

Keywords: Leap Motion, Oculus Rift, Oculus Touch, Virtual Reality, Unity 3D, Wall Climbing

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Ida Sang Hyang Widhi Wasa, Tuhan Yang Maha Esa atas limpahan berkat dan rahmat-Nya sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul:

RANCANG BANGUN APLIKASI BERBASIS REALITAS VIRTUAL PADA SIMULASI *WALL CLIMBING* MENGGUNAKAN OCULUS RIFT, OCULUS TOUCH DAN LEAP MOTION

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 Dr. Eng. Darlis Herumurti S.Kom., M.Kom. dan Ibu Anny Yuniarti, S.Kom., M.Comp.Sc. selaku dosen pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
3. Teman sekaligus penasihat selama pembuatan tugas akhir ini Surya Darma dan Dias Adhi yang selalu sabar dan tidak pernah letih untuk membantu penulis setiap penulis menemukan kesusahan.
4. Sahabat satu kontrakan selama 4 tahun : Krisnadi, Ary, Tedja, Gilang yang selalu memberikan dukungan dan motivasi kepada penulis ketika rasa putus asa dan ingin menyerah mendominasi hari-hari penulis di penghujung masa pembuatan tugas akhir ini.
5. Sahabat dekat Gerald, Dio, Glenn, Ivansat, Subhan yang selalu memberikan dukungan dan motivasi kepada penulis ketika rasa putus asa dan ingin menyerah mendominasi

hari-hari penulis di penghujung masa pembuatan tugas akhir ini.

6. Ida Ayu Kade Rizki yang selalu memberi dukungan dan semangat kepada penulis selama menyelesaikan tugas akhir ini.
7. Sahabat Gaes yang selalu memberikan dukungan dan motivasi kepada penulis ketika rasa putus asa dan ingin menyerah mendominasi hari-hari penulis di penghujung masa pembuatan tugas akhir ini.
8. Administrator Lab IGS yang setia berada di laboratorium IGS, membantu penulis selama perkuliahan dan pengerjaan tugas akhir ini.
9. Segenap teman-teman yang bersamaan mengambil TA IGS Subhan, Yuga, Narendra, Adi, Tian, Dias yang telah menemani selama pembuatan Tugas Akhir di lab IGS.
10. Teman-teman C1F yang senantiasa menepati janjinya untuk saling menguatkan ketika satu jatuh dan lemah, dan merekatkan ketika satu hilang dan pecah.
11. Teman-teman TPKH ITS khususnya Ekalawya 2015 atas suka duka yang dijalani bersama selama dirantauan.

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

Surabaya, Juni 2019

GD Wahyu Nugraha Subagia

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Simulasi panjat dinding atau yang lebih dikenal dengan *Wall Climbing* merupakan kegiatan yang memiliki nilai-nilai simulasi sekaligus memiliki daya tarik tersendiri, seperti kesulitan yang beraneka ragam yang terdapat pada lekukan-lekukan yang dibuat sesuai dengan keinginan serta tingkat kesulitannya yang berbeda-beda. Federasi Panjat Tebing Indonesia (FPTI) didirikan 21 April 1988 di Indonesia sendiri simulasi panjat tebing telah cukup memasyarakat dan berkembang pesat. Hal ini terbukti dengan adanya banyak agenda kegiatan ekspedisi panjat tebing maupun kompetisi panjat tebing buatan yang dilakukan oleh organisasi pecinta alam atau perkumpulan pemanjat baik tingkat daerah maupun nasional. Dan menjadi salah satu cabang simulasi yang dipertandingan di tingkat Asian Games.

Dalam perkembangan teknologi dari waktu ke waktu mengalami kemajuan yang sangat pesat. Seiring dengan perkembangan itu pula, teknologi yang digunakan untuk mengembangkan sebuah aplikasi semakin banyak. Salah satu contohnya adalah teknologi yang sedang berkembang dan menarik perhatian yaitu aplikasi berbasis realitas virtual. Realitas virtual merupakan teknologi yang dapat membuat pengguna berinteraksi dengan suatu lingkungan yang disimulasikan oleh komputer, suatu lingkungan sebenarnya yang ditiru atau benar-benar lingkungan yang hanya ada di imajinasi saja.

Oculus Rift adalah salah satu contoh alat yang sering digunakan untuk membuat sebuah aplikasi berbasis realitas virtual yang diciptakan oleh Oculus VR, sebuah divisi dari Facebook Inc, Oculus Rift termasuk perangkat realitas virtual yang terbilang sangat canggih, karena pada perangkat Oculus Rift tidak

membutuhkan *smartphone* untuk menjalankannya, Oculus Rift memiliki perangkat lain untuk memainkannya seperti terdapat *remote* atau biasa disebut dengan istilah *controller* untuk menambahkan sensasi yang berbeda layaknya bermain dengan dunia nyata, lalu juga terdapat kamera untuk mendeteksi gerakan saat bermain, dan terakhir memiliki *stick controller Xbox* yang kegunaannya seperti *remote controller* tetapi untuk *stick controller* ini memiliki lebih banyak tombol sehingga saat memakai perangkat ini pemain hanya sekedar menggenggam alatnya lalu bermain layaknya memainkan *playstation.* Alat yang kedua adalah Leap Motion Controller, alat ini adalah alat tambahan yang dapat dihubungkan ke komputer kemudian dapat digunakan layaknya menggantikan mouse maupun keyboard karena Leap Motion Controller dapat membantu penggunanya mengendalikan komputer hanya dengan gerakan tangan maupun jari diudara. Bentuk Leap Motion Controller berukuran kotak yang cukup kecil sehingga mudah dibawa kemanapun. Cara kerja Leap Motion Controller ini adalah pengguna cukup meletakan perangkat ini didekat komputer dan kemudian perangkat akan mendeteksi keberadaan tangan maupun jari yang selanjutnya pengguna dapat menggunakan gerakan tangan atau jari yang diinginkan serta gerakan kombinasi yang sudah diatur.

Dalam hal ini penulis ingin membuah sebuah aplikasi Simulasi *Wall Climbing* yang berbasis realitas virtual yang menggunakan alat Oculus Rift dan Leap Motion Controller untuk mendeteksi gerakan tangan dan jari. Simulasi *Wall Climbing* ini dirancang dengan realitas virtual difungsikan untuk memberikan sensasi kepada para pendaki pemula yang ingin merasakan sensai terlebih dahulu bagaimana merasakan saat melakukan pemanjatan atau bahkan untuk pemain yang ingin mengetahui bagaimana rasanya memanjat tebing tanpa melakukan di dunia nyata tapi bisa sangat real dengan aplikasi ini. Pemain harus menggunakan Oculus Rift sebagai alat realitas virtual dan akan dipasangkan alat Leap Motion Controller pada bagian kacamata untuk dapat mendeteksi

gerakan tangan dan jari saat melakukan pemanjatan. Pemain harus memanjat dari bawah dinding hingga keatas dengan memegang poin-poin yang disedakan tanpa terlepas dan terjatuh.

1.2. Rumusan Permasalahan

Berikut rumusan masalah yang diangkat dalam tugas akhir ini adalah:

1. Bagaimana rancangan visual simulasi *Wall Climbing*?
2. Bagaimana melakukan deteksi sensor dari Oculus Touch dan Leap Motion?
3. Bagaimana implementasi simulasi *Wall Climbing* pada kombinasi Oculus Rift dengan Oculus Touch dan Oculus Rift dengan Leap Motion?
4. Bagaimana implementasi dari rancangan di atas diselesaikan dengan *Game Engine* Unity 3D?
5. Bagaimana melakukan pengujian terhadap simulasi *Wall Climbing*?

1.3. Batasan Permasalahan

Berikut batasan masalah pada tugas akhir ini adalah:

1. Aplikasi yang dibuat merupakan aplikasi realitas virtual yang membutuhkan Oculus Rift, Oculus Touch dan Leap Motion Controller.
2. Aplikasi yang dibuat hanya bisa digunakan oleh satu pemain.
3. Lingkungan pengembangan yang digunakan menggunakan aplikasi Unity 3D.
4. Bahasa pemrograman yang digunakan mengggunakan Bahasa C#.

1.4. Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk mengimplementasikan simulasi *Wall Climbing* kedalam dunia digital dengan menggunakan dua metode yaitu Oculus Rift sebagai alat realitas virtual beserta Controller Oculus Rift untuk mendeteksi tangan dan Leap Motion Controller sebagai sensor untuk mengetahui gerak dari tangan dan jari.

1.5. Manfaat

Manfaat dari hasil tugas akhir ini.

1. Memberikan sensasi simulasi yang berbeda berbasis realitas virtual dengan menggunakan alat Oculus Rift dan Leap Motion.
2. Memberikan pengalaman untuk user pemula yang ingin belajar dan mengetahui bagaimana simulasi *Wall Climbing* dengan teknologi realitas virtual.
3. Mengetahui kekurangan dan kelebihan Oculus Rift dan Leap Motion

1.6. Metodologi

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

1.6.1. Studi literatur

Pada studi literatur ini, akan dipelajari sejumlah referensi yang relevan terhadap tugas akhir yang akan dikerjakan, seperti mempelajari cara kerja atau aturan bermain dalam simulasi *Wall Climbing* yang akan diimplementasikan untuk tugas akhir ini, mempelajari pembuatan aplikasi realitas virtual menggunakan Oculust Rift, mempelajari mengenai teknologi Leap Motion,

mempelajari Bahasa C# untuk penggunakan aplikasi Unity 3D, dan bagaimanan mengintegrasikan semuanya. Studi literatur ini didapatkan dari buku, internet serta materi-materi kuliah yang berhubungan dengan sistem yang dibangun.

1.6.2. Analisis dan desain sistem

Pada tahap ini dilakukan analisis dan pendefinisian kebutuhan sistem untuk masalah yang dihadapi, terutama analisis terkait skenario dalam simulasi *Wall Climbing* yang akan diterapkan pada sistem. Selanjutnya, dilakukan perancangan sistem dengan beberapa tahap sebagai berikut:

1. Mempelajari dokumentasi dan tutorial Unity 3D.
2. Mempelajari dokumentasi dan tutorial Oculus Rift.
3. Mempelajari dokumentasi dan tutorial Leap Motion Controller.
4. Perancangan *gameplay* realitas virtual dari simulasi *Wall Climbing.*

1.6.3. Implementasi sistem

Pada tahap ini akan dilakukan pembangunan sistem. Sistem yang dimaksud disini, yaitu aplikasi realitas virtual yang dibangun dengan menggunakan *tools* Unity 3D, dengan alat dan teknologi dari Oculus Rift dan Leap Motion Controller.

1.6.4. 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 sesuai dengan yang diharapkan . Pengujian akan dilakukan terhadap 10 mahasiswa Departemen Informatika, mereka akan menjadi penguji dan memainkan game Simulasi *Wall Climbing.* Pengujian dilakukan untuk memeriksa manakah alat yang lebih baik untuk penggunaannya pada aplikasi *Wall Climbing*. Selain itu juga untuk mengetahui kelebihan dan kekurangan pada aplikasi dan juga mengukur tingkat kepuasan pemain dalam memainkan aplikasi tersebut.

1.6.5. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan proses dokumentasi dan pembuatan laporan dari seluruh konsep, tinjauan pustaka, metode, implementasi, proses yang telah dilakukan, pengujian, evaluasi dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir.

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 Desain dan Perancangan Sistem

Bab ini membahas mengenai desain dan perancangan sistem yang akan dibangun.

Bab IV Implementasi Sistem

Bab ini membahas mengenai bagaimana implementasi sistem dari 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.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

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

2.1. Realitas Virtual

Realitas virtual adalah sebuah teknologi yang membuat pengguna dapat berinteraksi dengan lingkungan yang ada dalam dunia maya yang disimulasikan oleh komputer, sehingga pengguna merasa berada di dalam lingkungan tersebut. Teknologi realitas virtual sejatinya telah banyak diterapkan di beberapa sektor industri seperti kedokteran, penerbangan, pendidikan, arsitek, militer, hiburan, dan lain sebagainya. Realitas virtual sangat membantu dalam menyimulasikan sesuatu yang sulit untuk dihadirkan secara langsung dalam dunia nyata. Tentunya ini dapat membuat lebih praktis dan lebih ekonomis.

Kelebihan utama dari realitas virtual adalah pengalaman yang membuat pengguna merasakan sensasi dunia nyata dalam dunia virtual/maya. Bahkan perkembangan teknologi realitas virtual saat ini memungkinkan tidak hanya indra penglihatan dan pendengaran saja yang bisa merasakan sensasi nyata dari dunia maya dari realitas virtual, namun juga indra yang lainnya.

Untuk memunculkan sensasi nyata dari realitas virtual diperlukan perangkat pendukung. Paling tidak dibutuhkan sebuah *head mounted display* (HMD), yaitu sebuah perangkat pendukung yang dipasangkan *smartphone* yang mendukung realitas virtual untuk bisa merasakan sensasi realitas virtual. Terdapat 4 elemen penting dalam realitas virtual. Adapun 4 elemen itu adalah sebagai berikut:

a. *Virtual world*, sebuah konten yang menciptakan dunia virtual dalam bentuk *screenplay* maupun *script.*
b. *Immersion*, sebuah sensasi yang membawa pengguna teknologi realitas virtual merasakan ada di sebuah lingkungan nyata yang padahal fiktif. *Immersion* dibagi dalam 3 jenis, yakni:
   - *Mental immersion*, membuat mental penggunanya merasa seperti berada di dalam lingkungan nyata.
   - *Physical immersion*, membuat fisik penggunanya merasakan suasana di sekitar lingkungan yang diciptakan oleh realitas virtual tersebut.
   - *Mentally immersed*, memberikan sensasi kepada penggunanya untuk larut dalam lingkungan yang dihasilkan realitas virtual.
c. *Sensory feedback*, berfungsi untuk menyampaikan informasi dari *virtual world* ke indera penggunanya. Elemen ini mencakup visual (penglihatan), audio (pendengaran) dan sentuhan.
d. *Interactivity*, bertugas untuk merespons aksi dari pengguna, sehingga pengguna dapat berinteraksi langsung dalam medan fiktif atau *virtual world*.

Sebuah teknologi dapat dikatakan sebagai realitas virtual jika sudah memenuhi beberapa persyaratan, seperti tampilan gambar/grafis/visualisasi 3D tampak nyata dan sesuai dengan perspektif dari penggunanya, dan mampu mendeteksi semua gerakan dan respons dari pengguna baik itu gerakan kepala atau bola mata pengguna.

2.2. Unity 3D

Unity adalah aplikasi pengembangan permainan yang terintegrasi kuat dengan satu set lengkap alat intuitif dan alur kerja yang cepat untuk membuat 3D interaktif dan konten 2D. Unity merupakan *easy multiplatform publishing*. Unity memiliki took *Asset* yang menyediakan *Asset* untuk diunduh secara gratis maupun

berbayar. Terdapat pula *Unity Community* yang menyediakan tutorial secara gratis untuk semua pengguna unity. Fitur scripting yang disediakan, mendukung 3 bahasa pemrograman, yaitu JavaScript, C#, dan Boo. *Flexible* and *EasyMoving, rotating,* dan *scaling objects* hanya perlu sebaris kode. Begitu juga dengan *Duplicating, removing,* dan *changing properties*. Visual Properties Variables yang didefinisikan dengan scripts ditampilkan pada Editor. Bisa digeser, di *drag and drop*, bisa memilih warna dengan *color picker*. Berbasis .NET artinya perjalanan program dilakukan dengan *Open Source* .NET *platform*, Mono. Serta mendukung pengembangan aplikasi Microsoft, SONY, Qualcomm, BlackBerry, Samsung, Nintendo, Oculus Rift dan Intel.

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.

2.4. Leap Motion

Leap Motion Controller, alat ini adalah alat yang dapat dihubungkan ke komputer kemudian dapat digunakan layaknya menggantikan mouse maupun keyboard karena Leap Motion Controller dapat membantu penggunanya mengendalikan komputer hanya dengan gerakan tangan maupun jari diudara. Bentuk Leap Motion Controller berukuran kotak yang cukup kecil sehingga mudah dibawa kemanapun. Cara kerja Leap Motion Controller ini adalah pengguna cukup meletakan perangkat ini didekat komputer dan kemudian perangkat akan mendeteksi keberadaan tangan maupun jari yang selanjutnya pengguna dapat menggunakan gerakan tangan atau jari yang diinginkan serta gerakan kombinasi yang sudah diatur. Selain bisa menggantikan

fungsi *mouse*, Leap Motion pun bisa berubah fungsi menjadi *keyboard* dengan memancarkan sinar yang mewakili huruf dan angka pada *keyboard.* Untuk mengetik para pengguna cukup dengan menyentuh huruf layaknya mengetik pada *keyboard* biasa dengan akurasi 1/100 milimeter.

2.5. *Wall Climbing*

Wall Climbing adalah sebuah simulasi memanjat dinding yang dibuat secara artifisial dengan beberapa bentuk pegangan untuk tangan dan kaki, biasanya digunakan untuk memanjat dalam ruangan, dan juga bisa terletak diluar ruangan. Terkadang dinding terbuat dari bata dan kayu, tetapi dengan perkembangan jaman modern, bahan untuk pembuatan yang paling sering digunakan adalah papan multipleks tebal dengan lubang yang dibor ke dalamnya. Baru-baru ini, baja dan aluminium yang diproduksi juga telah digunakan. Dinding juga harus memiliki tempat untuk memasang tali pengaman, tetapi mungkin juga digunakan untuk melatih pendakian.

Setiap lubang berisi t-nut yang dibentuk khusus untuk memungkinkan pendaki memegang modular yang disekrupkan ke dinding. Dengan dinding yang terbuat dari baja atau aluminium, tali yang dipasang diatas dinding akan menjadi tali pengaman untuk pendaki. Lapisan luar dari dinding multiplek dilapisi dengan campuran beton, cat, dan poliuretan. Selain permukaan bertekstur dan pegangan tangan, dinding mungkin berisi struktur permukaan seperti indentions (incut) dan tonjolan (tonjolan), atau mengambil bentuk overhang, underhang atau retak.

