**IMPLEMENTASI SIMULASI PELATIHAN KASIR TOKO RETAIL BERBASIS *VIRTUAL REALITY* MENGGUNAKAN *GAME ENGINE UNITY 3D***

**SKRIPSI**

diajukan untuk menempuh ujian sarjana

pada Fakultas Matematika dan Ilmu Pengetahuan Alam  
Universitas Padjadjaran

AMIR SALIM

NPM 140810210015



UNIVERSITAS PADJADJARAN  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
PROGRAM STUDI TEKNIK INFORMATIKA  
SUMEDANG  
2025

DAFTAR ISI

[DAFTAR ISI ii](#_Toc215050023)

[DAFTAR TABEL iv](#_Toc215050024)

[DAFTAR GAMBAR v](#_Toc215050025)

[BAB I 1](#_Toc215050026)

[1.1 Latar Belakang 1](#_Toc215050027)

[1.2 Identifikasi Masalah 4](#_Toc215050028)

[1.3 Batasan Masalah 4](#_Toc215050029)

[1.4 Maksud, Tujuan, dan Manfaat Penelitian 5](#_Toc215050030)

[1.5 Metodologi Penelitian 5](#_Toc215050031)

[1.6 Sistematika Penulisan 6](#_Toc215050032)

[BAB II 8](#_Toc215050033)

[2.1 Perusahaan Retail 8](#_Toc215050034)

[2.2 Kasir 8](#_Toc215050035)

[2.3 Virtual Reality 9](#_Toc215050036)

[2.4 Unity Game Engine 10](#_Toc215050037)

[*2.5* Bahasa Pemrograman C# 11](#_Toc215050038)

[2.6 Game Development Life Cycle (GDLC) 11](#_Toc215050039)

[2.7 Pengujian Aplikasi 13](#_Toc215050040)

[2.7.1 Uji Fungsionalitas dengan Black Box Testing 14](#_Toc215050041)

[2.7.2 Uji Usabilitas dengan Evaluasi Heuristik 15](#_Toc215050042)

[BAB III 20](#_Toc215050044)

[3.1 Metode Pengembangan Sistem 20](#_Toc215050045)

[*3.2* *Initiation* 20](#_Toc215050046)

[3.2.1 Analisis Persona 21](#_Toc215050047)

[3.2.2 Analisis Kebutuhan Fungsional 21](#_Toc215050048)

[3.2.3 Analisis Kebutuhan Non-Fungsional 22](#_Toc215050049)

[3.3 *Pre-production* 24](#_Toc215050050)

[3.3.1 Rancangan Umum Aplikasi 25](#_Toc215050051)

[3.3.2 *Storyline* 26](#_Toc215050052)

[3.3.3 *Storyboard* 27](#_Toc215050053)

[3.3.4 *Core Loop* 29](#_Toc215050054)

[3.3.5 *Core Mechanics* 29](#_Toc215050055)

[3.3.6 Aset Seni 31](#_Toc215050056)

[3.3.7 *Technical Aspects* 35](#_Toc215050081)

[BAB IV 40](#_Toc215050082)

[4.1 Production 40](#_Toc215050083)

[4.1.1 Aset dan Lingkungan 40](#_Toc215050084)

[4.1.2 Pembuatan Efek Suara 44](#_Toc215050085)

[4.1.3 Pembuatan *User Interface* 50](#_Toc215050086)

[4.1.4 Pembuatan Konsumen 51](#_Toc215050087)

[4.1.5 Pembuatan Interaksi 54](#_Toc215050088)

[4.1.6 Pembuatan Sistem Kasir 63](#_Toc215050089)

[4.1.7 Pembuatan *Game Manager* 70](#_Toc215050090)

[DAFTAR PUSTAKA 78](#_Toc215050091)

DAFTAR TABEL

[Tabel 3.1 Storyboard 17](#_Toc52929958)

[Tabel 4.1 Waktu Penelitian 19](#_Toc52929959)

DAFTAR GAMBAR

[Gambar 3.1 Peta Area Lingkungan *game* 16](#_Toc52930058)

# **BAB I**

**PENDAHULUAN**

Bab ini membahas pengantar penelitian yang mencakup latar belakang, identifikasi masalah, batasan masalah, tujuan, manfaat, metodologi, serta sistematika penulisan. Seluruh bagian tersebut disusun untuk memberikan gambaran awal mengenai arah dan ruang lingkup penelitian.

1. **Latar Belakang**

Perkembangan teknologi informasi yang sangat cepat mewajibkan tiap pemegang kepentingan untuk berinovasi agar mendapatkan keuntungan terbaik dan kesempatan yang baru seiring dengan perkembangan teknologi. Bisnis ritel sebagai salah satu bisnis yang marak di masyarakat tidak luput pada hal ini. Menurut Shankar, Venkatesh, et al (2021) dengan mengadopsi teknologi terkini seperti *Virtual Reality* (VR) atau *Internet of Things*, suatu bisnis ritel mendapatkan keuntungan kepada beberapa pemangku kepentingan seperti karyawan dengan meningkatnya efektivitas dan produktivitas beserta pelanggan dengan meningkatnya kepuasan dan pengalaman belanja yang meningkat.

Dalam beberapa tahun terakhir, implementasi pelatihan menggunakan *Virtual Reality* telah berkembang dengan pesat. Pada bisnis yang mengimplementasikan *Customer Relationship Management* (CRM) penggunaan pelatihan menggunakan Virtual Reality dapat meningkatkan kecakapan karyawan, nilai kepuasan pelanggan, dan kecepatan pelatihan karyawan (Boppana, V. R., 2022). Pada kasus perusahaan retail yang menggunakan pelatihan Virtual Reality, karyawan merasa lebih percaya diri ketika berhadapan dengan pelanggan di dunia nyata karena Virtual Reality dapat mensimulasikan berbagai macam skenario ketika berhadapan dengan pelanggan pada dunia nyata (Boppana, V. R., 2022).

Yang, C. J., & Wu, C. F (2022) melakukan penelitian yang membandingkan apakah terdapat perbedaan oleh karyawan yang mengikuti pelatihan menggunakan Virtual Reality dan pelatihan konvensional menggunakan gambar. Penelitian ini membandingkan akurasi, kecepatan, dan motivasi akan pembelajaran dari karyawan yang dilatih menggunakan dua metode pelatihan yang berbeda. Penelitian ini menghasilkan bahwa rata rata karyawan yang dilatih menggunakan Virtual Reality lebih tinggi dari konvensional dengan detail 10,20 tingkat akurasi oleh pelatihan VR dan 8,90 oleh pelatihan konvensional. Dalam segi kecepatan, Karyawan yang dilatih dengan Virtual Reality rata rata lebih cepat dalam menyelesaikan tugas dengan rata-rata waktu penyelesaian sebesar 3,80 sedangkan rata-rata penyelesaian tugas oleh pelatihan konvensional adalah 8,70. Dalam segi motivasi akan pembelajaran, karyawan yang dilatih dengan menggunakan Virtual Reality lebih nyaman dan asistensi yang diberikan oleh Virtual Reality lebih besar daripada pelatihan konvensional.