Beberapa genggaman dibentuk untuk meniru kondisi batuan luar ruangan, termasuk beberapa yang terlalu besar dan dapat memiliki genggaman lain yang sesuai dengan keadaan asli dari tebing. Cara untuk melakukan simulasi ini pertama pendaki harus memakai tali pengaman yang diikatkan di pinggang untuk

mengurangi resiko pada saat pemain terlepas saat melakukan pendakian, yang kedua pemain harus memegang poin-poin pada *wall calimbing* untuk mencapai atas dengan cepat.

2.6. Oculus Rift

Oculus Rift adalah salah satu contoh alat yang sering digunakan untuk membuat sebuah permainan berbasis realitas virtual yang diciptakan oleh Oculus VR, sebuah divisi dari Facebook Inc, Oculus Rift termasuk perangkat realitas virtual yang terbilang sangat canggih, karena pada perangkat Oculus Rift tidak membutuhkan *smartphone* untuk menjalankannya, Oculus Rift memiliki perangkat lain untuk memainkannya seperti terdapat *remote* atau biasa disebut dengan istilah *controller* untuk menambahkan sensasi yang berbeda layaknya bermain dengan dunia nyata, lalu juga terdapat kamera untuk mendeteksi gerakan saat bermain, dan terakhir memiliki *stick controller Xbox* yang kegunaannya seperti *remote controller* tetapi untuk *stick controller* ini memiliki lebih banyak tombol sehingga saat memakai perangkat ini pemain hanya sekedar menggenggam alatnya lalu bermain layaknya memainkan *playstation.*

[Halaman ini sengaja dikosongkan]

BAB III
DESAIN DAN PERANCANGAN SISTEM

Bab ini membahas tentang desain dan perancangan Sistem Realitas Virtual untuk *Wall Climbing*. Pembahasan yang dilakukan meliputi analisis sistem, perancangan sistem, skenario simulasi, perancangan *user interface* dan aset aplikasi, dan perancangan antar muka sistem.

3.1. Analisis Perangkat Lunak

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. Deskripsi Umum Perangkat Lunak

Wall Climbing adalah sebuah simulasi yang dimana pemain harus memanjat dinding yang dibuat semirip mungkin dengan tebing yang diberi pegangan untuk tangan dan kaki. Simulasi yang sudah sangat terkenal hingga internasional, tetapi dengan sulitnya mencoba simulasi ini sendiri tanpa didampingi ahli dan pengalaman yang cukup. Dengan memanfaatkan teknologi realitas virtual sebagai metode baru dalam bidang simulasi dengan menggunakan alat Oculus Rift dan Leap Motion Controller sebagai fokusan dalam pembuatan Tugas Akhir ini.

Pembangunan sistem ini dimulai dari pembuatan *environment scene* yang akan dijadikan sebagai lingkungan tempat pengguna untuk melakukan simulasi *Wall Climbing* dengan teknologi realitas virtual. *Environment Scene* tersebut dibuat menggunakan tools Unity, dimana pada *environment scene* ini terdapat 2 pilihan teknologi yang berbeda yaitu dengan Oculus Rift dengan controllernya yang akan dibandingkan dengan penggunaan teknologi Leap Motion Controller. Diawali dengan menu yang

memperlihatkan pilihan *play* dan *exit*. Saat memilih *play* akan berpindah ke *scene* utama yang terdapat tembok dengan poin-poin yang bisa digenggam untuk memanjat.

3.1.2. 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 *menu* utama
F2	Pemain dapat memilih pilihan pada *menu* utama
F3	Pemain dapat memulai aplikasi
F4	Pemain dapat keluar dari aplikasi
F5	Pemain dapat memanjat pada *wall*
F6	Pemain dapat jatuh saat genggaman terlepas dari poin-poin pada *wall*

Tabel *3.2 Kebutuhan Non-Fungsional Sistem*

Kode	Deskripsi
NF1	Sistem dapat dijalankan dengan lancar
NF2	Sistem memiliki antar muka yang mudah dipahami

3.1.3. Karakteristik Pengguna

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

3.2. Perancangan Perangkat Lunak

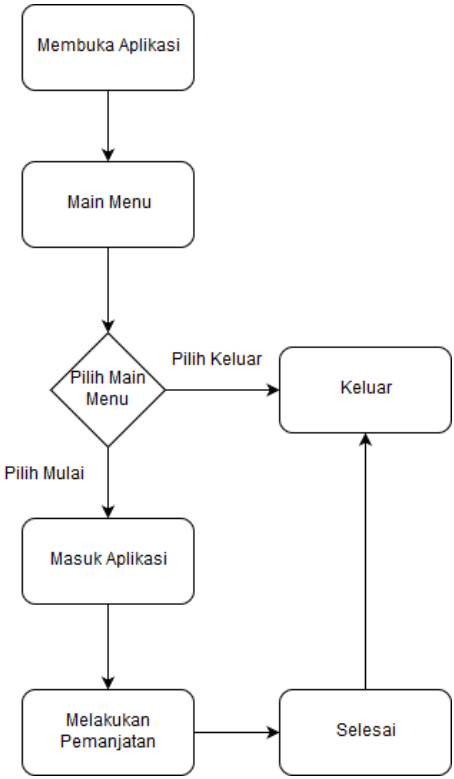
Tahap perancangan pada bab ini dibagi menjadi beberapa bagian yaitu perancangan proses dan skenario aturan main, perancangan antarmuka aplikasi, dan perancangan objek dalam *game*.

3.2.1. Perancangan Proses Simulasi

Sub bab ini membahas tentang bagaimana skenario pemain untuk melakukan panjat tebing yang akan diimplementasikan dengan 2 teknologi yang berbeda yaitu Oculus Rift dengan controller nya dan Leap Motion Controller ditambah dengan display untuk pemain dari Oculus Rift.

Secara umum proses ini dimulai dengan pemain membuka aplikasi dan akan menampilkan pilihan menu play untuk memulai kegiatan memanjat dan exit untuk keluar dari aplikasi. Pemain akan menuju ke halaman utama kegiatan yang akan terdapat objek *Wall Climbing* yang terdapat poin-poin untuk memanjat dengan cara menggengam dan menariknya kebawah maka pemain akan berpindah posisi kebagian atas sedikit demi sedikit hingga berhasil mencapai atas tembok.

Pada teknologi yang pertama yaitu Oculus Rift dengan menggunakan controllernya akan pemain akan menggunakan Oculus Headset untuk menampilkan display dari aplikasi dan memegang kedua controller dari Oculus untuk menjadi sensor dengan cara menekan triger dari controller dan akan menjadi tangan pemain pada saat memainkan aplikasi ini. Dengan teknologi yang kedua yaitu Leap Motion Controller, pertama pemain akan menggunakan Oculus Rift Headset untuk menjadi display dari pemain dan menggunakan Leap Motion Controller yang dipasang didepan Oculus Rift Headset untuk menjadi sensor dari tangan pada saat menggenggam dan tidak menggenggam sekaligus untuk menjadi kamera pengganti dari Oculus Rift Headset.



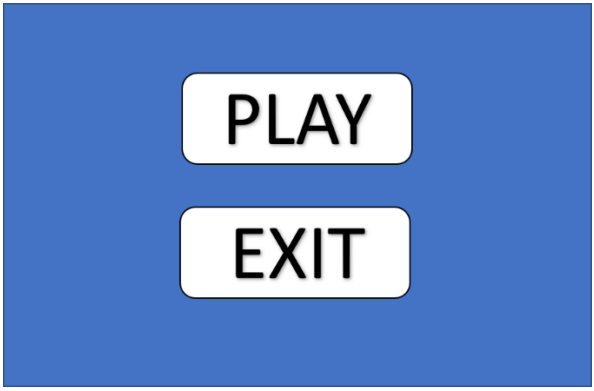
Gambar 3. 1 Diagram Alur Proses Aplikasi

3.2.2. Perancangan Antarmuka Pengguna

Subbab ini membahas bagaimana rancangan antarmuka pengguna yang akan digunakan untuk tugas akhir. Rancangan antarmuka yang dibahas meliputi ketentuan masukan dan rancangan halaman tampilan. Didalam aplikasi ini terdapat beberapa tampilan, yaitu tampilan *menu* utama aplikasi, tampilan akhir aplikasi.

3.2.2.1. Rancangan Tampilan *Menu* Utama

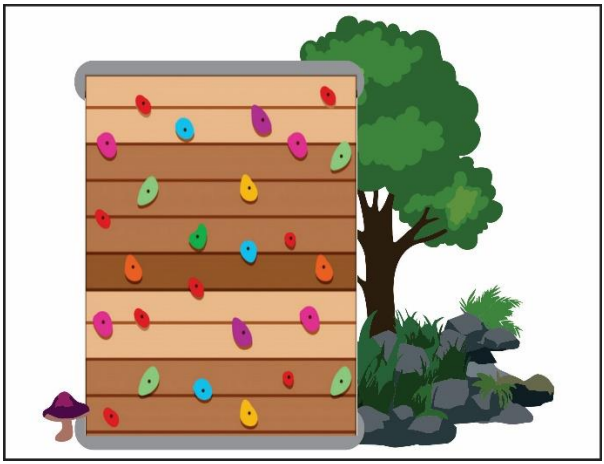
Tampilan *menu* utama terdapat dua tombol yaitu tombol play dan exit. Tampilan menu utama aplikasi dapat dilihat pada Gambar 3.2.



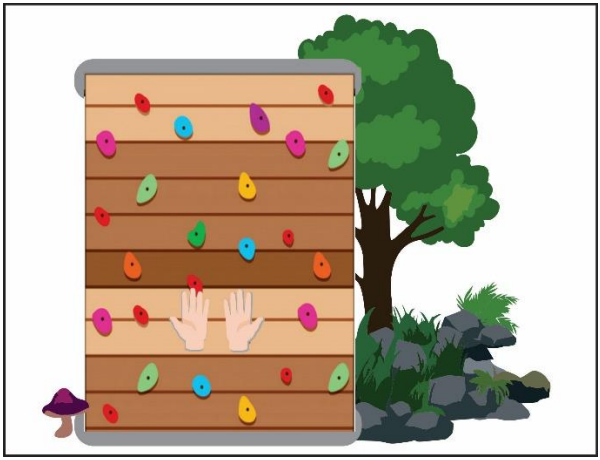
Gambar 3. 2 Rancangan Tampilan Menu Utama

3.2.2.2. Tampilan Aplikasi

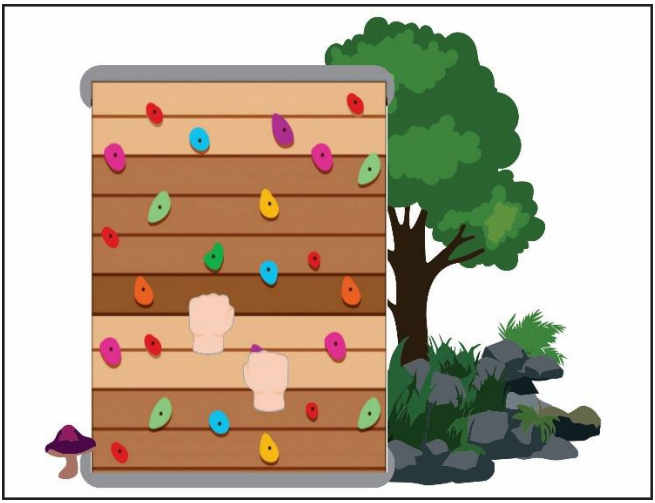
Tampilan aplikasi merupakan halaman yang muncul setelah pemain menekan tombol *play* pada menu aplikasi. Halaman ini merupakan halaman utama dari aplikasi, dimana pemain mulai melakukan interaksi terhadap dunia virtual. Pada tampilan ini terdapat tiga tampilan antarmuka yang dapat dilihat pada *Gambar 3.3* sampai dengan *Gambar 3.5*.



Gambar 3. 3 Rancangan Tampilan Ketika Pertama Baru Memulai Aplikasi



Gambar 3. 4 Rancangan Tampilan Ketika Tangan Pemain Terkena Sensor



Gambar 3. 5 Rancangan Tampilan Ketika Tangan Pemain Melakukan Grab

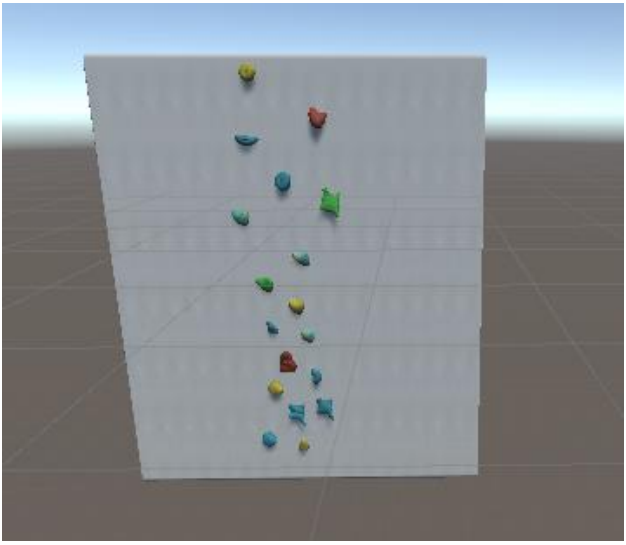
3.2.3. Perancangan Objek dalam Aplikasi

Pembuatan objek 3D dilakukan dengan memodelkan objek ke bentuk tiga dimensi terlebih dahulu. Setelah objek 3D terbentuk, selanjutnya adalah pemberian tekstur yang disesuaikan dengan objek aslinya. Sebagian besar pembuatan objek untuk *game* ini dibuat dari dalam Unity menggunakan 3D *object* unity dan beberapa *asset* yang dapat ditemukan dan diunduh melalui Asset Store Unity. Selain itu, kebanyakan model didapat melalui unduhan dari berbagai macam sumber.

Bab ini dibagi ke dalam empat subbab meliputi Perancangan Model *Wall*, Perancangan Model poin-poin pada *wall*, Perancangan Model tangan pemain, Perancangan model-model pendukung lainnya.

3.2.3.1. Perancangan Model *Wall*

Pada subbab ini dijelaskan hasil pembuatan objek *wall* sebagai inti dari aplikasi *Wall Climbing* ini. Pembuatannya, pembuat menggunakan asset Climbing Hall Pack dari *asset store* unity untuk membentuk *wall* sebagai objek utama. Objek *wall* pada aplikasi *Climbing Wall* ini dapat dilihat pada Gambar 3.6.



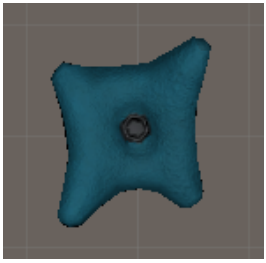
Gambar 3. 6 Wall Yang Digunakan

3.2.3.2. Perancangan Model *T-Nut* / Poin-poin pada *Wall*

Pada subbab ini dijelaskan hasil dari pembuatan poin-poin sebagai objek yang digenggam atau di *grab* oleh pemain agar dapat melakukan pemanjatan terhadap *wall.* Dengan menggunakan asset dari Runenmark Studio Climbing Hold dari *asset store* unity untuk ditambahkan pada wall yang sudah dibuat, dan meletakannya dari bawah wall hingga keatas. Model dari poin-poin ini dapat dilihat pada Gambar 3.7 sampai dengan Gambar 3.10.

Gambar 3. 7 T-Nut (1) Yang Digunakan

Gambar 3. 8 T-Nut (2) Yang Digunakan



Gambar 3. 9 T-Nut (3) Yang Digunakan

Gambar 3. 10 T-Nut (4) Yang Digunakan

3.2.3.3. Perancangan Model Tangan Pemain

Pada subbab ini dijelaskan hasil dari objek tangan yang ditampilkan ada dua karena menggunakan dua teknologi yang berbeda yaitu Model tangan dari Oculus Rift Controller dan Model tangan hasil *generate* dari Leap Motion Controller.

1. Model tangan Oculus Rift Controller

   Model tangan yang langsung dari controller tanpa menambahkan *asset* tangan dari gambar lain. Model tangan dari Oculus Rift Controller.

2. Model tangan Leap Motion Controller

   Model tangan yang langsung dari controller tanpa menambahkan *asset* tangan dari gambar lain. Model tangan dari Leap Motion Controller.

3.2.3.4. Perancangan Model-model Pendukung lainnya

Pada subbab ini dijelaskan pemakaian dan sebab penggunaan model-model 3D lain sebagai pendukung aplikasi. Model-model 3D didapat dari *Asset Store* Unity dari Fantasy Forest Set dengan mengunduh dan *import* gambar. Aset yang digunakan untuk menambah *environment* yang ada didalam aplikasi sehingga pemain merasa lebih menarik dan *immersive* pada saat bermain.

Model-model pendukung lain dapat dilihat pada Gambar 3.11 sampai dengan Gambar 3.13.

Gambar 3. 11 Pohon (1) Yang Digunakan

Gambar 3. 12 Pohon (2) Yang Digunakan

Gambar 3. 13 Pohon (3) Yang Digunakan

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#.

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.

Table 4. 1 Spesifikasi Perangkat

Perangkat	Spesifikasi
Perangkat Keras	• Prosesor: Intel® Core™ i7-7700U CPU @ 3.60GHz (4 CPUs), ~3.6GHz • Memori: 8192MB • VGA : NVIDIA GeForce GTX 1060 3Gb • Oculus Rift DK2 • Leap Motion
Perangkat Lunak	• Sistem Operasi Microsoft Windows 10 64-bit • Perangkat Pengembang Unity3D 2018.3.5f1 • Perangkat Pembantu Visual Studio Community 2017, Microsoft Word 2016

4.2. Implementasi Kontrol Hardware

Implementasi dari masing-masing alat yang digunakan untuk membangun aplikasi dimulai dari koneksi hingga deteksi sensor atau input untuk hardware yang digunakan.

4.2.1. Oculus Touch

1. Melakukan *download* dan *import* Oculus Integration melalui *asset store* unity.
2. Menggunakan LocalAvatarWithGrab untuk mendapatkan objek kamera dan sensor Oculus Touch.
3. Menambahkan OVRGrabber pada LeftHandAnchor untuk menjadikan tangan kiri bisa melakukan grab kepada objek dapat dilihat pada Kode Sumber 4.4.
4. Menambahkan OVRGrabber pada RightHandAnchor untuk menjadikan tangan kiri bisa melakukan grab kepada objek dapat dilihat pada Kode Sumber 4.4.
5. Menambahkan *script* untuk mengubah benda menjadi *anchor/*tumpuan pada saat melakukan *grab* dapat dilihat pada Kode Sumber 4.5.

4.2.2. Leap Motion

1. Install Leap Motion SDK Setup.
2. Melakukan *download* dan *import* Leap Motion Core Asset.
3. Menggunakan Leap Rig untuk mendapatkanobjek kamera dan sensor dari Leap Motion.
4. Melakukan *download* dan *import* Leap Motion Interaction Engine Module.
5. Menggunakan Interaction Manager agar tangan dari pemain bisa melakukan interaksi terhadap objek lain.
6. Menambahkan Attachment Hands untuk mendeteksi tangan terbuka atau tertutup dapat dilihat pada Kode Sumber 4.8.

7. Menambahkan *script* untuk mengubah benda menjadi *anchor/*tumpuan pada saat melakukan *grab* 4.9.

4.3. Implementasi Pembuatan Aplikasi

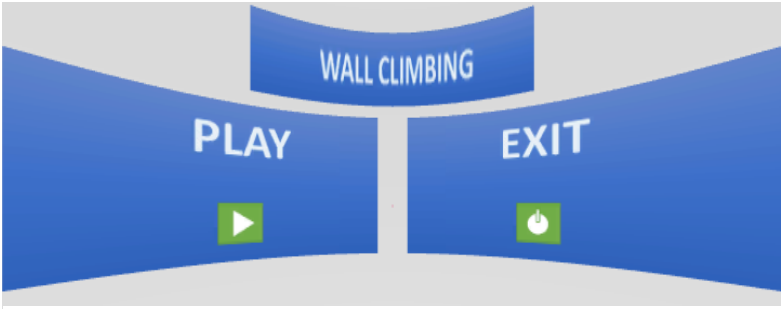
Implementasi dari masing-masing fungsi utama dituliskan menggunakan kode berbahasa C#. implementasi fungsi diurut berdasarkan antarmuka-antarmuka yang ada pada aplikasi.

4.3.1. Implementasi Menu Aplikasi

Pada subbab ini akan dibagi menjadi dua implementasi karena membuat dua aplikasi yang berbeda dengan teknologi Oculus Rift dan Leap Motion Controller.

4.3.1.1. Menu Aplikasi pada Aplikasi Oculus Rift

Gambar 4.1 merupakan tampilan implementasi dari menu aplikasi yang akan dijelaskan, sebagai berikut:



Gambar 4. 1 Tampilan Menu Aplikasi Pada Teknologi Oculus Rift

Pada Gambar 4.1 terdapat 2 tombol yaitu, *Play*, dan *Exit*. Berikut penjelasan dari tiap tombol:

1. *Play* untuk memulai aplikasi.
2. *Exit* untuk keluar dari aplikasi.

Pada Kode Sumber 4.1, terdapat beberapa fungsi untuk menjalankan halaman awal aplikasi ini. Diantaranya yaitu untuk pertama kali memulai permaian, dan untuk keluar dari aplikasi.

```
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("maingame");
11.     }
12.     public void exitgame()
13.     {
14.         Application.Quit();
15.         print("Quit");
16.     }
17. }
```

Kode Sumber 4. 1 Kontrol Aksi Tombol Menu (Oculus Rift)

4.3.1.2. Menu Aplikasi pada Aplikasi Leap Motion Controller

Gambar 4.2 merupakan tampilan implementasi dari menu aplikasi yang akan dijelaskan, sebagai berikut:

Gambar 4. 2 Tampilan Menu Aplikasi Pada Teknologi Leap Motion

Pada Gambar 4.2 terdapat 2 tombol yaitu, *Play*, dan *Exit*. Berikut penjelasan dari tiap tombol:

1. *Play* untuk memulai aplikasi.
2. *Exit* untuk keluar dari aplikasi.

Pada Kode Sumber 4.2, terdapat beberapa fungsi untuk menjalankan halaman awal aplikasi ini. Diantaranya yaitu untuk pertama kali memulai permaian, dan untuk keluar dari aplikasi.

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.  using UnityEngine.SceneManagement;
5.
6.  public class scenemanagement : MonoBehaviour
7.  {
8.      public void playgame ()
9.      {
10.         SceneManager.LoadScene("maingame");
11.     }
12.     public void exitgame()
13.     {
14.         Application.Quit();
15.         print("Quit");
16.     }}
```

Kode Sumber 4. 2 Kontrol Aksi Tombol Menu (Leap Motion)

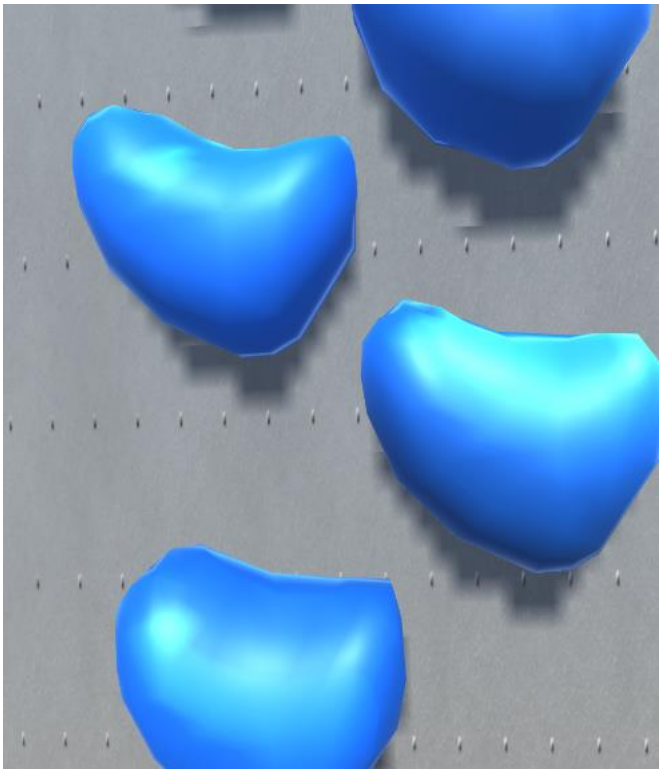
4.3.2. Implementasi Aplikasi

Tampilan aplikasi merupakan halaman tempat pemain melakukan interaksi. Pada halaman aplikasi, aplikasi dapat melakukan kegiatan memanjat *wall* dengan menggunakan tangan yang memakai 2 teknolgi yaitu Oculus Rift beserta controllernya dan Leap Motion Controller.

4.3.2.1. Fitur Utama Aplikasi pada Aplikasi Oculus Rift

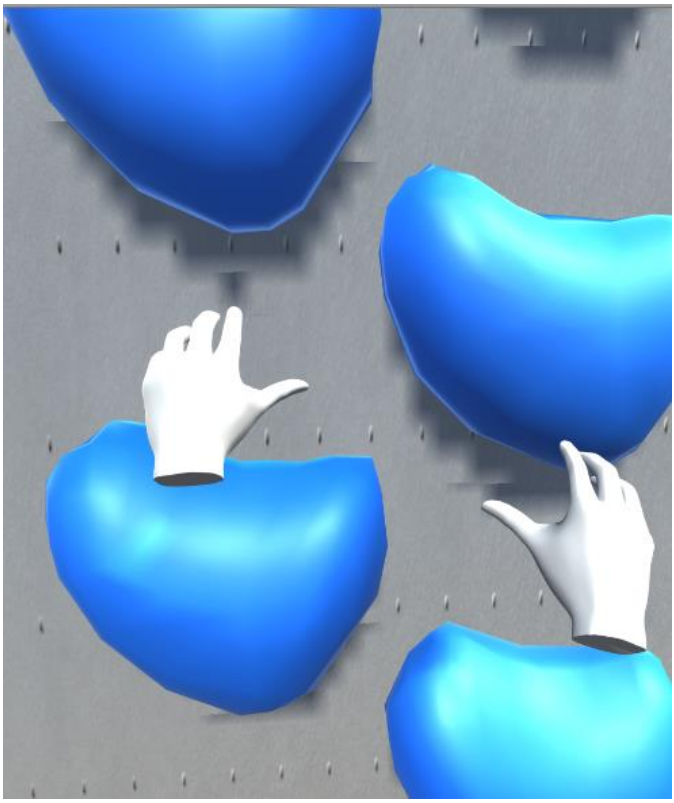
Pada implementasi ini pertama kali memasuki aplikasi pada saat menekan tombol pada menu. Tampilan ketika tidak melakukan kegiatan apapun.

Gambar 4. 3 Tampilan Awal Aplikasi



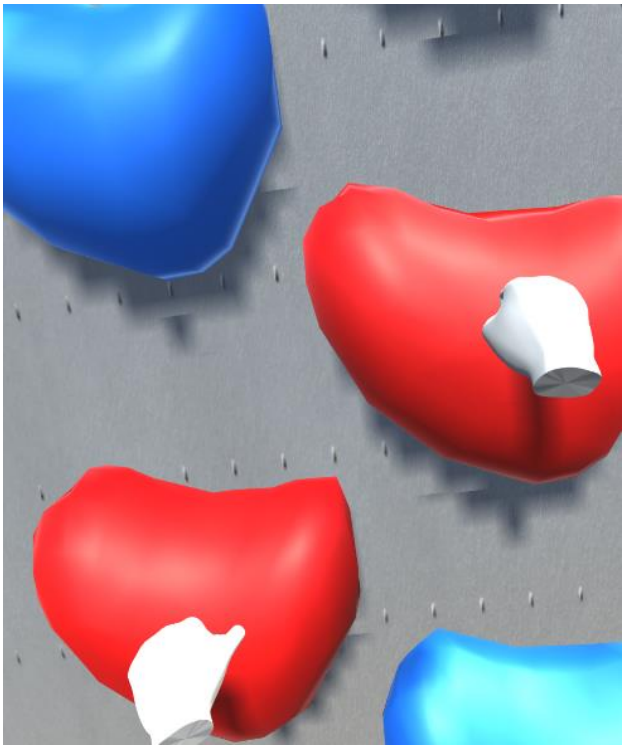
Gambar 4. 4 Tampilan Awal Aplikasi Dari Headset Oculus

Pada implementasi ini disaat pemain sedang menggunakan controller dari Oculus Rift tetapi belum melakukan kegiatan apapun dan posisi pemain berada pada bagian bawah *wall*.



Gambar 4. 5 Tampilan Saat Sensor Tangan Terdeteksi

Pada implementasi ini dimana pemain sedang melakukan kegiatan memanjat di *wall* yang disediakan dengan cara menggenggam poin-poin yang ada pada *wall* maka pemain akan berpindah posisi dari bawah hingga atas sesuai dengan poin yang digenggam.



Gambar 4. 6 Tampilan Saat Tangan Melakukan Grab

Implementasi fungsi dari tiap objek tindakan yang ada pada aplikasi *Wall Climbing* menggunakan Oculus Rift dengan Oculus Touch sebagai controller nya sebagai berikut :

1. Objek *t-nut /* poin
   Objek poin atau yang biasa dikatakan *t-nut* pada sistem panjat tebing buatan bergfungsi sebagai alat bantu untuk pemain memanjat dengan cara memegang dan menarik kebawah *t-nut* dan akan diberi fungsi agar bisa untuk di *grab* fungsi tersebut dapat dilihat pada Kode Sumber 4.3.

```
1.  using System;
2.  using UnityEngine;
3.
4.  public class OVRGrabbable : MonoBehaviou
    r
5.  {
6.  protected bool m_allowOffhandGrab = true
    ;
7.  protected bool m_snapPosition = false;
8.  protected bool m_snapOrientation = false
    ;
9.  protected Transform m_snapOffset;
10. protected Collider[] m_grabPoints = null
    ;
11. protected bool m_grabbedKinematic = fals
    e;
12. protected Collider m_grabbedCollider = n
    ull;
13. protected OVRGrabber m_grabbedBy = null;
14. void Awake()
15.     {
16.         if (m_grabPoints.Length == 0)
17.         {
18. // Get the collider from the grabbable
19.           Collider collider = this.GetCo
    mponent<Collider>();
20.            if (collider == null)
21.             {
22.                 throw new ArgumentExcept
    ion("Grabbables cannot have zero grab po
    ints and no collider -
    - please add a grab point or collider.")
    ;
23. }
24. // Create a default grab point
25.             m_grabPoints = new Collider[
    1] { collider };
26.         }
27.     }
28.       protected virtual void Start()
29.     {
```

```
30.         m_grabbedKinematic = GetComponen
    t<Rigidbody>().isKinematic;
31.     }
32. void OnDestroy()
33.     {
34.       if (m_grabbedBy != null)
35.         {
36. // Notify the hand to release destroyed
    grabbables
37.             m_grabbedBy.ForceRelease(thi
    s);
38.         }
39.     }
40. }
```

Kode Sumber 4. 3 Fungsi Agar Benda Bisa di Grab

2. Objek tangan
   Objek tangan pada pemain disini berfungsi sebagai kontrol dalam aplikasi *Wall Climbing* ini, kedua tangan berfungsi untuk melakukan *grab* / menggenggam ke bagian t-*nut* pada dinding yang dapat dilihat pada Kode Sumber 4.4.

```
1.  using System.Collections.Generic;
2.  using UnityEngine;
3.
4.  /// Allows grabbing and throwing of obje
    cts with the OVRGrabbable component on t
    hem.
5.  [RequireComponent(typeof(Rigidbody))]
6.  public class OVRGrabber : MonoBehaviour
     {
7.      // Grip trigger thresholds for picki
    ng up objects, with some hysteresis.
8.      public float grabBegin = 0.55f;
9.      public float grabEnd = 0.35f;
10.
11.     protected bool m_parentHeldObject =
    false;
```

```
12. protected Transform m_gripTransform = nu
    ll;
13. protected Collider[] m_grabVolumes = nul
    l;
14. protected OVRInput.Controller m_controll
    er;
15. protected Transform m_parentTransform;
16. protected bool m_grabVolumeEnabled = tru
    e;
17. protected Vector3 m_lastPos;
18. protected Quaternion m_lastRot;
19. protected Quaternion m_anchorOffsetRotat
    ion;
20. protected Vector3 m_anchorOffsetPosition
    ;
21. protected float m_prevFlex;
22. protected OVRGrabbable m_grabbedObj = nu
    ll;
23. protected Vector3 m_grabbedObjectPosOff
24. protected Quaternion m_grabbedObjectRotO
    ff;
25. protected Dictionary<OVRGrabbable, int>
    m_grabCandidates = new Dictionary<OVRGra
    bbable, int>();
26. protected bool operatingWithoutOVRCamera
    Rig = true;
27. public bool grabed;
28. /// The currently grabbed object.
29. public OVRGrabbable grabbedObject
30.     {
31.         get { return m_grabbedObj; }
32.     }
33.
34. public void ForceRelease(OVRGrabbable gr
    abbable)
35.     {
36.         bool canRelease = (
37.             (m_grabbedObj != null) &&
38.             (m_grabbedObj == grabbable)

39.             );
40.         if (canRelease)
```

```
41.         {
42.             GrabEnd();
43.         }
44.     }
45. void OnTriggerEnter(Collider otherCollid
    er)
46.     {
47. // Get the grab trigger
48.         OVRGrabbable grabbable = otherCo
    llider.GetComponent<OVRGrabbable>() ? ?
    otherCollider.GetComponentInParent<OVRGr
    abbable>();
49. if (grabbable == null) return;
50.
51. // Add the grabbable
52.         int refCount = 0;
53.         m_grabCandidates.TryGetValue(gra
    bbable, out refCount);
54.         m_grabCandidates[grabbable] = re
    fCount + 1;
55.     }
56.
57. void OnTriggerExit(Collider otherCollide
    r)
58.     {
59.         OVRGrabbable grabbable = otherCo
    llider.GetComponent<OVRGrabbable>() ? ?
    otherCollider.GetComponentInParent<OVRGr
    abbable>();
60. if (grabbable == null) return;
61.
62. // Remove the grabbable
63.         int refCount = 0;
64.         bool found = m_grabCandidates.Tr
    yGetValue(grabbable, out refCount);
65.         if (!found)
66.         {
67.             return;
68.         }
69.
70.         if (refCount > 1)
71.         {
```