Perkembangan dunia ritel menunjukkan bahwa metode pelatihan kasir mengalami transformasi signifikan seiring kemajuan teknologi. Arkenback dan Lundin (2023) menjelaskan bahwa sejak awal abad ke-20, pelatihan kasir umumnya dilakukan melalui video instruksional yang menekankan keterampilan transaksi dan pelayanan pelanggan. Namun, meskipun teknologi *checkout* terus berkembang dari mekanis, komputerisasi hingga sistem terkoneksi, metode pelatihan berbasis video terbukti stagnan sejak tahun 1990-an dan kurang mampu menjawab kebutuhan keterampilan kasir di era digital yang menuntut kecepatan, ketepatan, serta kecakapan emosional dalam melayani pelanggan. Keterbatasan tersebut membuka ruang untuk pendekatan baru yang lebih interaktif dan adaptif. Salah satu alternatif yang menjanjikan adalah penggunaan teknologi Virtual Reality (VR). Studi Farzinnejad et al. (2025) menunjukkan bahwa VR mampu menciptakan lingkungan pelatihan imersif yang mensimulasikan interaksi pelanggan, pengelolaan produk, hingga proses transaksi.

Hasil uji coba pada pelatihan penjualan roti membuktikan bahwa peserta memperoleh pengalaman belajar yang lebih nyata, responsif, serta meningkatkan kepercayaan diri sebelum terjun ke dunia kerja. Selanjutnya, penelitian Oprins dan Korteling (2014) melalui Cashier Trainer juga mengonfirmasi bahwa simulasi digital dapat menghasilkan transfer of training yang efektif, bahkan pada beberapa aspek melampaui pelatihan On-the-Job Training (OJT) konvensional. Temuan ini memperlihatkan bahwa simulasi digital memungkinkan pembelajaran tugas rutin maupun non-rutin secara berulang dan terstruktur, dengan dukungan umpan balik otomatis yang dapat menggantikan sebagian peran instruktur. Berdasarkan kajian tersebut, penelitian mengenai penerapan VR untuk simulasi pelatihan kasir menjadi relevan dan penting dilakukan, tidak hanya untuk meningkatkan efektivitas pelatihan dan kesiapan kerja, tetapi juga sebagai solusi efisien dalam mengurangi biaya serta risiko dari OJT di dunia nyata.

Penelitian ini bertujuan untuk mengimplementasikan pelatihan kasir berbasis Virtual Reality menggunakan Unity 3D. Walaupun secara kemampuan tidak ada perbedaan signifikan antara Unity dan kompetitornya Unreal Engine dalam tahap pengembangan dunia Virtual, namun dalam segi kekuatan komputer Unity unggul pada penggunaan RAM, sumberdaya CPU, sumber daya GPU (Ciekanowska, A, et al, 2021). Berdasarkan hal tersebut, peneliti lebih memilih menggunakan Unity 3D daripada Unreal Engine untuk mengefesiensikan tahap pengembangan aplikasi.

1. **Identifikasi Masalah**

Berdasarkan latar belakang, terdapat urgensi untuk menciptakan bentuk pelatihan kasir yang lebih imersif dengan menggunakan VR guna menciptakan pengurangan biaya serta risiko dari OJT di dunia nyata. Adapun masalah yang akan dicari solusinya dalam penelitian ini adalah sebagai berikut:

Bagaimana membentuk lingkungan virtual dan mengintegrasikannya menjadi sebuah aplikasi VR untuk pelatihan kasir?

Bagaimana hasil evaluasi dari aplikasi VR untuk pelatihan kasir dilihat dari aspek fungsionalitas dan usabilitas?

1. **Batasan Masalah**

Dari identifikasi masalah tersebut, maka dalam penelitian ini terdapat beberapa batasan masalah pada sebagai berikut:

1. Aplikasi yang dijalankan optimal di Meta Quest 2
2. Aplikasi yang dikembangkan menggunakan Unity dan bahasa pemrograman C#
3. Batasan dari skenario yang dibuat adalah proses transaksi, *scan* barang, pengemasan barang, dan konsumen membayar barang.
4. **Maksud, Tujuan, dan Manfaat Penelitian**

Maksud dari penelitian ini adalah untuk membuat media pelatihan kasir menggunakan aplikasi berbasis Virtual Reality.

Adapun tujuan yang ingin dicapai oleh penulis dari penelitian ini adalah:

1. Membentuk lingkungan virtual dan mengintegrasikannya menjadi sebuah aplikasi VR untuk pelatihan kasir.
2. Mendapatkan hasil evaluasi dari aplikasi VR pelatihan kasir dilihat dari aspek fungsionalitas dan usabilitas.

Dari penelitian ini penulis berharap memberikan manfaat dengan hasil aplikasi simulasi VR pelatihan kasir yang dapat digunakan sebagai media pelatihan. Karyawan sebagai kasir mendapatkan kemudahan untuk melakukan pelatihan kasir. Selain itu, aplikasi ini dapat memberikan manfaat berupa studi kasus untuk pengembangan aplikasi VR menggunakan unity 3D terutama penggunaan *package* Meta XR.

1. **Metodologi Penelitian**

Jenis metode penelitian yang akan digunakan ialah metode pengembangan *Game Development Life Cycle* (GDLC) berdasarkan (Ramadhan & Widyani, 2013). Penulis memilih metode ini karena tahapan yang sederhana dan cocok dengan aplikasi yang akan dibuat. Adapun tahapan dari metode GDLC yang akan digunakan adalah sebagai berikut:

1. *Initiation,* merupakan tahapan analisis ide aplikasi
2. *Pre-production* merupakan tahapan perencanaan proyek dan desain *game.*
3. *Production* merupakan tahapan pembuatan baik dalam aspek teknikal atau artistik.
4. *Testing* merupakan tahapan pengujian permainan untuk mencari *bug* pada *game.*
5. *Beta,* merupakan tahapan untuk menguji hasil aplikasi ke pihak eksternal untuk mendapatkan saran dan kritik akan aplikasi.
6. *Release,* merupakan tahapan rilis aplikasi
7. **Sistematika Penulisan**

Untuk memberi gambaran yang jelas tentang penelitian ini, maka disusunlah sistematika penulisan yang berisi materi yang akan dibahas pada setiap bab. Sistematika dalam penulisan tugas akhir ini adalah sebagai berikut:

**BAB I PENDAHULUAN**

Pada bab ini dijelaskan tentang latar belakang dari topik penulisan skripsi, pokok permasalahan berupa identifikasi dan batasan masalah, tujuan dan manfaat yang diharapkan dari penulisan skripsi, metodologi yang digunakan serta sistematika penulisan.

**BAB II TINJAUAN PUSTAKA**

Pada bab ini dijelaskan seluruh landasan teori yang berhubungan dengan penelitian, yaitu tentang metode pengembangan perangkat lunak yang digunakan, penjelasan teoritis mengenai bahasa pemrograman serta teori lainnya guna memahami permasalahan yang dibahas.

**BAB III ANALISIS DAN PERANCANGAN**

Pada bab ini dijelaskan tentang metode pengembangan aplikasi yang digunakan meliputi tahap *initiation* aplikasi berupa analisis persona, kebutuhan fungsional dan non-fungsional dan *pre-processing ­*pada GDLC yakni rancangan umum aplikasi, *storyline, storyboard, core loop, core mechanics,* Aset seni, da*n technical aspects*.

**BAB IV HASIL DAN PEMBAHASAN**

Pada bab ini dijelaskan tentang implementasi aplikasi yang telah dibangun, tampilan aplikasi, pengujian aplikasi, serta hasil dari pengembangan aplikasi menggunakan Unity.

**BAB V KESIMPULAN DAN SARAN**

Bab ini merupakan penutup yang berisi kesimpulan dan saran dari penelitian yang sudah dilakukan.

# **BAB II**

**TINJAUAN PUSTAKA**

Bab ini menyajikan landasan teori yang meliputi konsep retail, peran kasir, teknologi Virtual Reality, Unity 3D, bahasa pemrograman C#, serta metode pengembangan GDLC. Seluruh teori tersebut menjadi dasar penting dalam mendukung proses perancangan dan implementasi aplikasi simulasi pelatihan kasir.

* 1. Perusahaan Retail

Perusahaan retail adalah entitas bisnis yang menjual produk langsung kepada konsumen akhir untuk penggunaan pribadi, bukan untuk dijual kembali (Levy & Weitz, 2020). Retail dapat berbentuk toko fisik, online, atau kombinasi dari keduanya. Perusahaan retail menggunakan teknologi seperti sistem manajemen inventaris, point of sale (POS), dan analitik data untuk meningkatkan efisiensi operasional dan pengalaman pelanggan.

Menurut penelitian oleh Smith et al. (2021), implementasi teknologi analitik dalam perusahaan retail seperti Walmart meningkatkan efisiensi rantai pasokan dan mengurangi biaya operasional. Sehingga, pemanfaatan teknologi untuk retail sangat penting dalam aspek inovasi bisnis suatu perusahaan, salah satunya adalah dengan menggunakan pelatihan karyawan dengan Virtual Reality.

* 1. Kasir

Kasir adalah individu atau sistem otomatis yang bertanggung jawab untuk memproses transaksi pembayaran di titik penjualan (POS) (Johnson, 2020). Kasir modern sering menggunakan perangkat lunak POS yang terintegrasi dengan sistem inventaris untuk mempercepat transaksi dan mengurangi kesalahan manusia dalam melakukan pekerjaan.

Penelitian oleh Lee & Kim (2022) menunjukkan bahwa penerapan self-checkout di toko retail besar seperti Target meningkatkan kepuasan pelanggan dan efisiensi operasional. Yang, C. J., & Wu, C. F (2022) melakukan penelitian yang membandingkan apakah terdapat perbedaan oleh kasir yang mengikuti pelatihan menggunakan Virtual Reality dan pelatihan konvensional menggunakan gambar. Penelitian ini membandingkan akurasi, kecepatan, dan motivasi akan pembelajaran dari karyawan yang dilatih menggunakan dua metode pelatihan yang berbeda. Penelitian ini menghasilkan bahwa rata rata karyawan yang dilatih menggunakan Virtual Reality lebih tinggi dari konvensional dengan detail 10,20 tingkat akurasi oleh pelatihan Virtual Reality dan 8,90 oleh pelatihan konvensional. Dalam segi kecepatan, Karyawan yang dilatih dengan Virtual Reality rata rata lebih cepat dalam menyelesaikan tugas dengan rata rata waktu penyelesaian sebesar 3,80 sedangkan rata rata penyelesaian tugas oleh pelatihan konvensional adalah 8,70.

* 1. Virtual Reality

Virtual Reality (VR) adalah teknologi yang memungkinkan pengguna untuk berinteraksi dengan lingkungan digital yang disimulasikan secara imersif menggunakan perangkat seperti headset VR (Burdea & Coiffet, 2021). VR digunakan dalam berbagai bidang seperti pelatihan medis, simulasi penerbangan, pendidikan, dan hiburan.

Studi oleh Gonzalez et al. (2023) menunjukkan bahwa penggunaan VR dalam pelatihan bedah meningkatkan keterampilan praktis dan mengurangi waktu belajar dibandingkan dengan metode tradisional. Pada bisnis yang mengimplementasikan Customer Relationship Management (CRM) penggunaan pelatihan menggunakan Virtual Reality dapat meningkatkan kecakapan karyawan, nilai kepuasan pelanggan, dan kecepatan pelatihan karyawan (Boppana, V. R., 2022)

* 1. Unity Game Engine

Unity adalah mesin permainan lintas platform yang digunakan untuk mengembangkan game 2D dan 3D serta aplikasi interaktif lainnya (Goldstone, 2020). Unity digunakan dalam pengembangan game, simulasi, aplikasi augmented reality (AR) dan VR, serta visualisasi arsitektur baik dalam pekerjaan sebagai tim ataupun individu.

Penelitian oleh Zhang & Liu (2021) menunjukkan bahwa Unity digunakan secara luas dalam pengembangan aplikasi pendidikan berbasis VR, meningkatkan keterlibatan dan pemahaman siswa. adalah bahasa pemrograman berorientasi objek yang dikembangkan oleh Microsoft, digunakan secara luas dalam pengembangan aplikasi desktop, web, dan game (Albahari & Albahari, 2020). Menurut studi oleh Hernandez et al. (2022), penggunaan C# dalam pengembangan game dengan Unity meningkatkan efisiensi coding dan mempermudah integrasi dengan berbagai platform.

Unity mendukung proses pembuatan dunia *game* dengan cara membagi tiap tiap dunia atau tangkapan layar pada tiap aplikasi menjadi *scene.* Di dalam *scene,* terdapat objek – objek pada *game* seperti lingkungan toko dan objek – objek yang dapat berinteraksi dengna pemain. Di dalam *game object,* terdapat *game component* yang berfungsi untuk memberikan perilaku tertentu pada *game object.* Salah satu contohnya adalah, terdapat *game object* kubus yang diberikan *game component* dengan nama *collider* untuk memungkinkan objek lain dapat menyentuh objek ini. Selain *collider, Rigidbody* juga merupakan *game component* yang banyak digunakan dikarenakan fungsinya untuk memberikan perilaku fisika pada *game object.*

* 1. Bahasa Pemrograman C#

C# adalah bahasa pemrograman berorientasi objek yang dikembangkan oleh Microsoft, digunakan secara luas dalam pengembangan aplikasi desktop, web, dan game (Albahari & Albahari, 2020). Menurut studi oleh Hernandez et al. (2022), penggunaan C# dalam pengembangan game dengan Unity meningkatkan efisiensi coding dan mempermudah integrasi dengan berbagai platform.

C# sebagai bahasa pemrograman yang terdapat pada Unity, digunakan di penelitian ini untuk mengembangkan aplikasi berbasis Virtual reality. Dikarenakan paradigm C# yang berbasis objek, memudahkan pengembang untuk mengambil beberapa komponen atau *library* yang relevan dengan pengembangan aplikasi Virtual Reality.

* 1. Game Development Life Cycle (GDLC)

Beberapa definisi GDLC yang ada diantaranya adalah milik Blitz Games Studios dengan proses secara urut pitch, pre-production, main production, alpha, beta, master. Lalu GDLC yang didefinisikan oleh Arnold Hendrick dengan proses secara urut prototype, pre-production, production, beta, live. Ada juga GDLC yang didefinisikan oleh Joshua McGrath yang memiliki proses design, develop, evaluate, redevelop, test, review release, release. Dan juga GDLC yang didefinisikan oleh Heather Chandler yang memiliki proses *pre-production, production, testing, post- production*). Berdasarkan beberapa GDLC tersebut. Ramadhan & Widyani memberikan alur GDLC yang menyempurnakan beberapa alur GDLC sebelumnya yang terdiri dari *initiation, pre-production, production, testing, Beta,* dan *release* (Ramadan & Widyani, 2013).