```
72.             m_grabCandidates[grabbable]
    = refCount - 1;
73.         }
74.         else
75.         {
76.             m_grabCandidates.Remove(grab
    bable);
77.         }
78.     }
```

Kode Sumber 4. 4 Fungsi Untuk Memegang Benda

3. Tindakan *grab* / memegang
   Grab / memegang disini adalah proses disaat tangan pemain yang terkena sensor dari Oculus Rift Controller akan membentuk sebuah tangan dan akan diberi fungsi memegang kepada objek *t-nut* yang sudah disediakan pada *wall* dapat dilihat pada Kode Sumber 4.5 dan Kode Sumber 4.6.

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.
5.  public class pull : MonoBehaviour
6.  {
7.      public OVRGrabber handGrab;
8.      public Rigidbody rbBody;
9.      public GameObject Body;
10.     private Rigidbody palmRB;
11.     private Vector3 boPos;
12.
13.     public Vector3 prevPos;
14.     public bool canGrip;
15.     // Start is called before the first
    frame update
16. void Start()
17.     {
18.         prevPos = gameObject.transform.l
    ocalPosition;
```

```
19.         palmRB = gameObject.GetComponent
    <Rigidbody>();
20.     }
21. // Update is called once per frame
22. void Update()
23.     {
24.         if (canGrip && handGrab.grabed =
    = true)
25.         {
26.             canGrip = true;
27.             rbBody.useGravity = false;
28.             rbBody.isKinematic = true;
29.             Body.transform.position += (
    prevPos - gameObject.transform.localPosi
    tion);
30.         }
31.         else
32.         {
33.             canGrip = false;
34.             rbBody.useGravity = true;
35.             rbBody.isKinematic = false;
36.         }
37.         prevPos = gameObject.transform.l
    ocalPosition;
38.     }
39. private void OnTriggerStay(Collider othe
    r)
40.     {
41.         canGrip = true;
42.         palmRB.constraints = RigidbodyCo
    nstraints.FreezeAll;
43.     }
44. private void OnTriggerExit(Collider othe
    r)
45.     {
46.         canGrip = false;
47.         palmRB.constraints = RigidbodyCo
    nstraints.None;
48.     }
49. }
```

Kode Sumber 4. 5 Fungsi Untuk Mengubah Benda Sebagai Tumpuan

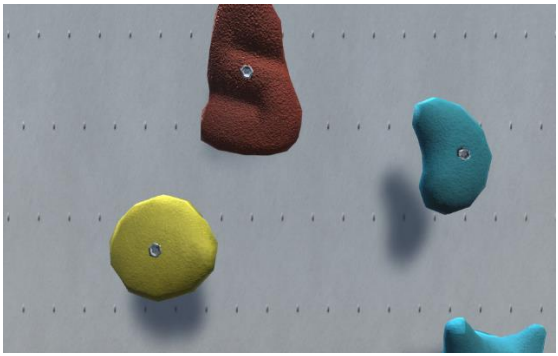
```
1.  public class changecolor : MonoBehaviour
2.  {
3.      public GameObject batu;
4.      public Color warna;
5.      public Color defaultColor;
6.      // Start is called before the first
    frame update
7.      void Start()
8.      {
9.          defaultColor = batu.GetComponent
    <Renderer>().sharedMaterial.color;
10.         warna = batu.GetComponent<Render
    er>().sharedMaterial.color;
11.     }
12.     // Update is called once per frame
13.     void Update()
14.     {
15.     }
16.     public void Changecolor()
17.     {
18.         //Debug.Log("berubah");
19.         batu.GetComponent<Renderer>().sh
    aredMaterial.color = Color.red;
20.     }
21.     public void Changecolor1()
22.     {
23.         //Debug.Log("exit");
24.         batu.GetComponent<Renderer>().sh
    aredMaterial.color = defaultColor;
25.     }
26. }
```

Kode Sumber 4. 6 Fungsi Untuk Mengubah Warna T-nut

4.3.2.2. Fitur Utama Aplikasi pada Aplikasi Leap Motion Controller

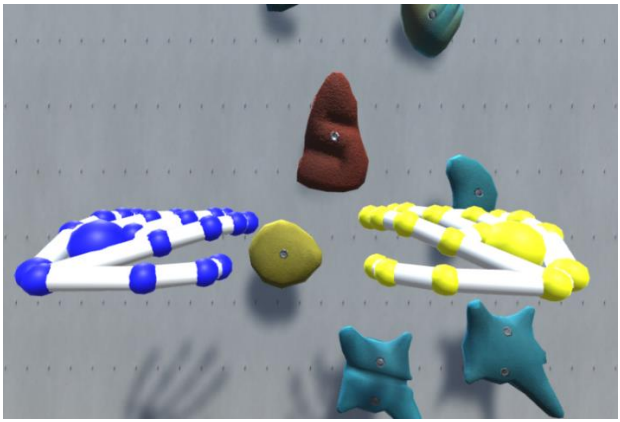
Pada implementasi ini pertama kali memasuki aplikasi pada saat menekan tombol *play* pada menu. Tampilan ketika tidak melakukan kegiatan apapun.

Gambar 4. 7 Tampilan Saat Awal Aplikasi



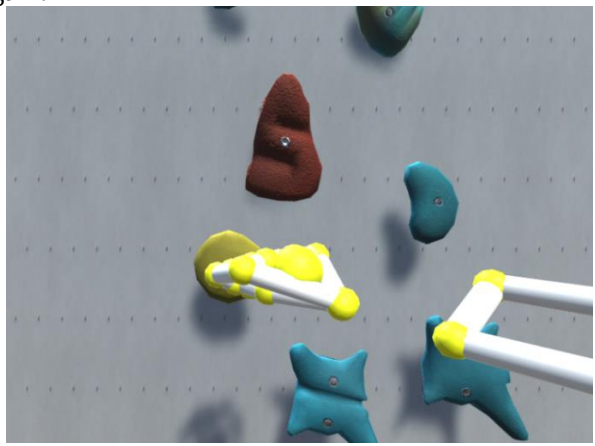
Gambar 4. 8 Tampilan Saat Awal Aplikasi Pada Headset Oculus

Pada implementasi ini disaat pemain sedang meletakan tangan didepan sensor dari Leap Motion controller tetapi belum melakukan kegiatan apapun dan posisi pemain berada pada bagian bawah *wall*.



Gambar 4. 9 Tampilan Saat Sensor Tangan Terdeteksi Leap Motion

Pada implementasi ini dimana pemain sedang melakukan kegiatan memanjat di *wall* yang disediakan dengan cara menggenggam poin-poin yang ada pada *wall* maka pemain akan berpindah posisi dari bawah hingga atas sesuai dengan poin yang digenggam.



Gambar 4. 10 Tampilan Saat Melakukan Grab

Implementasi fungsi dari tiap objek tindakan yang ada pada aplikasi *Wall Climbing* yang menggunakan teknologi Oculus Rift dengan Leap Motion Controller ini sebagai berikut :

1. Objek *t-nut* / poin
   Objek poin atau yang biasa dikatakan *t-nut* pada sistem panjat tebing buatan berfungsi sebagai alat bantu untuk pemain memanjat dengan cara memegang dan menarik kebawah *t-nut* dan akan diberi fungsi agar bisa untuk di *grab* fungsi tersebut dapat dilihat pada Kode Sumber 4.7.

```
1.  public void BeginGrasp(List<InteractionC
    ontroller> controllers) {
2.          _justGrasped = true;
3.   if (isSuspended) {
4.              _suspendingController.ReleaseG
    rasp();
5.          }
6.   if (!allowMultiGrasp && isGrasped) {
7.              _graspingControllers.Query().F
    irst().ReleaseGrasp();
8.          }
9.   foreach(var controller in controllers)
    {
10.             _graspingControllers.Add(contr
    oller);
11.  if (moveObjectWhenGrasped) {
12.                 graspedPoseHandler.AddCont
    roller(controller);
13.             }
14.         OnPerControllerGraspBegin(controll
    er);
15.         }
16.  if (_graspingControllers.Count == contr
    ollers.Count) {
17.             _dragBeforeGrasp = rigidbody.d
    rag;
18.             _angularDragBeforeGrasp = rigi
    dbody.angularDrag;
```

```
19.             _wasKinematicBeforeGrasp = rig
    idbody.isKinematic;
20. switch (graspedMovementType) {
21. case GraspedMovementType.Inherit: break;
22. case GraspedMovementType.Kinematic:
23. rigidbody.isKinematic = true; break;
24. case GraspedMovementType.Nonkinematic:
25. rigidbody.isKinematic = false; break;
26.             }
27.             rigidbody.drag = 0F;
28.             rigidbody.angularDrag = 0F;
29.             OnGraspBegin();
30.         }
31.     }
32. public void EndGrasp(List<InteractionCon
    troller> controllers) {
33. if (_graspingControllers.Count == contro
    llers.Count && isSuspended) {
34.             EndSuspension(controllers[0]);
35.         }
36. foreach(var controller in controllers) {

37.             _graspingControllers.Remove(co
    ntroller);
38.             OnPerControllerGraspEnd(contro
    ller);
39. if (moveObjectWhenGrasped) {
40.                 graspedPoseHandler.RemoveC
    ontroller(controller);
41.             }
42.         }
43. if (_graspingControllers.Count == 0) {
44.             rigidbody.drag = _dragBeforeGr
    asp;
45.             rigidbody.angularDrag = _angul
    arDragBeforeGrasp;
46.
47.             rigidbody.isKinematic = _wasKi
    nematicBeforeGrasp;
48. if (controllers.Count == 1) {
49.                 throwHandler.OnThrow(this,
     controllers.Query().First());
```

```
            }
            OnGraspEnd();
if (_justGrasped) _justGrasped = false;

        }
    }
```

Kode Sumber 4. 7 Fungsi Agar Benda Bisa di Grab

2. Objek tangan

Objek tangan pada pemain disini berfungsi sebagai kontrol dalam aplikasi *Wall Climbing* ini, kedua tangan berfungsi untuk melakukan *grab* / menggenggam ke bagian t-*nut* pada dinding sebelumnya harus mengetahui bagian-bagian ruas dari tangan dan juga mengetahui tangan terbuka atau tertutup yang dapat dilihat pada Kode Sumber 4.8 dan Kode Sumber 4.9.

```
using Leap.Unity.Attributes;
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEditor;
using UnityEngine;
namespace Leap.Unity.Attachments{
public class AttachmentHands : MonoBehaviour {
private AttachmentPointFlags _attachmentPoints = AttachmentPointFlags.Palm | AttachmentPointFlags.Wrist;
public AttachmentPointFlags attachmentPoints {
        get {
          return _attachmentPoints;
        }
        set {
          if (_attachmentPoints != value) {
            Undo.IncrementCurrentGroup();
```

```
17.             Undo.SetCurrentGroupName("Modify Attachment Points");
18.
19.             Undo.RecordObject(this, "Modify AttachmentHands Points");
20.             _attachmentPoints = value;
21.             refreshAttachmentHandTransforms();
22.           }
23.         }
24.       }
25. void Update() {
26.         PrefabType prefabType = PrefabUtility.GetPrefabType(this.gameObject);
27. if (prefabType == PrefabType.Prefab || prefabType == PrefabType.ModelPrefab) {
28.           return;
29.         }
30.         bool requiresReinitialization = false;
31.         using (new ProfilerSample("Attachment Hands Update", this.gameObject)) {
32. for (int i = 0; i < _attachmentHands.Length; i++) {
33.             var attachmentHand = attachmentHands[i];
34. if (attachmentHand == null) {
35.               requiresReinitialization = true;
36.               break;
37.             }
38.             var leapHand = handAccessors[i]();
39.             attachmentHand.isTracked = leapHand != null;
40.
41.             using (new ProfilerSample(attachmentHand.gameObject.name + " Update Points")) {
42.               foreach(var point in attachmentHand.points) {
```

```
                point.SetTransformUsingHand(leapHand);
              }
            }
          }
          if (requiresReinitialization) {
            reinitialize();
          }
        }
      }
```

Kode Sumber 4. 8 Fungsi Untuk Mengecek Ruas-Ruas Jari

```
using UnityEngine;
using System.Collections;
using System;
using Leap.Unity.Attributes;
namespace Leap.Unity{
public class ExtendedFingerDetector : Detector {
public Camera camera;
public bool grab = false;
public float Period = .1f; //seconds
private IEnumerator watcherCoroutine;
void Awake() {
        watcherCoroutine = extendedFingerWatcher();
      }
IEnumerator extendedFingerWatcher() {
        Hand hand;
        while (true) {
bool fingerState = false;
if (HandModel != null && HandModel.IsTracked) {
            hand = HandModel.GetLeapHand();
if (hand != null) {
              fingerState = matchFingerState(hand.Fingers[0], Thumb)
                && matchFingerState(hand.Fingers[1], Index)
```

```
23.                 && matchFingerState(hand
    .Fingers[2], Middle)
24.                 && matchFingerState(hand
    .Fingers[3], Ring)
25.                 && matchFingerState(hand
    .Fingers[4], Pinky);
26. int extendedCount = 0;
27. for (int f = 0; f < 5; f++) {
28. if (hand.Fingers[f].IsExtended) {
29.                   extendedCount++;
30.                 }
31.               }
32.               fingerState = fingerState
    &&
33.                             (extendedCoun
    t <= MaximumExtendedCount) &&
34.                             (extendedCoun
    t >= MinimumExtendedCount);
35. //not extended finger
36. if (HandModel.IsTracked && fingerState)
37.               {
38.                 Activate();
39.                 grab = true;
40.               }
41. //extended finger
42. else if (!HandModel.IsTracked || !finger
    State)
43.               {
44.                 Deactivate();
45.                 grab = false;
46.               }
47.             }
48.           }
49. else if (IsActive) {
50.         Deactivate();
51.       }
52.       yield return new WaitForSeconds(Pe
    riod);
53.         }
54.       }
55. private bool matchFingerState(Finger fin
    ger, PointingState requiredState) {
```

```
56. return (requiredState == PointingState.E
    ither) ||
57.          (requiredState == PointingState
    .Extended && finger.IsExtended) ||
58.          (requiredState == PointingState
    .NotExtended && !finger.IsExtended);
59.     }
60. }
61. public enum PointingState { Extended, No
    tExtended, Either }
62. }
```

Kode Sumber 4. 9 Fungsi Untuk Mengetaui Extended Finger

3. Tindakan *grab* / memegang
   Grab / memegang disini adalah proses disaat tangan pemain yang terkena sensor dari Leap Motion Controller akan membentuk sebuah tangan dan akan diberi fungsi memegang kepada objek *t-nut* yang sudah disediakan pada *wall* dapat dilihat pada kode sumber 4.10.

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.  using Leap.Unity;
5.
6.  public class pull : MonoBehaviour
7.  {
8.  public ExtendedFingerDetector handGrab;

9.  public Rigidbody Body;
10. private Rigidbody palmRB;
11. public Vector3 prevPos;
12. public bool canGrip;
13. void Start()
14.     {
15.         prevPos = gameObject.transform.l
    ocalPosition;
16.         palmRB = gameObject.GetComponent
    <Rigidbody>();
17.     }
```