A diagram of a production cycle

AI-generated content may be incorrect.

Gambar 2. Alur GDLC menurut Ramadan & Widyani

*Initiation* membahas metode untuk menghasilkan ide dan konsep game, termasuk panduan brainstorming dalam bentuk pertanyaan. *Pre-production* menjelaskan elemen desain game seperti deskripsi, karakter, alur cerita, kontrol, fitur, serta pembuatan GDD, pembuatan prototipe, dan daftar pemeriksaan tahap pre-production. Production berfokus pada pemrograman dan pembuatan aset, dilengkapi panduan kualitas aset, contoh perubahan arsitektur game, dan daftar deliverable. *Testing* menjelaskan metode pengujian sesuai kriteria kualitas pada setiap tahap prototipe beserta contohnya. *Beta* menekankan pentingnya beta testing, jenisnya, serta menyediakan metode, checklist, dan contoh kuesioner playtesting. *Release* menguraikan cara merilis paket game, aktivitas pascaproduksi, dan perencanaan lanjutan untuk pengembangan game (Ramadan & Widyani, 2013).

Adapun definisi lain GDLC dalah proses sistematis yang mencakup tahapan-tahapan dalam pengembangan permainan, mulai dari konsepsi hingga peluncuran dan pemeliharaan (Rogers, 2020). Penelitian oleh Thompson & Nguyen (2021) menunjukkan bahwa penerapan GDLC yang efektif dalam pengembangan game indie meningkatkan kualitas produk akhir dan mempercepat waktu peluncuran ke pasar.

* 1. Pengujian Aplikasi

Pengujian aplikasi merupakan tahapan *testing* dalam GDLC dimana aplikasi akan diuji kesesuaiannya sebelum diberikan kepada pengguna. Tahapan *testing* pada penelitian ini dibagi menjadi dua tahapan, tahapan pengujian internal yang dilakukan oleh peneliti dan tahapan pengujian eksternal yang dilakukan diluar peneliti. Untuk pengujian Internal, *testing* dilakukan menggunakan *Blackbox testing*, sedangkan untuk tahapan pengujian eksternal menggunakan Evaluasi heuristik.

Pengujian aplikasi dibagi menjadi dua untuk memaksimalkan produk akhir dari aplikasi agar sesuai dengan kebutuhan dan tujuan yang diinginkan. Pengujian internal memaksimalkan aplikasi agar sesuai terlebih dahulu dengan target peneliti sedangkan pengujian eksternal memaksimalkan aplikasi agar mendapatkan saran dan kesesuaian

1. Uji Fungsionalitas dengan Black Box Testing

Black Box Testing merupakan metode pengujian perangkat lunak yang berfokus pada pemeriksaan fungsi-fungsi aplikasi tanpa memeriksa struktur kode internal. Tang (2017) menjelaskan bahwa dalam black box testing, aplikasi diperlakukan sebagai “kotak hitam” di mana tester hanya memberikan input dan mengevaluasi apakah output sesuai dengan spesifikasi. Pengujian ini dilakukan pada level antarmuka dan fungsi program.

Equivalence Partition (EP) adalah teknik dalam black box testing yang membagi domain input menjadi beberapa kelas (partition) yang dianggap setara. Setiap kelas mewakili kelompok nilai input yang diprediksi menghasilkan output yang sama. Dengan demikian, penguji cukup mengambil satu sampel dari tiap kelas untuk diuji, sehingga jumlah test case menjadi lebih efisien.

Tang (2017) memasukkan equivalence class partition sebagai salah satu teknik utama dalam black box testing untuk memvalidasi fungsi melalui pembagian input menjadi kelas-kelas yang representatif. Gunawan et al. (2023) memperlihatkan bagaimana black box testing berbasis Equivalence Partition digunakan secara aplikatif untuk memastikan fungsi VR berjalan sesuai harapan pengguna. Contoh pengujiannya meliputi:

1. Validasi suara berdasarkan kecepatan pukulan (dibagi dalam kelas input)
2. Validasi sistem navigasi dan collision
3. Pengujian fungsi menu
4. Testing suara tiap elemen Wilahan secara terpisah
5. Uji Usabilitas dengan Evaluasi Heuristik

Evaluasi heuristik awalnya diperkenalkan oleh Jakob Nielsen dan Rolf Molich untuk menilai usability antarmuka 2D. Namun, perkembangan sistem interaktif berbasis virtual reality (VR), multimedia, dan 3D environments membutuhkan pendekatan evaluasi yang lebih spesifik. Sutcliffe & Gault (2004) kemudian mengusulkan seperangkat heuristik baru yang disesuaikan dengan karakteristik VR, terutama dari segi imersi, interaktivitas, dan navigasi. Terdapat 12 kriteria penilaian yang digunakan untuk mengukur usabilitas aplikasi menurut teori Sutcliffe, rincian 12 kriteria tersebut terdapat pada tabel 2.1.

Tabel 2. Kriteria Penilaian Usabilitas Menurut Teori Sutcliffe.

| No | Kriteria | Keterangan |
| --- | --- | --- |
| 1 | *Natural Engagement* | Interaksi penggunadi dalam VR mirip dengan keadaan di dunia nyata. |
| 2 | *Natural expression of action* | Pengguna memiliki interaksi bebas dalam melakukan eksplorasi dalam VR. |
| 3 | *Compatibility with user’s task and domain,* | Objek yang terdapat dalam VR memiliki perilaku yang sesuai dengan aslinya. |
| 4 | *Close coordination of action and representation* | Representasi objek dengan nyata dan tanpa *lag.* |
| 5 | *Realistic feedback* | Umpan balik yang diberikan oleh aplikasi dapat dilihat dan sesuai perintah pengguna |
| 6 | *Faithful viewpoint* | Representasi visual disesuaikan dengan persepsi normal pengguna. |
| 7 | *Navigation and orientation support* | Pengguna mengetahui posisi mereka di dalam VR |
| 8 | *Clear entry and exit point* | Pengguna dapat masuk dan keluar dari VR dengan mudah |
| 9 | *Consistent departures* | Objek ditampilkan dengan konsisten tanpa perubahan mendadak |
| 10 | *Support of learning* | Aplikasi memberikan pembelajaran bagi pengguna |
| 11 | *Clear turn-taking* | Aplikasi memiliki waktu giliran yang tepat dalam berkomunikasi tanpa tumpang tindih |
| 12 | *Sense of presence* | Pengguna merasakan sensasi seperti di dunia nyata ketika berada di dalam VR |

Terdapat beberapa kriteria yang akan dipakai untuk menilai aplikasi oleh ahli. Kriteria ini disesuaikan dengan kebutuhan aplikasi VR.Kriteria penilaian dibagi menjadi *usability of virtual environment* dan *interaction within virtual environment.* *Usability of virtual environment* meliputi segala aspek terkait usabilitas aplikasi. Sedangkan, *interaction within virtual environment* meliputi interaksi yang berfokus bagaimana interaksi di VR berjalan di aplikasi. Tabel 2.2 adalah kriteria penilaian *usability of virtual environment* dan Tabel 2.3 adalah kriteria penilaian *interaction within virtual environment*.