```
18.
19.     // Update is called once per frame
20. void Update()
21.     {
22.         if (canGrip && handGrab.grab ==
    true)
23.         {
24.             Body.useGravity = false;
25.             Body.isKinematic = true;
26.             Body.transform.position += (
    prevPos - gameObject.transform.localPosi
    tion)*7;
27.         }
28.         else
29.         {
30.             Body.useGravity = true;
31.             Body.isKinematic = false;
32.         }
33.         prevPos = gameObject.transform.l
    ocalPosition;
34.     }
35.     private void OnTriggerEnter(Collider
     other)
36.     {
37.         canGrip = true;
38.         palmRB.constraints = RigidbodyCo
    nstraints.FreezeAll;
39.     }
40.     private void OnTriggerExit(Collider
    other)
41.     {
42.         canGrip = false;
43.         palmRB.constraints = RigidbodyCo
    nstraints.None;
44.     }
45. }
```

Kode Sumber 4. 10 Fungsi Menjadikan Benda Sebagai Tumpuan (2)

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan mengenai rangkaian uji coba dan evaluasi yang dilakukan. Proses pengujian dilakukan menggunakan metode *blackbox* berdasarkan skenario yang telah ditentukan.

5.1. Lingkungan Pengujian

Lingkungan penguijan sistem pada pengerjaan tugas ini dilakukan pada lingkungan dan alat kakas pada Tabel 5.1 berikut:

Tabel 5. 1 Lingkungan Pengujian Sistem

Perangkat	Spesifikasi
Perangkat Keras	• Prosesor: Intel® Core™ i7-7700U CPU @ 3.60GHz (4 CPUs), ~3.6GHz • Memori: 8192MB • VGA : NVIDIA GeForce GTX 1060 3Gb • Oculus Rift DK2 • Leap Motion
Perangkat Lunak	• Sistem Operasi Microsoft Windows 10 64-bit • Perangkat Pengembang Unity3D 2018.3.5f1 • Perangkat Pembantu Visual Studio Community 2017, Microsoft Word 2016

5.2. Pengujian Fungsionalitas

Pengujian fungsionalitas aplikasi ini dapat dilakukan secara mandiri. Pengujian ini bertujuan untuk mengetahui kesesuaian keluaran dari setiap tahap dan langkah penggunaan fitur terhadap skenario yang dipersiapkan. Skenario yang dibuat

mengacu pada subbab 3.2. Skenario uji coba fungsionalitas yang dilakukan terhadap aplikasi yang dibangun dijelaskan pada Tabel 5.2.

Tabel 5. 2 Skenario Uji Coba Fungsionalitas

Kode Uji Coba	Deskripsi Uji Coba
UF-001	Uji coba pada menu aplikasi
UF-002	Uji coba dalam aplikasi

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

5.2.1. Uji Coba Menu Aplikasi

Pada subbab 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 aplikasi yang akan diuji adalah fungsionalitas tombol yang terdapat di menu utama, yaitu tombol *Play* dan *Exit*. Tampilan menu aplikasi dapat dilihat pada Gambar 4.1. Skenario yang telah diuji terdapat pada Tabel 5.2.

Tabel 5. 3 Hasil Uji Coba Menu Aplikasi

ID	**UF-001**
Nama	Uji Coba Pada Menu Aplikasi
Tujuan uji coba	Pengguna mengetahui fungsionalitas tombol yang ada pada menu aplikasi
Kondisi awal	Pemain berada pada menu aplikasi
Skenario 1	***Pemain memilih tombol Play***
Masukan	Menekan tombol *Play* pada dunia virtual
Keluaran yang diharapkan	Pemain berpindah ke halaman utama aplikasi

Hasil uji coba	Berhasil
Kondisi Akhir	Pemain berada pada halaman aplikasi
Skenario 2	***Pemain memilih tombol Exit***
Masukan	Menekan tombol *Exit* pada dunia virtual
Keluaran yang diharapkan	Pemain berpindah keluar dari aplikasi
Hasil uji coba	Berhasil
Kondisi Akhir	Pemain berpindah keluar aplikasi

Hasil uji dari skenario 1 berpindah ke halaman aplikasi dapat dilihat pada Gambar 4.3 dan skenario 2 keluar dari aplikasi, pemain akan keluar dari aplikasi.

5.2.2. Uji Coba Aplikasi

Pada subbab ini dijelaskan mengenai skenario yang dilakukan dan hasil yang didapatkan dari pengujian fungsionalitas pada aplikasi. Penjelasan disajikan dengan menampilkan kondisi awal, masukan, keluaran, hasil yang dicapai, dan kondisi akhir. Skenario yang telah diuji terdapat pada Tabel 5.3.

Tabel 5. 4 Hasil Uji Coba Aplikasi

ID	**UF-002**
Nama	Uji Coba Pada Aplikasi
Tujuan uji coba	Pengguna mengetahui fungsionalitas interaksi yang ada pada aplikasi
Kondisi awal	Pemain berada pada halaman aplikasi
Skenario 1	***Pemain bergerak keatas***
Masukan	Menggenggam *t-nut* pada dinding

Keluaran yang diharapkan	Pemain bergerak naik
Hasil uji coba	Berhasil
Kondisi Akhir	Pemain bergerak naik
Skenario 2	***Pemain terjatuh***
Masukan	Melepaskan genggaman pada *t-nut*
Keluaran yang diharapkan	Pemain terjatuh menuju dasar pada dinding
Hasil uji coba	Berhasil
Kondisi Akhir	Pemain terjatuh dan berada dibawah

Hasil dari skenario 1 yaitu pemain bergerak keatas karena adanya gerakan menggenggam dan menarik kebawah. Pada skenario 2 pemain yang sudah berada melayang di dinding mencoba melepaskan genggaman dari *t-nut* maka pemain akan terjatuh kembali kedasar dari dinding.

5.2.3. Hasil Uji Coba

Pada sub bab ini diberikan hasil evaluasi dari pengujian yang dilakukan pada aplikasi. Hasil evaluasi dapat dilihat pada Tabel 5.5

Tabel 5. 5 Hasil Evaluasi

ID	Deskripsi	Kemungkinan / Skenario	Perilaku Terlaksana
UF-001	Uji Coba Menu Aplikasi	Skenario 1	Ya
		Skenario 2	Ya
UF-002	Uji Coba Aplikasi	Skenario 1	Ya
		Skenario 2	Ya

5.3. Pengujian Pengguna

Pengujian pada aplikasi 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 aplikasi yang dibangun dari sisi pengguna. Selain dari segi keberhasilan aplikasi yang dapat dijalankan sesuai keinginan disini penulis juga ingin mengetahui tingkat kepuasan terhadap 2 teknologi berbeda yang digunakan seperti kekurangan dan kelebihan masing-masing. Hal ini dapat dicapai dengan meminta penilaian dan tanggapan dari pengguna terhadap sejumlah aspek aplikasi yang ada.

5.3.1. Skenario Pengujian Pengguna

Dalam melakukan pengujian aplikasi, pengguna diminta mencoba menjalankan aplikasi untuk mencoba semua fungsionalitas dan fitur yang ada. Pengujian aplikasi oleh pengguna dilakukan dengan sebelumnya memberikan informasi seputar aplikasi, kegunaan, dan fitur-fitur yang dimiliki. Setelah informasi tersampaikan, pengguna kemudian diarahkan untuk langsung mencoba aplikasi dengan spesifikasi lingkungan yang

sama dengan yang telah diuraikan pada Tabel 5.1 Lingkungan Uji Coba.

Jumlah pengguna yang terlibat dalam pengujian perangkat sebanyak 10 orang. Dalam melakukan pengujian pengguna melakukan percobaan lebih dari satu kali dengan 2 teknologi yang berbeda yaitu pertama dengan menggunakan Oculus Rift dan yang kedua menggunakan Leap Motion Controller untuk masing-masing pengguna.

Dalam memberikan penilaian dan tanggapan, pengguna diberikan kuesioner pengujian aplikasi. Kuesioner pengujian ini dilakukan secara online melalui *website* google *form* dengan tautan https://intip.in/KUESIONERCLIMBING dan hasilnya akan ditampilkan pada tautan https://intip.in/TANGGAPANKUESIONERCLIMBING. Kuesioner pengujian ini memiliki beberapa aspek penilaian seputar desain antarmuka, *immersivity*, *controller* dan tingkat kenyamanan aplikasi. Nilai yang diberikan rentang nilai 1 hingga 5 dengan rincian pada Tabel 5.6 . pada bagian akhir terdapat saran untuk perbaikan fitur. Detil kuesioner pengguna dapat dilihat pada Tabel 5.7.

Tabel 5. 6 Rentan Nilai

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

Tabel 5. 7 Format Kuesioner

No	Parameter	STS	TS	N	S	SS
1	Aplikasi ini memiliki tampilan warna dan desain yang menarik	0	0	1	9	0
2	Aplikasi ini memiliki tata letak menu dan tampilan yang mudah dipahami	0	0	2	7	0
3	Saya merasakan sensasi layaknya dunia nyata	0	0	5	3	2
4	Saya merasa seperti sedang melakukan simulasi memanjat	0	1	3	4	2
5	Aplikasi ini dapat melatih kemampuan untuk berpikir dan bertindak cepat	0	0	4	4	2
6	Saya lebih mudah memainkan ini dengan Controller Oculus Rift	0	0	0	4	6
7	Saya lebih mudah memainkan ini dengan Leap Motion Controller	1	3	6	0	0
8	Aplikasi berjalan lancar tanpa adanya lag / crash	0	0	2	6	2
9	Saya merasa nyaman saat menggunakan aplikasi ini	0	0	3	6	1
10	Kontrol untuk pemain tidak membingungkan	0	0	2	1	7

5.3.2. Karakteristik Pengujian Pengguna

Pada subbab ini ditunjukkan pengguna yang bertindak sebagai penguji coba aplikasi yang dibangun. Tidak terdapat

kriteria atau keahlian khusus yang harus dimiliki pengguna karena aplikasi ini ditunjukkan kepada berbagai kalangan pengguna baik yang suka menggunakan aplikasi seperti ini ataupun tidak. Dalam pengujian aplikasi ini telah dilakukan kepada 10 partisipan dengan rentan usia 20-22 tahun, maka yang pertama didapatkan ialah karakteristik pengguna, yaitu sebagai berikut.

- Semua pennguna pernah memainkan permainan digital yang berbasis realitas virtual (10/10). Hal ini menjadikan pengguna lebih mudah dalam penggunaan aplikasi.
- Mayoritas (8/10) pengguna pernah memainkan permainan berbasis realitas virtual menggunakan Oculus Rift.
- Mayoritas (7/10) pengguna pernah memainkan permainan berbasis realitas virtual menggunakan Leap Motion Controller.

5.3.3. Hasil Pengujian Pengguna

Sistem penilaian didasarkan pada skala perhitungan satu sampai lima dimana skala satu menunjukkan nilai terendah dan skala lima menunjukkan skala tertinggi. Penilaian akhir kemudian dilakukan dengan menghitung berapa banyak penguji yang memilih suatu skala tertentu dan kemudian dicari nilai rata-ratanya. Hasil uji coba dipaparkan secara lengkap dengan disertai Tabel yang dapat dilihat pada Tabel 5.8 dan Tabel 5.9 .

Tabel 5. 8 Hasil Pengujian Pengguna

No	Pernyataan	Penilaian					Rata-Rata
		1	2	3	4	5	
1	Aplikasi ini memiliki tampilan warna dan desain yang menarik	0	0	1	9	0	3,9
2	Aplikasi ini memiliki tata letak menu dan	0	0	2	7	0	3,4

	tampilan yang mudah dipahami						
3	Saya merasakan sensasi layaknya dunia nyata	0	0	5	3	2	3,7
4	Saya merasa seperti sedang melakukan simulasi memanjat	0	1	3	4	2	3,7
5	Aplikasi ini dapat melatih kemampuan untuk berpikir dan bertindak cepat	0	0	4	4	2	3,8
6	Saya lebih mudah memainkan ini dengan Controller Oculus Rift	0	0	0	4	6	4,6
7	Saya lebih mudah memainkan ini dengan Leap Motion Controller	1	3	6	0	0	2,5
8	Aplikasi berjalan lancer tanpa adanya lag / crash	0	0	2	6	2	4
9	Saya merasa nyaman saat	0	0	3	6	1	3,8

	menggunakan aplikasi ini						
10	Kontrol untuk pemain tidak membingungkan	0	0	2	1	7	4,5

Tabel 5. 9 Hasil Akhir Pengujian Pengguna

No	Pernyataan	Rata-Rata	Total	Total (%)
Parameter Antarmuka				
1	Aplikasi ini memiliki tampilan warna dan desain yang menarik	3,9	3,7	73 %
2	Aplikasi ini memiliki tata letak menu dan tampilan yang mudah dipahami	3,4		
Parameter Immersivity				
3	Saya merasakan sensasi layaknya dunia nyata	3,7	3,7	74,7 %
4	Saya merasa seperti sedang melakukan simulasi memanjat	3,7		
5	Aplikasi ini dapat melatih kemampuan untuk berpikir dan bertindak cepat	3,8		
Parameter Controller				
6	Saya lebih mudah memainkan ini dengan Controller Oculus Rift	4,6	3,6	71 %
7	Saya lebih mudah memainkan ini dengan Leap Motion Controller	2,5		
Parameter Kenyamanan				
8	Aplikasi berjalan lancar tanpa adanya lag / crash	4	4,1	82 %

9	Saya merasa nyaman saat menggunakan aplikasi ini	3,8		
10	Kontrol untuk pemain tidak membingungkan	4,5		

5.3.4. Kritik dan Saran Pengguna

Dalam memberikan penilain dan tanggapan, pengguna diberikan kuesioner pengujian aplikasi. Kuesioner pengujian aplikasi ini terdapat bagian kritik dan saran untuk perbaikan fitur kedepannya. Kritik dan sara penggunan dapat dilihat pada Tabel 5.10 .

Tabel 5. 10 Kritik dan Saran Pengguna

No	Nama	Kritik dan Saran
1	Partisipan 1	Kurang berasa panjat tebing masih banyak bug. Perbaiki bug, kasih asset lebih banyak biar menarik
2	Partisipan 2	Pola panjat tebingnya itu-itu saja. Ditambah variasi
3	Partisipan 3	Variasi permainan kurang banyak, cenderung monoton serta ada beberapa kejadian lag hingga perlu close aplikasi. Menambahkan fitur untuk restart posisi pemain agar bisa mengulang dari awal disaat pemain terjatuh
4	Partisipan 4	Aplikasi keren, tampilan simpel dan nyaman dimainkan, mungkin bisa perbanyak fitur

5	Partisipan 5	Sudah bagus, tambahkan lebih bnyak objek
6	Partisipan 6	bisa diperbaiki lagi untuk pergerakan player
7	Partisipan 7	Tidak adanya target yang harus dicapai pada permainan, Tambahkan misi pada permainan agar pemain tidak bosan saat bermain
8	Partisipan 8	Tidak bisa memanjat karna jarak player terlalu jauh dari tebing, Dibuat bisa bergerak sebagai player agar bisa mengatur posisi yang pas untuk memanjat. setiap orang kan juga ber beda beda jarak jangkauan tangannya
9	Partisipan 9	Pergerakan yang diberikan oleh sistem sedikit kurang terasa nyata. Waktu untuk berpindah mungkin bisa ditambah agar pemain merasa seperti memiliki bobot, Konsumsi untuk tester:)
10	Partisipan 10	Controllernya kurang bersahabat, ditambahin info cara penggunaan

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 5.5. Dari data yang terdapat pada Tabel tersebut, diketahui

bahwa aplikasi yang dibuat telah berjalan sesuai dengan skenario yang diharapkan. Ada beberapa kelebihan dan kekurangan yang ada pada saat pengujian adapun hal-hal sebagai berikut.

5.4.1. Kelebihan

1. Pada Oculus Rift dan Oculus Touch sensor dan ruang lingkup untuk pergerakan tangan cukup luas.
2. Pada Oculus Rift dan Oculus Touch baik digunakan untuk aplikasi yang bisa bergerak leluasa seperti *climbing.*
3. Hasil *generate* tangan dari sensor Oculus adalah objek jadi lebih mudah untuk melakukan interaksi terhadap objek lain.
4. Leap Motion Controller adalah alat yang memiliki spesifikasi ringan dan mudah dijalankan pada perangkat computer dengan spesifikasi yang rendah.

5.4.2. Kekurangan

1. Pada Oculus Rift perangkat komputer untuk menjalankan aplikasi harus memiliki spesifikasi tinggi karena alat yang digunakan memiliki spesifikasi yang tinggi.
2. Pada Leap Motion Controller sensor untuk ruang lingkup tangan sangat terbatas.
3. Pada Leap Motion Controller bentuk tangan tidak membentuk sebuah objek dan tidak memiliki collider.
4. Pada Leap Motion Controller saat *generate* bentuk tangan kadang tidak sesuai dengan pergerakan tangan pemain.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan 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. Aplikasi dapat menampilkan tampilan rancangan visual simulasi *Wall Climbing*.
2. Aplikasi dapat mendeteksi sensor dari Oculus Touch dan Leap Motion.
3. Aplikasi berhasil melakukan implementasi simulasi *Wall Climbing* pada Oculus Rift - Oculus Touch dan Oculus Rift – Leap Motion.
4. Aplikasi berhasil dibuat dengan *Game Engine Unity.*
5. Pemain dapat merasakan sensasi melakukan panjat dinding.
6. Pemain dapat membandingkan keunggulan dari 2 alat yang digunakan untuk aplikasi.
7. Berdasarkan hasil uji coba fungsionalitas, aplikasi berhasil dibangun sesuai dengan rancangan.

6.2. Saran

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

1. Menambahkan beberapa jenis dinding yang berbeda untuk variasi pemain mencoba aplikasi.
2. Ditambahkan menu saat melakukan pemanjatan untuk reset posisi pemain atau keluar dari aplikasi.
3. Menambahkan pergerakan untuk pemain agar bisa mengatur posisi yang pas untuk memanjat.
4. Menambahkan info cara menggunakan aplikasi.

DAFTAR PUSTAKA

[1] Wikipedia, "Wall Climbing," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Climbing_wall. [Diakses 9 Desember 2018].

[2] Nesabamedia, "Pengertian VR," Virtual Reality, [Online]. Available: https://www.nesabamedia.com/pengertian-vr-virtual-reality. [Diakses 9 Desember 2018].

[3] Unity, "Unity," Unity, [Online]. Available: https://unity3d.com/unity. [Diakses 9 Desember 2018].

[4] Teo Filus, "C Sharp," codepolitan, 18 Januari 2017. [Online]. Available: https://www.codepolitan.com/pengenalan-bahasa-pemrograman-c-587effa1cb95b. [Diakses 9 Desember 2018].

[5] Adhitya Wibawa Putra, "Leap Motion," gadgetren, 1 Juni 2014 [Online]. Available: https://docs.unity3d.com/ScriptReference/ Windows.Speech.PhraseRecognizer.Start.html. [Diakses 18 Januari 2019].

[6] Oculus, "Oculus Rift," Oculus, [Online]. Available: https://www.oculus.com/rift/. [Diakses 18 Januari 2019].

[Halaman ini sengaja dikosongkan]

LAMPIRAN

A. Hasil kuesioner

Nama Lengkap

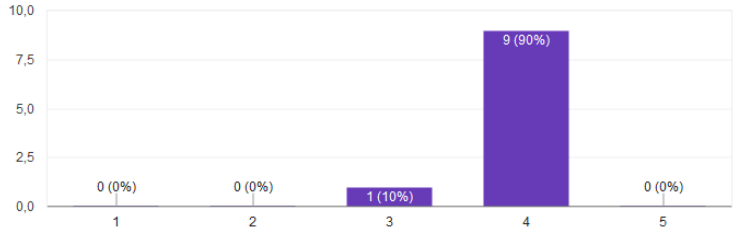
10 tanggapan

Narendra Haryo Bismo
YUGA MITRA HAKIKI
Surya Darma
Dias Adhi Pratama
Andrean Januar P
Nur Muhammad Husnul Habib Yahya
Subhan Maulana
Aditya Pratama
Rafi R. Ramadhan
Adi Darmawan

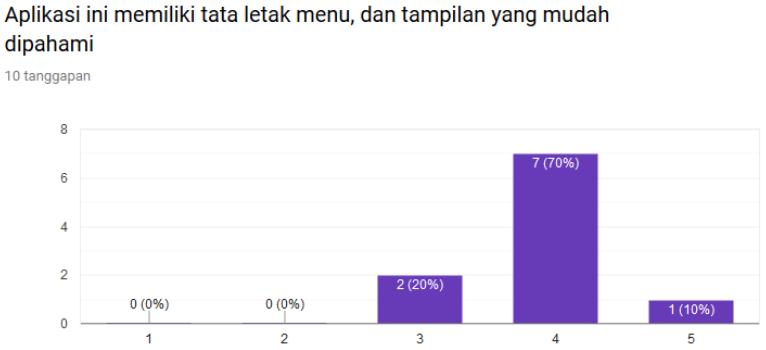
Gambar A. 1 Nama Penguji Aplikasi

Aplikasi ini memiliki tampilan warna, dan desain antarmuka yang menarik

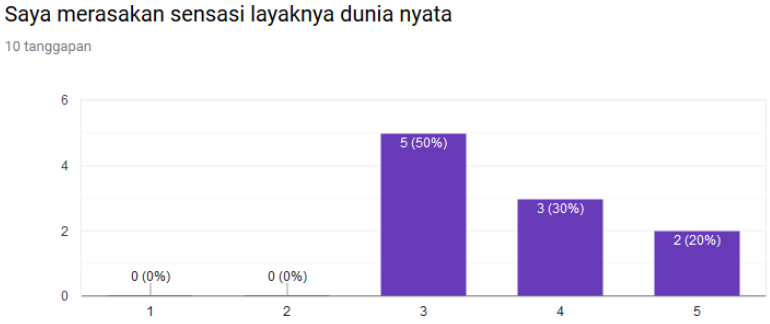
10 tanggapan



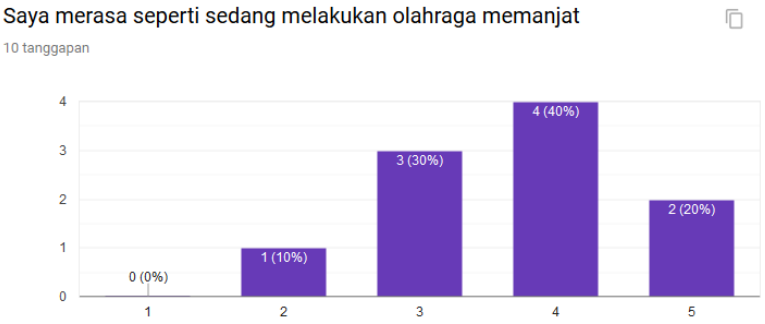
Gambar A. 2 Hasil Kuesioner Tentang Tampilan Antarmuka Aplikasi



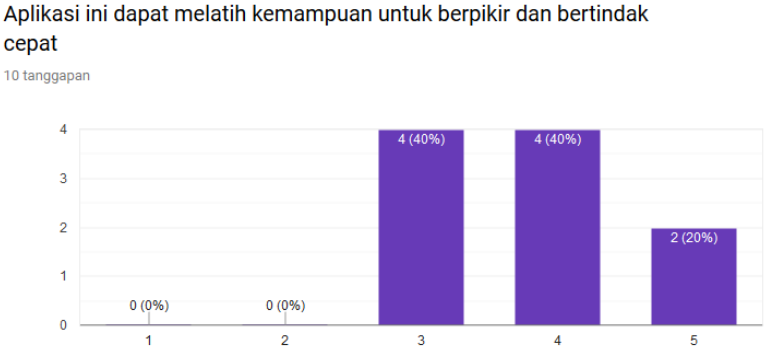
Gambar A. 3 Hasil Kuesioner Tentang Tata Letak Menu Aplikasi



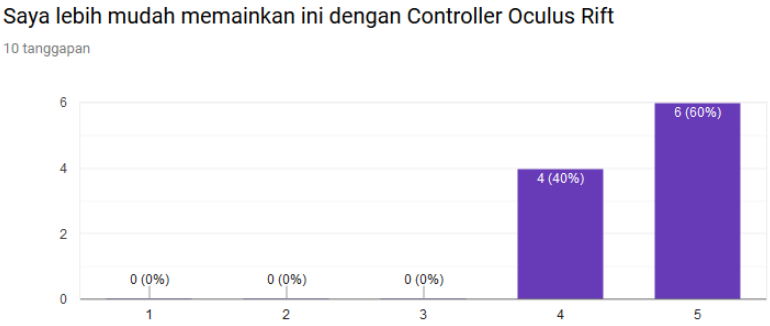
Gambar A. 4 Hasil Kuesioner Tentang Sensasi Pemain Pada Aplikasi



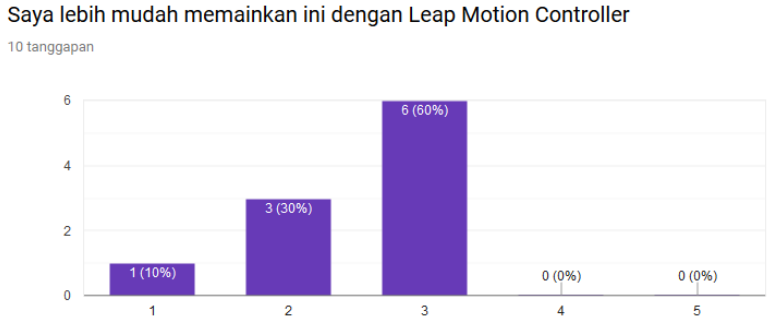
Gambar A. 5 Hasil Kuesioner Tentang Pengalaman Saat Mencoba Aplikasi



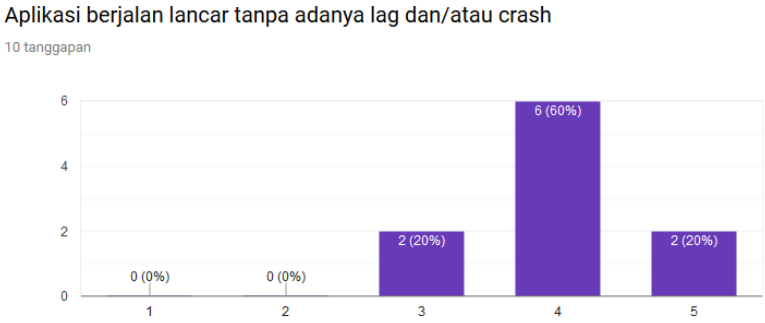
Gambar A. 6 Hasil Kuesioner Tentang Manfaat Saat Mencoba Aplikasi



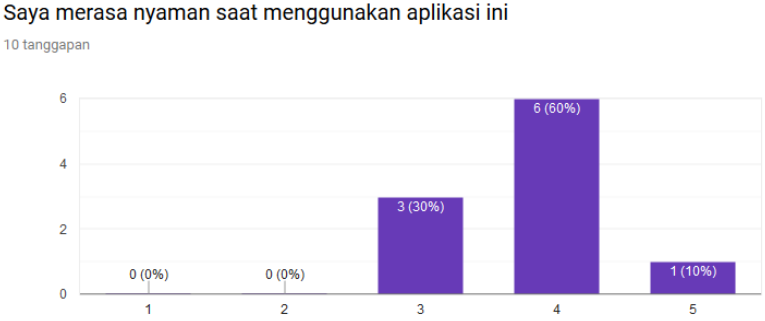
Gambar A. 7 Hasil Kuesioner Tentang Pengalaman Pemain Mencoba Dengan Oculus Rift dan Controller



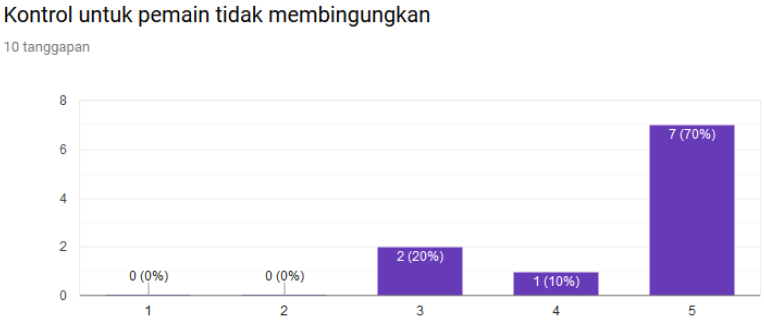
Gambar A. 8 Hasil Kuesioner Tentang Pengalaman Pemain Mencoba Dengan Leap Motion Controller



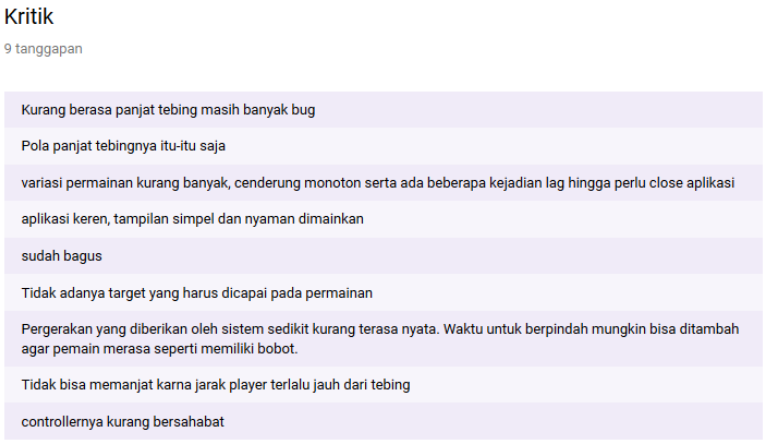
Gambar A. 9 Hasil Kuesioner Tentang Adanya Bug Pada Aplikasi



Gambar A. 10 Hasil Kuesioner Tentang Uji Kenyamanan Pemain Terhadap Aplikasi



Gambar A. 11 Hasil Kuesioner Tentang Kenyamanan Pemain Terhadap Kontrol Yang Digunakan



Gambar A. 12 Hasil Kuesioner Kritik Pemain

Saran

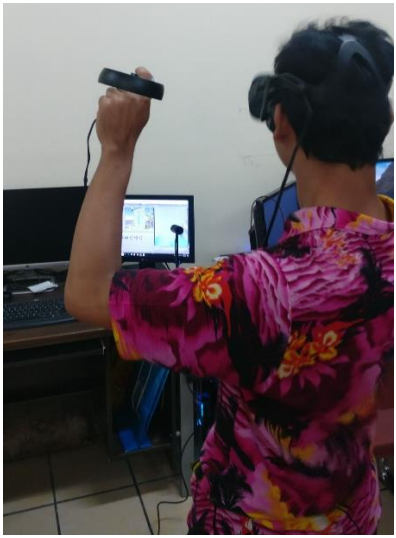
10 tanggapan

Perbaiki bug, kasih asset lebih banyak biar menarik
Ditambah variasi
menambahkan fitur untuk restart posisi pemain agar bisa mengulang dari awal disaat pemain terjatuh
mungkin bisa perbanyak fitur
tambahkan lebih bnyak objek
bisa diperbaiki lagi untuk pergerakan player
Tambahkan misi pada permainan agar pemain tidak bosan saat bermain
Konsumsi untuk tester:)
Dibuat bisa bergerak sebagai player agar bisa mengatur posisi yang pas untuk memanjat. setiap orang kan juga ber beda beda jarak jangkauan tangannya
ditambahin info cara penggunaan

Gambar A. 13 Hasil Kuesioner Saran Pemain

B. Foto Pengujian Oleh Penguji

Gambar B. 1 Foto Pengujian Oleh Penguji Keempat



Gambar B. 2 Foto Pengujian Oleh Penguji Ke 6

Gambar B. 3 Foto Pengujian Oleh Penguji Ketujuh

Gambar B. 4 Foto Pengujian Oleh Penguji Kedelapan

Gambar B. 5 Foto Pengujian Oleh Penguji Kesembilan

Gambar B. 6 Foto Pengujian Oleh Penguji Kesepuluh

C. Kode Sumber Lengkap