Tabel 2. Kriteria Penilaian *usability of virtual environment*

| No | Kriteria | Keterangan |
| --- | --- | --- |
| 1 | *Quality of visual output* | Kualitas gambar VR yang baik |
| 2 | *Quality of audio output* | Kualitas suara VR yang baik |
| 3 | *Consistency of sensory outputs* | Keselarasan gambar dan suara dalam VR |
| 4 | *Natural engagement* | Keserupaan interaksi VR dengan di dunia nyata |
| 5 | *Faithful viewpoints* | Tidak ada jeda waktu antar rotasi dan perubahan sudut pandang |
| 6 | *Support for learning* | Tombol jelas dan mudah dipahami |
| 7 | *Compatibility with user’s task and domain* | Perilaku objek dari dunia virtual sama dengan aslinya |
| 8 | *Sense of presence* | VR memberikan kesan seakan di dunia nyata |
| 9 | *Cybersickness* | VR tidak membuat pusing |

Tabel 2. Kriteria Penilaian *Interaction Within Virtual Environment*

|  |  |  |
| --- | --- | --- |
| No | Kriteria | Keterangan |
| 1 | *Scenario* | Skenarion dan situasi yang ada pada VR relevan sebagai media pelatihan kasir |
| 2 | *Overall content strength* | *Game* karakter mampu mendistraksi pengguna |
| 3 | *User’s task* | Terdapat tugas dan scenario yang jelas |

Kriteria – kriteria tersebut akan ditentukan untuk menentukan Kesimpulan apakah aplikasi VR memenuhi syarat sebagai media pelatihan kasir. Untuk menilai tiap kriteria tersebut, tiap kriteria akan dibentuk menjadi lima poin dalam skala rikert dengan rincian sangat tidak sesuai sampai sangat sesuai dengan rentang angka satu sampai dengan lima.

Skala Likert pertama kali diperkenalkan oleh Rensis Likert pada tahun 1932 sebagai salah satu metode pengukuran sikap, opini, dan persepsi seseorang terhadap suatu objek. Menurut Sugiyono (2017), skala Likert digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang mengenai fenomena sosial. Responden diminta memberikan tingkat persetujuan atau ketidaksetujuan terhadap serangkaian pernyataan yang disusun oleh peneliti. Skala poin likert ditampilkan dengan format pada tabel 2.4 berikut.

Tabel 2. Skala Poin Likert

|  |  |
| --- | --- |
| Pilihan | Nilai |
| Sangat Tidak Setuju | 1 |
| Tidak Setuju | 2 |
| Cukup | 3 |
| Setuju | 4 |
| Sangat Setuju | 5 |

Untuk mengolah data skala likert, dilakukan beberapa tahapan yakni :

1. Menentukan Interval untuk memproses penilaian dengan persamaan 2.1

Persamaan 2. Interval Proses Penilaian

1. Menentukan total nilai untuk masing masing sikap tiap penilaian menggunakan persamaan 2.2

Persamaan 2. Total Nilai Untuk Masing Masing Sikap

Keterangan: T = total jumlah responen Pn = Pilihan angka nilai Likert

1. Menghitung total nilai interpretasi masing-masing hasil menggunakan persamaan 2.3

Persamaan 2. Total Nilai Interpretasi

Keterangan: A=angka tertinggi nilai Likert T=jumlah responden

# **BAB III**

**ANALISIS DAN PERANCANGAN**

Bab ini membahas analisis kebutuhan serta perancangan sistem sebagai langkah awal pengembangan aplikasi simulasi pelatihan kasir berbasis Virtual Reality. Analisis dan rancangan disusun untuk memastikan aplikasi berjalan sesuai tujuan penelitian dan memberikan pengalaman pelatihan yang optimal bagi pengguna.

1. Metode Pengembangan Sistem

Aplikasi menggunakan pengembangan sistem dengan metode *Game Development Life Cycle* (GDLC) karena *iterative*. Tahapan yang digunakan dalam GDLC meliputi tahap *initiation, pre-production, production, testing, beta,*dan *release*.

Aplikasi ”*Virtual Reality* Simulasi Pelatihan Kasir” menggunakan banyak aset dan keilmuan yang beragam seperti *programming, animasi, art,* dan musik sehingga GDLC cocok untuk mengintegrasi semua aspek tersebut. Permainan pada aplikasi ini memerlukan metode GDLC untuk merancang *game* mekanik, *game rules, fun experience,* dan *core loop.*

1. *Initiation*

Tahapan *initiation* meliputi identifikasi masalah dan objektif penelitian. Identifikasi terhadap konsep dan deskripsi mengenai perangkat lunak dan perangkat keras juga menjadi bahasan pada tahap *initiation*. Selain itu, terdapat beberapa analisis yang dilakukan yakni analisis persona, analisis kebutuhan fungsional, dan analisis kebutuhan non-fungsional.

1. Analisis Persona

Pengguna utama dari sistem VR Simulasi Latihan Kasir terdiri dari beberapa kategori:

1. Mahasiswa/ Pelajar SMK Bisnis dan Manajemen : Mahasiswa atau pelajar yang mempelajari manajemen bisnis atau akuntansi dan memerlukan pelatihan kasir sebagai bagian dari kurikulum mereka.

2. Karyawan Baru di Toko Retail : Karyawan baru yang memerlukan pelatihan cepat dan efektif sebelum mulai bekerja secara langsung di kasir.

3. Instruktur/ Pengajar : Pengajar atau instruktur yang bertanggung jawab untuk memberikan pelatihan kasir kepada siswa atau karyawan baru.

1. Analisis Kebutuhan Fungsional

Kebutuhan fungsional meliputi kebutuhan yang terkait langsung dengan sistem VR simulasi kasir. Kebutuhan fungsional dari aplikasi adalah:

* + - 1. Aplikasi dapat mensimulasikan gambaran konsumen datang ke kasir dan berinteraksi dengan kasir berupa datang ke toko, berkeliling di toko, memberikan gambar, dan memberikan uang.
      2. Aplikasi dapat mensimulasikan proses *scan* barang.
      3. Aplikasi dapat mensimulasikan proses interaksi dengan komputer di kasir untuk input harga dan barang.
      4. Aplikasi dapat mensimulasikan proses penyimpanan uang dan pemberian uang ke konsumen.
      5. Aplikasi dapat mensimulasikan proses pengemasan barang.

1. Analisis Kebutuhan Non-Fungsional

Kebutuhan non-fungsional meliputi analisis kebutuhan peforma, keamanan, portabilitas, *usability,* realibilitas, *maintanibility, compatibility,* perangkat lunak, dan perangkat keras yang digunakan dalam proses pengembangan sistem VR simulasi kasir. Analisis disampaikan sebagai berikut.

* 1. Kebutuhan Performa

1. Aplikasi harus berjalan dengan frame rate minimal 60 FPS agar pengalaman pengguna di VR terasa nyaman dan tidak menimbulkan motion sickness.
2. Waktu loading scene tidak lebih dari 5 detik untuk menjaga interaktivitas dan kelancaran simulasi.
3. Ukuran aplikasi tidak melebihi 1 GB sehingga masih dapat diinstal pada perangkat dengan spesifikasi menengah.
   1. Kebutuhan Keamanan
4. Aplikasi tidak membutuhkan akses berlebihan ke perangkat (misalnya hanya mengakses input VR dan *controller*).
   1. Kebutuhan *Usability* (Kemudahan Penggunaan)
5. Antarmuka pengguna dibuat sederhana, dengan ikon dan instruksi yang mudah dipahami untuk meminimalisasi kebingungan.
6. Pengguna baru harus dapat memahami cara mengoperasikan aplikasi dalam waktu kurang dari 5 menit.
7. Navigasi dalam VR harus intuitif, memanfaatkan kontroler VR sebagai alat interaksi utama.
   1. Kebutuhan Reliabilitas
8. Aplikasi harus stabil dan tidak crash selama minimal 30 menit penggunaan terus-menerus.
9. Interaksi seperti mengambil barang, memasukkan ke kasir, dan melakukan transaksi harus selalu memberikan respon yang konsisten.
   1. Kebutuhan Maintainability
10. Struktur kode dibuat modular agar dapat dikembangkan lebih lanjut, misalnya menambah jenis barang, prosedur kasir baru, atau mode pelatihan tambahan.
11. Dokumentasi kode dan rancangan sistem disiapkan untuk memudahkan proses pemeliharaan.
    1. Kebutuhan Compatibility
12. Sistem kompatibel dengan headset VR standalone maupun berbasis PC.
13. Sistem dapat menyesuaikan resolusi grafis sesuai dengan kemampuan perangkat yang digunakan.
    1. Perangkat Lunak

Perangkat lunak yang digunakan untuk pengembangan aplikasi adalah sebagai berikut :

Unity 6000.0.39f1, aplikasi *game engine.*

Visual Studio Code, yaitu aplikasi yang digunakan untuk melakukan pemrograman dalam bahasa pemrograman C#.

Meta XR, yaitu *package* yang digunakan untuk implementasi fitur VR pada aplikasi.

* 1. Perangkat Keras

Kebutuhan perangkat keras meliputi perangkat keras yang digunakan untuk mengembangkan dan menjalankan aplikasi. Perangkat keras untuk pengembangan aplikasi berupa laptop dengan spesifikasi yaitu:

1. Manufaktur: Asus ROG Zephyrus G14
2. Processor: AMD Ryzen 7 4800HS with Radeon Graphics 2.90 GHz
3. RAM: 24.0 GB
4. Sistem Operasi: Windows 10

Perangkat keras yang digunakan untuk menjalankan aplikasi berupa *Virtual Reality Headset* dengan spesifikasi:

Manufaktur : Meta Quest 2

Resolusi : 1832 x 1920 per mata

Prosesor : Qualcomm Snapdragon XR2

RAM : 6 GB

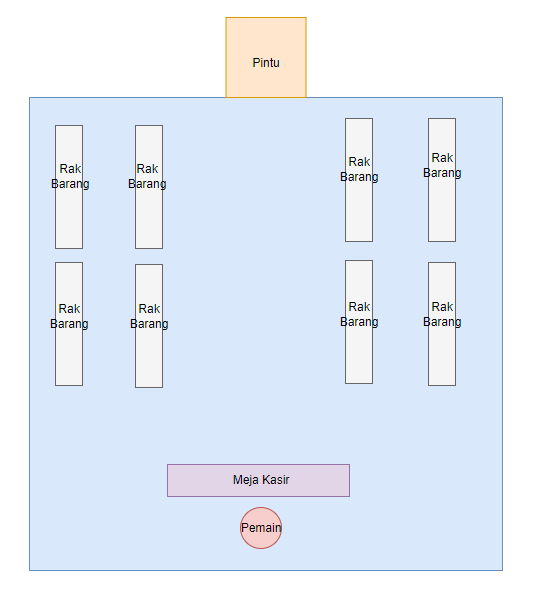
Penyimpanan : 256 GB

Controller : *Two touch controllers*

1. Pre-production

Tahap ini meliputi desain perangkat lunak yang akan dibuat. Desain perangkat lunak yang dibuar berupa rancangan umum aplikasi, *storyline, storyboard, core mechanics, core loop,* aset seni,dan *technical aspects.*

1. Rancangan Umum Aplikasi



Gambar 3. Peta area lingkungan *game*

Aplikasi ini memiliki tujuan untuk melayani pelanggan yang ada sebagai kasir berdasarkan waktu dan jumlah tertentu. Ketika waktu sudah habis, maka pelanggan tidak lagi melayani konsumen dan konsumen akan terus bedatangan sebelum waktu habis. Di akhir akan ada ulasan akan seberapa baik peforma dari pengguna menggunakan skor.

Gambar 3.1 merupakan peta dari permainan. Game ini memiliki setting lingkungan di sebuah minimarket, dimana pemain akan muncul pertama kali di belakang meja kasir dan menunggu pembeli menghampiri. Pembeli akan mengelilingi toko untuk sementara dan akan menghampiri meja kasir pada waktu tertentu.

1. *Storyline*

*Storyline* pada aplikasi ini dirancang dengan memperhatikan prosedur pelayanan kasir pada umumnya. Contoh kasus yang diambil adalah pelayanan kasir pada toko retail Famili yang berlokasi di Bekasi. Berikut merupakan prosedur pelayanan kasir:

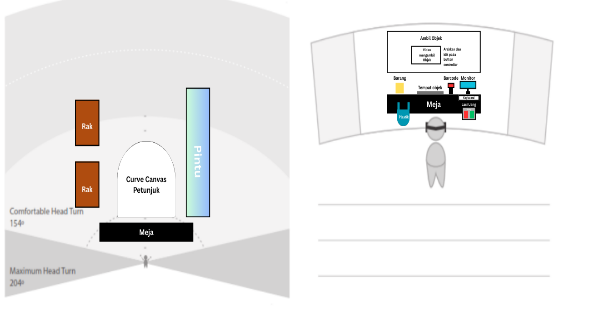
1. Konsumen menghampiri meja kasir dan memberikan produk yang akan dibeli.
2. Kasir melakukan *scan* pada barcode seluruh barang yang diberikan konsumen.
3. Kasir menetapkan harga sesuai barang yang di-*scan*.
4. Konsumen memberikan uang pada kasir.
5. Kasir menyimpan uang tersebut.
6. Kasir menempatkan barang pada plastik dan memberikan kembalian apabila ada.

Berdasarkan prosedur tersebut, maka penulis merancang *storyline* pada aplikasi VR simulasi kasir. Pertama, kasir akan menunggu konsumen datang ke dalam toko. Konsumen akan datang dan berkeliling ke toko sebentar sebelum ke meja kasir. Ketika, konsumen datang ke kasir, konsumen akan mengantri di depan meja kasir untuk menunggu proses transaksi. Kemudian, kasir akan melakukan *scan* pada seluruh barang yang diberikan, menetapkan harga, mengambil uang dari konsumen, kemudian mengemas barang dan memberikan kembalian.