```
41. using System;
42. using UnityEngine;
43. 
44. public class OVRGrabbable : MonoBehaviour
45. {
46.     [SerializeField]
47.     protected bool m_allowOffhandGrab = true;
48.     [SerializeField]
49.     protected bool m_snapPosition = false;
50.     [SerializeField]
51.     protected bool m_snapOrientation = false;
52.     [SerializeField]
53.     protected Transform m_snapOffset;
54.     [SerializeField]
55.     protected Collider[] m_grabPoints = null;
56. 
57.     protected bool m_grabbedKinematic = false;
```

```
58.     protected Collider m_grabbedCollider = null;
59.     protected OVRGrabber m_grabbedBy = null;
60.
61.     /// If true, the object can currently be grab
    bed.
62.     public bool allowOffhandGrab
63.     {
64.         get { return m_allowOffhandGrab; }
65.     }
66.
67.     /// If true, the object is currently grabbed.
68.     public bool isGrabbed
69.     {
70.         get { return m_grabbedBy != null; }
71.     }
72.
73.     /// If true, the object's position will snap
    to match snapOffset when grabbed.
74.     public bool snapPosition
75.     {
76.         get { return m_snapPosition; }
77.     }
78.
79.     /// If true, the object's orientation will sn
    ap to match snapOffset when grabbed.
80.     public bool snapOrientation
81.     {
82.         get { return m_snapOrientation; }
83.     }
84.
85.     /// An offset relative to the OVRGrabber wher
    e this object can snap when grabbed.
86.     public Transform snapOffset
87.     {
88.         get { return m_snapOffset; }
89.     }
90.
91.     /// Returns the OVRGrabber currently grabbing
     this object.
92.     public OVRGrabber grabbedBy
```

```
93.     {
94.         get { return m_grabbedBy; }
95.     }
96.
97.     /// The transform at which this object was gr
    abbed.
98.     public Transform grabbedTransform
99.     {
100.        get { return m_grabbedCollider.transform;
     }
101.    }
102.
103.    /// The Rigidbody of the collider that was us
    ed to grab this object.
104.    public Rigidbody grabbedRigidbody
105.    {
106.        get { return m_grabbedCollider.attachedRi
    gidbody; }
107.    }
108.
109.    /// The contact point(s) where the object was
     grabbed.
110.    public Collider[] grabPoints
111.    {
112.        get { return m_grabPoints; }
113.    }
114.
115.    /// Notifies the object that it has been grab
    bed.
116.    virtual public void GrabBegin(OVRGrabber hand
    , Collider grabPoint)
117.    {
118.        m_grabbedBy = hand;
119.        m_grabbedCollider = grabPoint;
120.        gameObject.GetComponent<Rigidbody>().isKi
    nematic = true;
121.    }
122.
123.    /// Notifies the object that it has been rele
    ased.
124.    virtual public void GrabEnd(Vector3 linearVel
    ocity, Vector3 angularVelocity)
```

```
125.     {
126.         Rigidbody rb = gameObject.GetComponent<Ri
    gidbody>();
127.         rb.isKinematic = m_grabbedKinematic;
128.         rb.velocity = linearVelocity;
129.         rb.angularVelocity = angularVelocity;
130.         m_grabbedBy = null;
131.         m_grabbedCollider = null;
132.     }
133.       void Awake()
134.     {
135.         if (m_grabPoints.Length == 0)
136.         {
137.             // Get the collider from the grabbabl
    e
138.             Collider collider = this.GetComponent
    <Collider>();
139.             if (collider == null)
140.             {
141.                 throw new ArgumentException("Grab
    bables cannot have zero grab points and no collid
    er -- please add a grab point or collider.");
142.             }
143.
144.             // Create a default grab point
145.             m_grabPoints = new Collider[1] { coll
    ider };
146.         }
147.     }
148.       protected virtual void Start()
149.     {
150.         m_grabbedKinematic = GetComponent<Rigidbo
    dy>().isKinematic;
151.     }
152.
153.     void OnDestroy()
154.     {
155.         if (m_grabbedBy != null)
156.         {
157.             // Notify the hand to release destroy
    ed grabbables
158.             m_grabbedBy.ForceRelease(this);
```