1. *Storyboard*

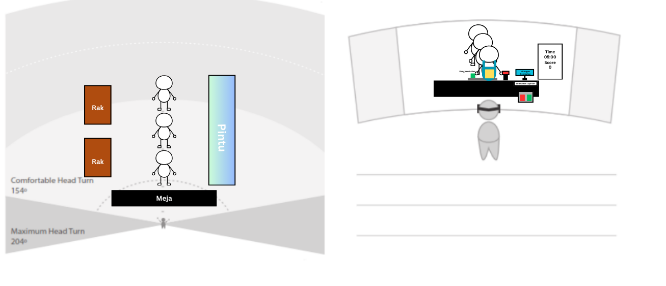
Storyboard meliputi gambaran yang terjadi pada aplikasi. Storyboard digambarkan menjadi 12 tahapan yaitu; halaman menu utama, halaman memilih permainan, halaman tutorial, halaman simulasi satu, halaman simulasi dua, halaman simulasi tiga, halaman simulasi empat, halaman simulasi lima, halaman simulasi enam, halaman simulasi tujuh, halaman simulasi delapan, halaman simulasi sembilan. Tabel 3.1 berisi gambar, judul, situasi, dan deskripsi situasi.

Terdapat dua mode permainan yang dapat dimainkan oleh pemain, yakni mode *tutorial* dan mode permainan utama. Mode tutorial memberikan petunjuk cara bermain pada aplikasi VR berupa tombol yang harus di klik dan alur yang harus diikuti oleh pemain. Mode permainan utama adalah mode simulasi utama dengan batasan waktu.Dua mode ini memiliki alur Gambaran pada *storyboard* yang berbeda. *Storyboard* mode *tutorial* ditunjukkan pada Gambar 3.2.



Gambar 3. *Storyboard* Mode Tutorial

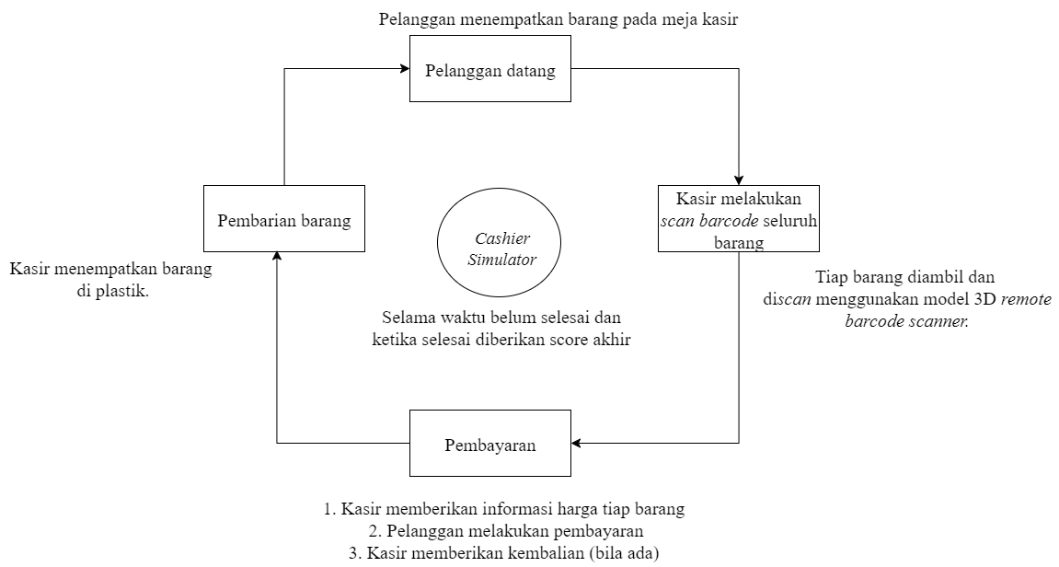
Mode *tutorial* berisikan canvas petunjuk di depan pemain yang memberikan video petunjuk yang harus dilakukan oleh pemain untuk lebih mudah memahami hal hal apa saja yang bisa dan harus dilakukan pada aplikasi. Pemain diharuskan melakukan aktivitas sesuai dengan paduan pada kanvas petunjuk untuk melanjutkan proses tutorial. Selain mode tutorial, terdapat mode perminan utama, gambaran mode permainan utama terdapat pada storyboard pada Gambar 3.3.



Gambar 3. *Storyboard* Mode Permainan Utama

Mode permainan utama memberikan simulasi pelayanan kasir dengan sekelompok konsumen satu persatu. Konsumen akan melakukan penjelajahan toko, mengantri diantrean, dan melakukan proses transaksi di kasir. Pada mode ini, pengguna akan diberikan tantangan berupa seberapa banyak jumlah ketepatan pelayanan konsumen dan banyaknya konsumen yang dapat dilayani dengan waktu yang terbatas. Proses pelayanan berupa *scan* barang, input uang yang diberikan, pengemasan barang, pemberian uang kembalian, dan pemberian barang yang telah dikemas.

1. *Core Loop*

Core loop dari permainan meliputi alur permainan dalam penggunaan aplikasi. Core loop dimulai dari kedatangan pelanggan sampai proses akhir pelayanan konsumen. Pelayanan akan terus dapat dilakukan sampai waktu habis. Ketika pemain berhasil melakukan pelayanan tanpa kesalahan, maka skor pemainakan bertambah. Dalam penelitian ini *core loop* ditunjukkan pada Gambar 3.4.