```
        }
    }
}
```

Kode Sumber C. 1 Fungsi Benda Agar Bisa Di Grab Pada OculusRift

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

/// Allows grabbing and throwing of objects with the OVRGrabbable component on them.
[RequireComponent(typeof(Rigidbody))]
public class OVRGrabber : MonoBehaviour
{
    // Grip trigger thresholds for picking up objects, with some hysteresis.
    public float grabBegin = 0.55f;
    public float grabEnd = 0.35f;

    protected bool m_parentHeldObject = false;

    protected Transform m_gripTransform = null;

    protected Collider[] m_grabVolumes = null;

    protected OVRInput.Controller m_controller;

    protected Transform m_parentTransform;

    protected bool m_grabVolumeEnabled = true;
    protected Vector3 m_lastPos;
    protected Quaternion m_lastRot;
    protected Quaternion m_anchorOffsetRotation;

    protected Vector3 m_anchorOffsetPosition;
    protected float m_prevFlex;
    protected OVRGrabbable m_grabbedObj = null;
    protected Vector3 m_grabbedObjectPosOff;
    protected Quaternion m_grabbedObjectRotOff;
    protected Dictionary<OVRGrabbable, int> m_grabCandidates = new Dictionary<OVRGrabbable, int>();
```

```
110.     protected bool operatingWithoutOVRCameraRig =
     true;
111.
112.     public bool grabed;
113.
114.     /// The currently grabbed object.
115.     public OVRGrabbable grabbedObject
116.     {
117.         get { return m_grabbedObj; }
118.     }
119.
120.         public void ForceRelease(OVRGrabbable gra
    bbable)
121.     {
122.         bool canRelease = (
123.             (m_grabbedObj != null) &&
124.             (m_grabbedObj == grabbable)
125.             );
126.         if (canRelease)
127.         {
128.             GrabEnd();
129.         }
130.     }
131.
132.     protected virtual void Awake()
133.     {
134.         m_anchorOffsetPosition = transform.localP
    osition;
135.         m_anchorOffsetRotation = transform.localR
    otation;
136.
137.         // If we are being used with an OVRCamera
    Rig, let it drive input updates, which may come f
    rom Update or FixedUpdate.
138.
139.         OVRCameraRig rig = null;
140.         if (transform.parent != null && transform
    .parent.parent != null)
141.             rig = transform.parent.parent.GetComp
    onent<OVRCameraRig>();
142.
143.         if (rig != null)
```

```
144.         {
145.             rig.UpdatedAnchors += (r) = > {OnUpda
    tedAnchors(); };
146.             operatingWithoutOVRCameraRig = false;

147.         }
148.     }
149.
150.     protected virtual void Start()
151.     {
152.         m_lastPos = transform.position;
153.         m_lastRot = transform.rotation;
154.         if (m_parentTransform == null)
155.         {
156.             if (gameObject.transform.parent != nu
    ll)
157.             {
158.                 m_parentTransform = gameObject.tr
    ansform.parent.transform;
159.             }
160.             else
161.             {
162.                 m_parentTransform = new GameObjec
    t().transform;
163.                 m_parentTransform.position = Vect
    or3.zero;
164.                 m_parentTransform.rotation = Quat
    ernion.identity;
165.             }
166.         }
167.     }
168.
169.     void FixedUpdate()
170.     {
171.         if (operatingWithoutOVRCameraRig)
172.             OnUpdatedAnchors();
173.     }
174.
175.     // Hands follow the touch anchors by calling
    MovePosition each frame to reach the anchor.
176.     void OnUpdatedAnchors()
177.     {
```

```
178.         Vector3 handPos = OVRInput.GetLocalControllerPosition(m_controller);
179.         Quaternion handRot = OVRInput.GetLocalControllerRotation(m_controller);
180.         Vector3 destPos = m_parentTransform.TransformPoint(m_anchorOffsetPosition + handPos);
181.         Quaternion destRot = m_parentTransform.rotation * handRot * m_anchorOffsetRotation;
182.         GetComponent<Rigidbody>().MovePosition(destPos);
183.         GetComponent<Rigidbody>().MoveRotation(destRot);
184.
185.         if (!m_parentHeldObject)
186.         {
187.             MoveGrabbedObject(destPos, destRot);
188.         }
189.         m_lastPos = transform.position;
190.         m_lastRot = transform.rotation;
191.
192.         float prevFlex = m_prevFlex;
193.         // Update values from inputs
194.         m_prevFlex = OVRInput.Get(OVRInput.Axis1D.PrimaryHandTrigger, m_controller);
195.
196.         CheckForGrabOrRelease(prevFlex);
197.     }
198.
199.     void OnDestroy()
200.     {
201.         if (m_grabbedObj != null)
202.         {
203.             GrabEnd();
204.         }
205.     }
206.
207.     void OnTriggerEnter(Collider otherCollider)
208.     {
209.         // Get the grab trigger
```

```
210.         OVRGrabbable grabbable = otherCollider.Ge
    tComponent<OVRGrabbable>() ? ? otherCollider.GetC
    omponentInParent<OVRGrabbable>();
211.         if (grabbable == null) return;
212.
213.         // Add the grabbable
214.         int refCount = 0;
215.         m_grabCandidates.TryGetValue(grabbable, o
    ut refCount);
216.         m_grabCandidates[grabbable] = refCount +
    1;
217.     }
218.
219.     void OnTriggerExit(Collider otherCollider)
220.     {
221.         OVRGrabbable grabbable = otherCollider.Ge
    tComponent<OVRGrabbable>() ? ? otherCollider.GetC
    omponentInParent<OVRGrabbable>();
222.         if (grabbable == null) return;
223.
224.         // Remove the grabbable
225.         int refCount = 0;
226.         bool found = m_grabCandidates.TryGetValue
    (grabbable, out refCount);
227.         if (!found)
228.         {
229.             return;
230.         }
231.
232.         if (refCount > 1)
233.         {
234.             m_grabCandidates[grabbable] = refCoun
    t - 1;
235.         }
236.         else
237.         {
238.             m_grabCandidates.Remove(grabbable);
239.         }
240.     }
241.
242.     protected void CheckForGrabOrRelease(float pr
    evFlex)
```

```
243.     {
244.         if ((m_prevFlex >= grabBegin) && (prevFle
    x < grabBegin))
245.         {
246.             GrabBegin();
247.             grabed = true;
248.         }
249.         else if ((m_prevFlex <= grabEnd) && (prev
    Flex > grabEnd))
250.         {
251.             GrabEnd();
252.             grabed = false;
253.         }
254.     }
255.
256.     protected virtual void GrabBegin()
257.     {
258.         float closestMagSq = float.MaxValue;
259.         OVRGrabbable closestGrabbable = null;
260.         Collider closestGrabbableCollider = null;

261.
262.         // Iterate grab candidates and find the c
    losest grabbable candidate
263.         foreach(OVRGrabbable grabbable in m_grabC
    andidates.Keys)
264.         {
265.             bool canGrab = !(grabbable.isGrabbed
    && !grabbable.allowOffhandGrab);
266.             if (!canGrab)
267.             {
268.                 continue;
269.             }
270.
271.             for (int j = 0; j < grabbable.grabPoi
    nts.Length; ++j)
272.             {
273.                 Collider grabbableCollider = grab
    bable.grabPoints[j];
274.                 // Store the closest grabbable
```

```
275.                 Vector3 closestPointOnBounds = gr
    abbableCollider.ClosestPointOnBounds(m_gripTransf
    orm.position);
276.                 float grabbableMagSq = (m_gripTra
    nsform.position - closestPointOnBounds).sqrMagnit
    ude;
277.                 if (grabbableMagSq < closestMagSq
    )
278.                 {
279.                     closestMagSq = grabbableMagSq
    ;
280.                     closestGrabbable = grabbable;

281.                     closestGrabbableCollider = gr
    abbableCollider;
282.                 }
283.             }
284.         }
285.
286.         // Disable grab volumes to prevent overla
    ps
287.         GrabVolumeEnable(false);
288.
289.         if (closestGrabbable != null)
290.         {
291.             if (closestGrabbable.isGrabbed)
292.             {
293.                 closestGrabbable.grabbedBy.Offhan
    dGrabbed(closestGrabbable);
294.             }
295.
296.             m_grabbedObj = closestGrabbable;
297.             m_grabbedObj.GrabBegin(this, closestG
    rabbableCollider);
298.
299.             m_lastPos = transform.position;
300.             m_lastRot = transform.rotation;
301.
302.             // Set up offsets for grabbed object
    desired position relative to hand.
303.             if (m_grabbedObj.snapPosition)
304.             {
```

```
305.                 m_grabbedObjectPosOff = m_gripTra
    nsform.localPosition;
306.                 if (m_grabbedObj.snapOffset)
307.                 {
308.                     Vector3 snapOffset = m_grabbe
    dObj.snapOffset.position;
309.                     if (m_controller == OVRInput.
    Controller.LTouch) snapOffset.x = -
    snapOffset.x;
310.                     m_grabbedObjectPosOff += snap
    Offset;
311.                 }
312.             }
313.             else
314.             {
315.                 Vector3 relPos = m_grabbedObj.tra
    nsform.position - transform.position;
316.                 relPos = Quaternion.Inverse(trans
    form.rotation) * relPos;
317.                 m_grabbedObjectPosOff = relPos;
318.             }
319.
320.             if (m_grabbedObj.snapOrientation)
321.             {
322.                 m_grabbedObjectRotOff = m_gripTra
    nsform.localRotation;
323.                 if (m_grabbedObj.snapOffset)
324.                 {
325.                     m_grabbedObjectRotOff = m_gra
    bbedObj.snapOffset.rotation * m_grabbedObjectRotO
    ff;
326.                 }
327.             }
328.             else
329.             {
330.                 Quaternion relOri = Quaternion.In
    verse(transform.rotation) * m_grabbedObj.transfor
    m.rotation;
331.                 m_grabbedObjectRotOff = relOri;
332.             }
333.
```

```
334.             MoveGrabbedObject(m_lastPos, m_lastRo
    t, true);
335.             if (m_parentHeldObject)
336.             {
337.                 m_grabbedObj.transform.parent = t
    ransform;
338.             }
339.         }
340.     }
341.
342.     protected virtual void MoveGrabbedObject(Vect
    or3 pos, Quaternion rot, bool forceTeleport = fal
    se)
343.     {
344.         if (m_grabbedObj == null)
345.         {
346.             return;
347.         }
348.
349.         Rigidbody grabbedRigidbody = m_grabbedObj
    .grabbedRigidbody;
350.         Vector3 grabbablePosition = pos + rot * m
    _grabbedObjectPosOff;
351.         Quaternion grabbableRotation = rot * m_gr
    abbedObjectRotOff;
352.
353.         if (forceTeleport)
354.         {
355.             grabbedRigidbody.transform.position =
     grabbablePosition;
356.             grabbedRigidbody.transform.rotation =
     grabbableRotation;
357.         }
358.         else
359.         {
360.             grabbedRigidbody.MovePosition(grabbab
    lePosition);
361.             grabbedRigidbody.MoveRotation(grabbab
    leRotation);
362.         }
363.     }
364.
```

```
365.     protected void GrabEnd()
366.     {
367.         if (m_grabbedObj != null)
368.         {
369.             OVRPose localPose = new OVRPose{ posi
    tion = OVRInput.GetLocalControllerPosition(m_cont
    roller), orientation = OVRInput.GetLocalControlle
    rRotation(m_controller) };
370.             OVRPose offsetPose = new OVRPose{ pos
    ition = m_anchorOffsetPosition, orientation = m_a
    nchorOffsetRotation };
371.             localPose = localPose * offsetPose;
372.
373.             OVRPose trackingSpace = transform.ToO
    VRPose() * localPose.Inverse();
374.             Vector3 linearVelocity = trackingSpac
    e.orientation * OVRInput.GetLocalControllerVeloci
    ty(m_controller);
375.             Vector3 angularVelocity = trackingSpa
    ce.orientation * OVRInput.GetLocalControllerAngul
    arVelocity(m_controller);
376.
377.             GrabbableRelease(linearVelocity, angu
    larVelocity);
378.         }
379.
380.         // Re-
    enable grab volumes to allow overlap events
381.         GrabVolumeEnable(true);
382.     }
383.
384.     protected void GrabbableRelease(Vector3 linea
    rVelocity, Vector3 angularVelocity)
385.     {
386.         m_grabbedObj.GrabEnd(linearVelocity, angu
    larVelocity);
387.         if (m_parentHeldObject) m_grabbedObj.tran
    sform.parent = null;
388.         m_grabbedObj = null;
389.     }
390.
```

```
391.     protected virtual void GrabVolumeEnable(bool
    enabled)
392.     {
393.         if (m_grabVolumeEnabled == enabled)
394.         {
395.             return;
396.         }
397.
398.         m_grabVolumeEnabled = enabled;
399.         for (int i = 0; i < m_grabVolumes.Length;
     ++i)
400.         {
401.             Collider grabVolume = m_grabVolumes[i
    ];
402.             grabVolume.enabled = m_grabVolumeEnab
    led;
403.         }
404.
405.         if (!m_grabVolumeEnabled)
406.         {
407.             m_grabCandidates.Clear();
408.         }
409.     }
410.
411.     protected virtual void OffhandGrabbed(OVRGrab
    bable grabbable)
412.     {
413.         if (m_grabbedObj == grabbable)
414.         {
415.             GrabbableRelease(Vector3.zero, Vector
    3.zero);
416.         }
417.     }
418. }
```

Kode Sumber C. 2 Fungsi Untuk Memegang Benda Pada Oculus Rift

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.
5.
```

```
6.  public class pull : MonoBehaviour
7.  {
8.      public OVRGrabber handGrab;
9.      public Rigidbody rbBody;
10.     public GameObject Body;
11.     private Rigidbody palmRB;
12.     private Vector3 boPos;
13.     public pullkiri tangKiri;
14.
15.     public changecolor warna;
16.
17.     public Vector3 prevPos;
18.     public bool canGrip;
19.     public bool feedback = false;
20.     // Start is called before the first frame upd
    ate
21.     void Start()
22.     {
23.         prevPos = gameObject.transform.localPosit
    ion;
24.         palmRB = gameObject.GetComponent<Rigidbod
    y>();
25.     }
26.
27.     // Update is called once per frame
28.     void Update()
29.     {
30.         if (feedback)
31.         {
32.             warna.Changecolor();
33.         }
34.
35.
36.         if (canGrip && handGrab.grabed == true)
37.         {
38.             feedback = true;
39.             canGrip = true;
40.             rbBody.useGravity = false;
41.             rbBody.isKinematic = true;
42.             Body.transform.position += (prevPos -
     gameObject.transform.localPosition);
43.         }
```

```
44.         else
45.         {
46.             rbBody.useGravity = true;
47.             rbBody.isKinematic = false;

48.         }
49.         prevPos = gameObject.transform.localPosit
    ion;
50.     }
51.     private void OnTriggerStay(Collider other)
52.     {
53.         if(other.gameObject.layer == 15)
54.         {
55.             Debug.Log(other);
56.             warna = other.gameObject.GetComponent
    <changecolor>();
57.             warna.batu = other.gameObject;
58.             canGrip = true;
59.             palmRB.constraints = RigidbodyConstra
    ints.FreezeAll;
60.         }
61.     }
62.     private void OnTriggerExit(Collider other)
63.     {
64.         canGrip = false;
65.         palmRB.constraints = RigidbodyConstraints
    .None;
66.         if (feedback)
67.         {
68.             warna.Changecolor1();
69.             feedback = false;
70.         }
71.
72.     }
73. }
```

Kode Sumber C. 3 Fungsi Untuk Mengubah Benda Sebagai Tumpuan dan Memberikan Feedback Saat Grab Benda

```
55. private HashSet<InteractionController> _graspingC
    ontrollers = new HashSet<InteractionController>()
    ;
56.
57. private bool _wasKinematicBeforeGrasp;
58. private bool _justGrasped = false;
59.
60. private float _dragBeforeGrasp = 0F;
61. private float _angularDragBeforeGrasp = 0.05F;
62.
63. private IGraspedPoseHandler _graspedPoseHandler;

64.
65. public IGraspedPoseHandler graspedPoseHandler{
66.   get {
67.     if (_graspedPoseHandler == null) {
68.       _graspedPoseHandler = new KabschGraspedPose
    (this);
69.     }
70.     return _graspedPoseHandler;
71.   }
72.   set {
73.     _graspedPoseHandler = value;
74.   }
75. }
76.
77. private KinematicGraspedMovement _lazyKinematicGr
    aspedMovement;
78. private KinematicGraspedMovement _kinematicGraspe
    dMovement{
79.   get {
80.     if (_lazyKinematicGraspedMovement == null) {

81.       _lazyKinematicGraspedMovement = new Kinemat
    icGraspedMovement();
82.     }
83.     return _lazyKinematicGraspedMovement;
84.   }
85. }
86.
87. private NonKinematicGraspedMovement _lazyNonKinem
    aticGraspedMovement;
```

```
88. private NonKinematicGraspedMovement _nonKinematic
    GraspedMovement{
89.   get {
90.     if (_lazyNonKinematicGraspedMovement == null)
     {
91.       _lazyNonKinematicGraspedMovement = new NonK
    inematicGraspedMovement();
92.     }
93.     return _lazyNonKinematicGraspedMovement;
94.   }
95. }
96.
97. private IThrowHandler _throwHandler;
98.
99. public IThrowHandler throwHandler{
100.   get {
101.     if (_throwHandler == null) {
102.       _throwHandler = new SlidingWindowThrow();
103.     }
104.     return _throwHandler;
105.   }
106.   set {
107.     _throwHandler = value;
108.   }
109. }
110.
111. public void BeginGrasp(List<InteractionController
    > controllers) {
112.       _justGrasped = true;
113.
114.       // End suspension by ending the grasp on th
    e suspending hand,
115.       // calling EndGrasp immediately.
116.       if (isSuspended) {
117.           _suspendingController.ReleaseGrasp();
118.       }
119.
120.       // If multi-
    grasp is not allowed, release the old grasp.
121.       if (!allowMultiGrasp && isGrasped) {
122.           _graspingControllers.Query().First().Re
    leaseGrasp();
```

```
123.        }
124.
125.        // Add each newly grasping hand to internal
     reference and pose solver.
126.        foreach(var controller in controllers) {
127.            _graspingControllers.Add(controller);
128.
129.            if (moveObjectWhenGrasped) {
130.                graspedPoseHandler.AddController(co
    ntroller);
131.            }
132.
133.            // Fire interaction callback.
134.            OnPerControllerGraspBegin(controller);

135.        }
136.
137.        // If object wasn't grasped before, store r
    igidbody settings and
138.        // fire object interaction callback.
139.        if (_graspingControllers.Count == controlle
    rs.Count) {
140.
141.            // Remember drag settings pre-
    grasp, to be restored on release.
142.            _dragBeforeGrasp = rigidbody.drag;
143.            _angularDragBeforeGrasp = rigidbody.ang
    ularDrag;
144.
145.            // Remember kinematic state.
146.            _wasKinematicBeforeGrasp = rigidbody.is
    Kinematic;
147.            switch (graspedMovementType) {
148.            case GraspedMovementType.Inherit: break
    ; // no change
149.            case GraspedMovementType.Kinematic:
150.                rigidbody.isKinematic = true; break
    ;
151.            case GraspedMovementType.Nonkinematic:

152.                rigidbody.isKinematic = false; brea
    k;
```

```
153.            }
154.
155.            // Set rigidbody drag/angular drag to z
    ero.
156.            rigidbody.drag = 0F;
157.            rigidbody.angularDrag = 0F;
158.
159.            OnGraspBegin();
160.        }
161.    }
162.
163.    public void EndGrasp(List<InteractionController
    > controllers) {
164.        if (_graspingControllers.Count == controlle
    rs.Count && isSuspended) {
165.            // No grasped hands: Should not be susp
    ended any more;
166.            // having been suspended also means we
    were only grasped by one hand
167.            EndSuspension(controllers[0]);
168.        }
169.
170.        foreach(var controller in controllers) {
171.            _graspingControllers.Remove(controller)
    ;
172.
173.            // Fire interaction callback.
174.            OnPerControllerGraspEnd(controller);
175.
176.            if (moveObjectWhenGrasped) {
177.                // Remove each hand from the pose s
    olver.
178.                graspedPoseHandler.RemoveController
    (controller);
179.            }
180.        }
181.
182.        // If the object is no longer grasped by an
    y hands, restore state and
183.        // activate throw handler.
184.        if (_graspingControllers.Count == 0) {
```

```
185.            // Restore drag settings from prior to
    the grasp.
186.            rigidbody.drag = _dragBeforeGrasp;
187.            rigidbody.angularDrag = _angularDragBef
    oreGrasp;
188.
189.            // Revert kinematic state.
190.            rigidbody.isKinematic = _wasKinematicBe
    foreGrasp;
191.
192.            if (controllers.Count == 1) {
193.                throwHandler.OnThrow(this, controll
    ers.Query().First());
194.            }
195.
196.            OnGraspEnd();
197.
198.            if (_justGrasped) _justGrasped = false;

199.        }
200.    }
201.
202.    public void StayGrasped(List<InteractionControl
    ler> controllers) {
203.        if (moveObjectWhenGrasped) {
204.            Vector3    origPosition = rigidbody.pos
    ition;
205.            Quaternion origRotation = rigidbody.rot
    ation;
206.            Vector3    newPosition;
207.            Quaternion newRotation;
208.
209.            graspedPoseHandler.GetGraspedPosition(o
    ut newPosition, out newRotation);
210.
211.            fixedUpdateGraspedMovement(new Pose(ori
    gPosition, origRotation),
212.                new Pose(newPosition, newRotation),

213.                controllers);
214.
```

```
215.            throwHandler.OnHold(this, controllers);

216.        }
217.
218.        OnGraspStay();
219.
220.        _justGrasped = false;
221.    }
222.
223.    protected virtual void fixedUpdateGraspedMoveme
    nt(Pose origPose, Pose newPose,
224.        List<InteractionController> controllers) {

225.        IGraspedMovementHandler graspedMovementHand
    ler
226.            = rigidbody.isKinematic ?
227.            (IGraspedMovementHandler)_kinematicGras
    pedMovement
228.            : (IGraspedMovementHandler)_nonKinemati
    cGraspedMovement;
229.        graspedMovementHandler.MoveTo(newPose.posit
    ion, newPose.rotation,
230.            this, _justGrasped);
231.
232.        OnGraspedMovement(origPose.position, origPo
    se.rotation,
233.            newPose.position, newPose.rotation,
234.            controllers);
235.    }
236.
237.    protected InteractionController _suspendingCont
    roller = null;
238.
239.    public void BeginSuspension(InteractionControll
    er controller) {
240.        _suspendingController = controller;
241.
242.        OnSuspensionBegin(controller);
243.    }
244.
245.    public void EndSuspension(InteractionController
     controller) {
```

```
246.        _suspendingController = null;
247.
248.        OnSuspensionEnd(controller);
249.    }
250.
251.        #region Forces
252.
253.        private bool _appliedForces = false;
254.    protected Vector3 _accumulatedLinearAcceleratio
    n = Vector3.zero;
255.    protected Vector3 _accumulatedAngularAccelerati
    on = Vector3.zero;
256.
257.    public void FixedUpdateForces() {
258.        if (!isGrasped) {
259.            //Only apply if non-
    zero to prevent waking up the body
260.            if (_accumulatedLinearAcceleration != V
    ector3.zero) {
261.                rigidbody.velocity += _accumulatedL
    inearAcceleration * Time.fixedDeltaTime;
262.            }
263.
264.            if (_accumulatedAngularAcceleration !=
    Vector3.zero) {
265.                rigidbody.angularVelocity += _accum
    ulatedAngularAcceleration * Time.fixedDeltaTime;

266.            }
267.
268.            //Reset so we can accumulate for the ne
    xt frame
269.            _accumulatedLinearAcceleration = Vector
    3.zero;
270.            _accumulatedAngularAcceleration = Vecto
    r3.zero;
271.
272.            _appliedForces = false;
273.        }
274.    }
275.     #endregion
276.         #region Colliders
```

```
277.
278.        protected List<Collider> _interactionCollid
    ers = new List<Collider>();
279.
280.    public void RefreshInteractionColliders() {
281.        Utils.FindColliders<Collider>(this.gameObje
    ct, _interactionColliders,
282.            includeInactiveObjects: false);
283.
284.        _interactionColliders.RemoveAll(
285.            c = > c.GetComponent<IgnoreColliderForI
    nteraction>() != null);
286.
287.        refreshInteractionColliderLayers();
288.    }
289.
290.    #endregion
```

Kode Sumber C. 4 Fungsi Agar Benda Bisa Di Grab Pada Leap Motion

```
52. using Leap.Unity.Attributes;
53. using System;
54. using System.Collections;
55. using System.Collections.Generic;
56.
57. #if UNITY_EDITOR
58. using UnityEditor;
59. #endif
60. using UnityEngine;
61.
62. namespace Leap.Unity.Attachments{
63.
64.     public class AttachmentHands : MonoBehaviour
    {
65.
66.        private AttachmentPointFlags _attachmentPoi
    nts = AttachmentPointFlags.Palm | AttachmentPoint
    Flags.Wrist;
67.        public AttachmentPointFlags attachmentPoint
    s {
68.          get {
69.            return _attachmentPoints;
```

```
70.         }
71.         set {
72.           if (_attachmentPoints != value) {
73.             #if UNITY_EDITOR
74.             Undo.IncrementCurrentGroup();
75.             Undo.SetCurrentGroupName("Modify Atta
    chment Points");
76.
77.             Undo.RecordObject(this, "Modify Attac
    hmentHands Points");
78.             #endif
79.
80.             _attachmentPoints = value;
81.             refreshAttachmentHandTransforms();
82.           }
83.         }
84.       }
85.
86.       private Func<Hand>[] _handAccessors;
87.
88.       public Func<Hand>[] handAccessors { get { r
    eturn _handAccessors; } set { _handAccessors = va
    lue; } }
89.
90.       private AttachmentHand[] _attachmentHands;

91.
92.       public AttachmentHand[] attachmentHands { g
    et { return _attachmentHands; } set { _attachment
    Hands = value; } }
93.
94.   #if UNITY_EDITOR
95.       void OnValidate() {
96.         if (getIsPrefab()) return;
97.
98.         reinitialize();
99.       }
100.  #endif
101.
102.      void Awake() {
103.  #if UNITY_EDITOR
104.        if (getIsPrefab()) return;
```

```
105.  #endif
106.
107.        reinitialize();
108.      }
109.
110.      private void reinitialize() {
111.        refreshHandAccessors();
112.        refreshAttachmentHands();
113.
114.        #if UNITY_EDITOR
115.        EditorApplication.delayCall += refreshAtt
    achmentHandTransforms;
116.        #else
117.        refreshAttachmentHandTransforms();
118.        #endif
119.      }
120.
121.      void Update() {
122.        #if UNITY_EDITOR
123.        PrefabType prefabType = PrefabUtility.Get
    PrefabType(this.gameObject);
124.        if (prefabType == PrefabType.Prefab || pr
    efabType == PrefabType.ModelPrefab) {
125.          return;
126.        }
127.        #endif
128.
129.        bool requiresReinitialization = false;
130.
131.        using (new ProfilerSample("Attachment Han
    ds Update", this.gameObject)) {
132.          for (int i = 0; i < _attachmentHands.Le
    ngth; i++) {
133.            var attachmentHand = attachmentHands[
    i];
134.
135.            if (attachmentHand == null) {
136.              requiresReinitialization = true;
137.              break;
138.            }
139.
140.            var leapHand = handAccessors[i]();
```

```
141.             attachmentHand.isTracked = leapHand !
    = null;
142.
143.             using (new ProfilerSample(attachmentH
    and.gameObject.name + " Update Points")) {
144.               foreach(var point in attachmentHand
    .points) {
145.                 point.SetTransformUsingHand(leapH
    and);
146.               }
147.             }
148.           }
149.
150.           if (requiresReinitialization) {
151.             reinitialize();
152.           }
153.         }
154.       }
155.
156.       private void refreshHandAccessors() {
157.           // If necessary, generate a left-
    hand and right-hand set of accessors.
158.           if (_handAccessors == null || _handAcce
    ssors.Length == 0) {
159.             _handAccessors = new Func<Hand>[2];
160.
161.             _handAccessors[0] = new Func<Hand>(()
     = > { return Hands.Left; });
162.             _handAccessors[1] = new Func<Hand>(()
     = > { return Hands.Right; });
163.           }
164.         }
165.
166.         private void refreshAttachmentHands() {
167.             // If we're a prefab, we'll be unable
     to set parent transforms, so we shouldn't create
     new objects in general.
168.             bool isPrefab = false;
169.             #if UNITY_EDITOR
170.             isPrefab = getIsPrefab();
171.             #endif
172.
```

```
173.             // If necessary, generate a left and
    right AttachmentHand.
174.             if (_attachmentHands == null || _atta
    chmentHands.Length == 0 || (_attachmentHands[0] =
    = null || _attachmentHands[1] == null)) {
175.                 _attachmentHands = new AttachmentHa
    nd[2];
176.
177.                 // Try to use existing AttachmentHa
    nd objects first.
178.                 foreach(Transform child in this.tra
    nsform.GetChildren()) {
179.                     var attachmentHand = child.GetCom
    ponent<AttachmentHand>();
180.                     if (attachmentHand != null) {
181.                         _attachmentHands[attachmentHand
    .chirality == Chirality.Left ? 0 : 1] = attachmen
    tHand;
182.                     }
183.                 }
184.
185.                 if (isPrefab && (_attachmentHands[0
    ] == null || _attachmentHands[0].transform.parent
     != this.transform
186.                                  || _attachmentHands[1
    ] == null || _attachmentHands[1].transform.parent
     != this.transform)) {
187.                     return;
188.                 }
189.
190.                 if (_attachmentHands[0] == null) {

191.                     GameObject obj = new GameObject()
    ;
192.                     #if UNITY_EDITOR
193.                     Undo.RegisterCreatedObjectUndo(ob
    j, "Created GameObject");
194.                     #endif
195.                     _attachmentHands[0] = obj.AddComp
    onent<AttachmentHand>();
196.                     _attachmentHands[0].chirality = C
    hirality.Left;
```

```
197.                }
198.                _attachmentHands[0].gameObject.name
     = "Attachment Hand (Left)";
199.                if (_attachmentHands[0].transform.p
    arent != this.transform) _attachmentHands[0].tran
    sform.parent = this.transform;
200.
201.                if (_attachmentHands[1] == null) {

202.                  GameObject obj = new GameObject()
    ;
203.                  #if UNITY_EDITOR
204.                  Undo.RegisterCreatedObjectUndo(ob
    j, "Created GameObject");
205.                  #endif
206.                  _attachmentHands[1] = obj.AddComp
    onent<AttachmentHand>();
207.                  _attachmentHands[1].chirality = C
    hirality.Right;
208.                }
209.                _attachmentHands[1].gameObject.name
     = "Attachment Hand (Right)";
210.                if (_attachmentHands[1].transform.p
    arent != this.transform) _attachmentHands[1].tran
    sform.parent = this.transform;
211.
212.                // Organize left hand first in sibl
    ing order.
213.                _attachmentHands[0].transform.SetSi
    blingIndex(0);
214.                _attachmentHands[1].transform.SetSi
    blingIndex(1);
215.              }
216.            }
217.
218.            private void refreshAttachmentHandTrans
    forms() {
219.              if (this == null) return;
220.
221.              #if UNITY_EDITOR
222.              if (getIsPrefab()) return;
223.              #endif
```

```
224.
225.             bool requiresReinitialization = false
    ;
226.
227.             if (_attachmentHands == null) {
228.               requiresReinitialization = true;
229.             }
230.             else {
231.               foreach(var hand in _attachmentHand
    s) {
232.                 if (hand == null) {
233.                     // AttachmentHand must have b
    een destroyed
234.                     requiresReinitialization = tr
    ue;
235.                     break;
236.                   }
237.
238.                   hand.refreshAttachmentTransform
    s(_attachmentPoints);
239.                 }
240.
241.                 #if UNITY_EDITOR
242.                 EditorUtility.SetDirty(this);
243.                 #endif
244.               }
245.
246.               if (requiresReinitialization) {
247.                 reinitialize();
248.               }
249.             }
250.
251.         #if UNITY_EDITOR
252.             private bool getIsPrefab() {
253.               PrefabType prefabType = PrefabUtili
    ty.GetPrefabType(this.gameObject);
254.               return (prefabType == PrefabType.Pr
    efab || prefabType == PrefabType.ModelPrefab);
255.             }
256.         #endif
257.           }
```

```
258. }
```

Kode Sumber C. 5 Fungsi Untuk Mengecek Ruas-ruas Jari

```
63. using UnityEngine;
64. using System.Collections;
65. using System;
66. using Leap.Unity.Attributes;
67. 
68. namespace Leap.Unity{
69. 
70.     public class ExtendedFingerDetector : Detecto
    r {
71.           public Camera camera;
72. 
73.           public bool grab = false;
74. 
75.       public float Period = .1f; //seconds
76. 
77.       [Tooltip("The hand model to watch. Set auto
    matically if detector is on a hand.")]
78.       public HandModelBase HandModel = null;
79. 
80. 
81.       public PointingState Thumb = PointingState.
    Either;
82. 
83.       public PointingState Index = PointingState.
    Either;
84. 
85.       public PointingState Middle = PointingState
    .Either;
86. 
87.       public PointingState Ring = PointingState.E
    ither;
88. 
89.       public PointingState Pinky = PointingState.
    Either;
90. 
91.       public int MinimumExtendedCount = 0;
92. 
93.       public int MaximumExtendedCount = 5;
```

```
94.
95.       public bool ShowGizmos = true;
96.
97.       private IEnumerator watcherCoroutine;
98.
99.       void OnValidate() {
100.        int required = 0, forbidden = 0;
101.        PointingState[] stateArray = { Thumb, Ind
    ex, Middle, Ring, Pinky };
102.        for (int i = 0; i < stateArray.Length; i+
    +) {
103.          var state = stateArray[i];
104.          switch (state) {
105.            case PointingState.Extended:
106.              required++;
107.              break;
108.            case PointingState.NotExtended:
109.              forbidden++;
110.              break;
111.            default:
112.              break;
113.          }
114.          MinimumExtendedCount = Math.Max(require
    d, MinimumExtendedCount);
115.          MaximumExtendedCount = Math.Min(5 - for
    bidden, MaximumExtendedCount);
116.          MaximumExtendedCount = Math.Max(require
    d, MaximumExtendedCount);
117.        }
118.
119.      }
120.
121.      void Awake() {
122.        watcherCoroutine = extendedFingerWatcher(
    );
123.      }
124.
125.      void OnEnable() {
126.        StartCoroutine(watcherCoroutine);
127.
128.      }
129.
```

```
130.      void OnDisable() {
131.        StopCoroutine(watcherCoroutine);
132.        Deactivate();
133.
134.      }
135.
136.      IEnumerator extendedFingerWatcher() {
137.        Hand hand;
138.        while (true) {
139.          bool fingerState = false;
140.          if (HandModel != null && HandModel.IsTr
    acked) {
141.            hand = HandModel.GetLeapHand();
142.            if (hand != null) {
143.              fingerState = matchFingerState(hand
    .Fingers[0], Thumb)
144.                && matchFingerState(hand.Fingers[
    1], Index)
145.                && matchFingerState(hand.Fingers[
    2], Middle)
146.                && matchFingerState(hand.Fingers[
    3], Ring)
147.                && matchFingerState(hand.Fingers[
    4], Pinky);
148.
149.              int extendedCount = 0;
150.              for (int f = 0; f < 5; f++) {
151.                if (hand.Fingers[f].IsExtended) {

152.                  extendedCount++;
153.                }
154.              }
155.              fingerState = fingerState &&
156.                            (extendedCount <= Maxi
    mumExtendedCount) &&
157.                            (extendedCount >= Mini
    mumExtendedCount);
158.              //not extended finger
159.              if (HandModel.IsTracked && fingerSt
    ate)
160.              {
161.                Activate();
```

```
                grab = true;
              }

              //extended finger
              else if (!HandModel.IsTracked || !f
    ingerState)
              {
                Deactivate();
                grab = false;
              }
            }
          }
        else if (IsActive) {
        Deactivate();
      }
      yield return new WaitForSeconds(Period);
        }
      }

    private bool matchFingerState(Finger finger,
    PointingState requiredState) {
    return (requiredState == PointingState.Either
    ) ||
         (requiredState == PointingState.Extended
     && finger.IsExtended) ||
         (requiredState == PointingState.NotExten
    ded && !finger.IsExtended);
    }
}
public enum PointingState { Extended, NotExtended
    , Either }
}
```

Kode Sumber C. 6 Fungsi Untuk Mengecek Extended Finger

BIODATA PENULIS

GD Wahyu Nugraha Subagia, lahir di Denpasar pada tanggal 10 Maret 1997. Penulis menempuh pendidikan mulai dari TK Darma Santi (2001-2002), TK Tresnaning Kumara Santhi (2002-2003), SD Negeri 4 Ubung Kaja (2003-2009), SMP Negeri 10 Denpasar (2009-2012), SMA Negeri 4 Denpasar (2012-2015) dan saat ini melanjutkan studinya di Departemen Informatika Institut Teknologi Sepuluh Nopember Surabaya.

Selama menempuh dunia perkuliahan, penulis aktif mengikuti organisasi Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS menjadi Staf Departemen Minat Bakat, organisasi Tim Pembina Kerohanian Hindu (TPKH) ITS menjadi Staf Departemen Seni dan Minat Bakat, kepanitiaan FTIF Festival 2017 menjadi Staf Internal, kepanitiaan Gemastik 2018 menjadi Staf Acara, Kepanitiaan Schematics 2017 menjadi BPH REEVA.

Dalam menyelesaikan Pendidikan sarjana, penulis mengambil bidang minat Interaksi, Grafika dan Seni (IGS). Untuk melakukan komunikasi kepada penulis dapat dihubungi melalui alamat ***e-mail*: nugrahawahyu5@gmail.com**.