Gambar 3. *Core Loop* Aplikasi

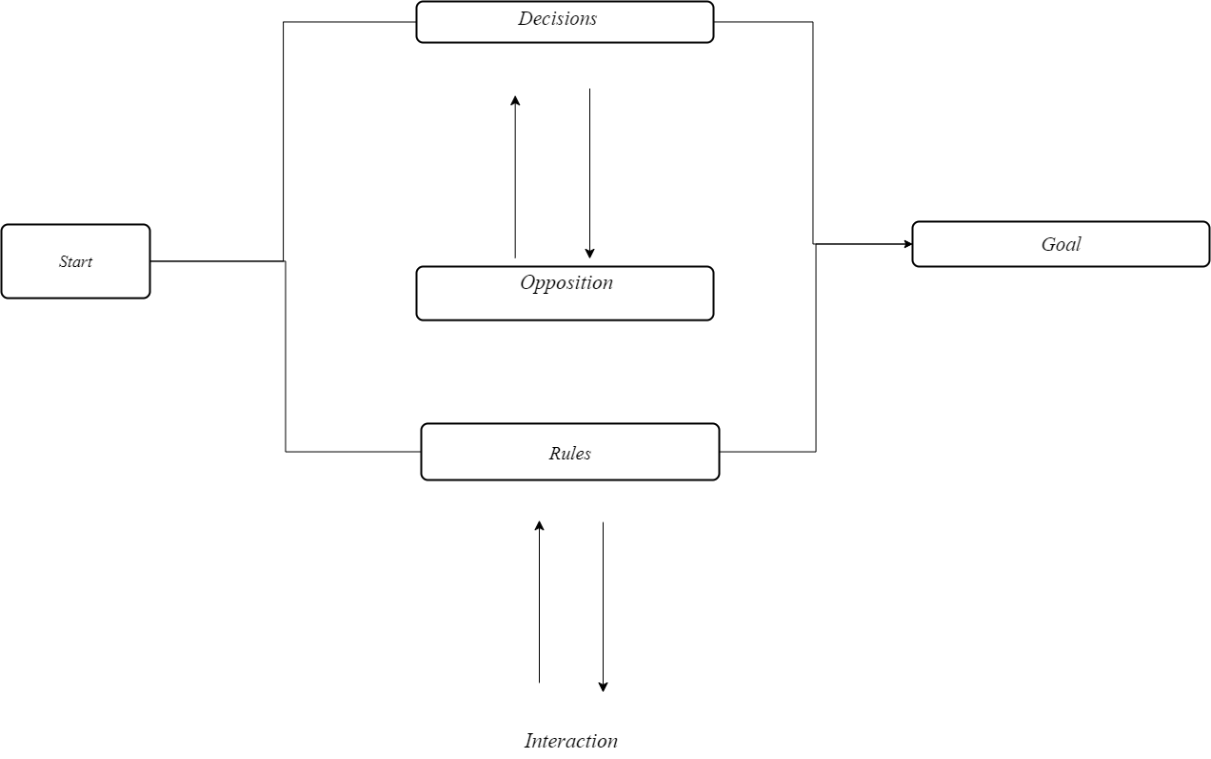
1. *Core Mechanics*

*Core mechanics* digambarkan menggunakan Librande Stone Framework. *Core mechanics* dimulai dari mekanik yang akan pengguna interaksikan pada awal memulai permainan. Berdasarkan Librande Stone Framework *Core Mechanics* dibagi menjadi *start, decision, opposition, goal,* dan *rules*. Setiap komponen tersebut saling memiliki interaksi yang berkaitan satu dan lainnya. Rincian tersebut tertera pada Tabel 3.1.

Tabel 3. Rincian Komponen *Librande Stone*

| No | Komponen | Keterangan |
| --- | --- | --- |
| 1 | *Start* (Mulai) | Persiapan *scene* atau aplikasi pemain di depan meja  dan ada batas waktu  permainan. Barang yang dapat diinteraksikan adalah kantong plastic, lemari uang, *barcode scanner,* komputer, dan barang pelanggan. Kasir akan menunggu sampai pelanggan datang dan membawa kumpulan barang. |
| 2 | *Decisions* (Keputusan) | Terdapat dua Keputusan yang bisa dibuat yaitu menunggu pelanggan datang dan melayani pelanggan berdasarkan langkah barang ditempatkan di meja kasir oleh konsumen, semua barang di*scan*  dengan alat *scan* dan total harga akan muncul dimonitor, pemain menginformasikan harga barang kepada konsumen, konsumen melakukan pembayaran, pemain memberikan kembalian, pemain menempatkan barang pada plastik |
| 3 | *Opposition* (Oposisi) | Rintangan dalam konteks aplikasi dapat dilihat menjadi dua aspek, aspek pelanggan yakni pengguna diharuskan memastikan semua step pelayanan dilaksanakan secara sistematis dan  berurut kemudian memastikan semua proses selesai sebelum bar kesabaran habis. Selain itu diluar pelanggan rintangan dapat dilihat yakni memastikan skor besar sesuai dengan waktu yang diberikan. |
| 3 | *Rules*(Peraturan) | Batas waktu dapat diatur diawal sesuai ketentuan pengguna, Score per pelanggan :  ketersesuain dengan sop sebelum waktu habis, pelayanan pelanggan :  *scan* barang, dimana barang satu persatu diarahkan ke barcode dan di *scan* dengan alat, menyimpan uang dari pelanggan ke tempat penyimpanan uang, mengemas barang dalam plastik beserta menempatkannya di meja kasir, dan memberikan kembalian apabila ada, dan *input* uang pelanggan ke monitor. Masing masing memberikan 1 poin. |
| 4 | *Interaction* (Interaksi) | Terdapat interaksi VR menggunakan *remote controller* |
| 5 | *Goal* (Tujuan) | *Main Goal*(Tujuan Utama) yaitu mendapatkan skor sebesar besarnya berdasarkan waktu yang ada dan *sub goal (*Tujuan Cadangan) yaitu mengikuti urutan pelayanan sesuai dengan urutan yang benar dan menggunakan alat kasir dengan baik. |

Komponen – komponen tersebut kemudian disatukan menjadi satu diagram yang membentuk diagram *Core mechanic* berdasarkan Librande Stone *Framework*. Gambar 3.8 adalah diagram *Core Mechanic* yang digabungkan.



Gambar 3. Penggabungan Core Mechanic.

1. Aset Seni

Terdapat beberapa aset seni yang digunakan untuk menguatkan kesan imersif dalam permainan. Terdapat tiga jenis aset, aset model tiga dimensi, aset *sound effect* (SFX), dan aset animasi. Berikut adalah kumpulan aset seni:

1. Aset model tiga dimensi meliputi model file FBX tiga dimensi yang akan diimplementasikan pada *game engine.* Tabel 3.2 adalah daftar aset model tiga dimensi yang dibutuhkan pada aplikasi.

Tabel 3. Daftar Aset Model Tiga Dimensi

| Kode Asset | Nama Objek | Pembuat | Pranalar | Unduh | 3D Objek |
| --- | --- | --- | --- | --- | --- |
| M00 | retailstore | JudeFelix | https://www.cgtrader.com/free-3d-models/exterior/cityscape/convenience-store-f3032ffa-c5e7-4af4-af97-d9c912c7718a | Februari 2025 | Convenience Store  Free low-poly 3D model_4 |
| M01 | idr\_100k | Agung Ade Yulianto | https://skfb.ly/oCSNA | Februari 2025 | A close-up of a currency  AI-generated content may be incorrect. |
| M02 | Idr\_50k | Agung Ade Yulianto | https://skfb.ly/oCSNE | Februari 2025 | A close-up of a currency note  AI-generated content may be incorrect. |
| M03 | Idr\_10k | Agung Ade Yulianto | https://skfb.ly/oCUFJ | Februari 2025 | A close-up of a currency note  AI-generated content may be incorrect. |
| M04 | Idr\_5k | Agung Ade Yulianto | https://skfb.ly/oCUBr | Februari 2025 | A close up of a currency  AI-generated content may be incorrect. |
| M05 | Plastic\_bag | zehdmaking | https://www.cgtrader.com/free-3d-models/industrial/other/plastic-e-papers-bags | Februari 2025 | A screen shot of a device  AI-generated content may be incorrect. |
| M06 | Product\_chips | asleeper | https://www.cgtrader.com/free-3d-models/food/miscellaneous/food-packaging-chips | Februari 2025 | A yellow bag of potato chips  AI-generated content may be incorrect. |
| M07 | Product\_energydrink | vysiondesign | https://www.cgtrader.com/free-3d-models/food/beverage/effect-energydrink-can-model | Februari 2025 | A can of energy drink  AI-generated content may be incorrect. |
| M08 | Product\_coke | alpborauyar3D | https://www.cgtrader.com/free-3d-models/food/beverage/coca-cola-glass-bottle-3d-model | Februari 2025 | A white bottle with red label  AI-generated content may be incorrect. |
| M09 | Product\_cereal | leounderwood | https://www.cgtrader.com/free-3d-models/food/miscellaneous/cereal-box-mysteri-os | Februari 2025 | A box of food on a white background  AI-generated content may be incorrect. |
| M10 | Consumen\_man1 | Adobe | maximo.com | Februari 2025 | A person with his arms out  AI-generated content may be incorrect. |
| M11 | Consumen\_man2 | Adobe | maximo.com | Februari 2025 | A person wearing a black uniform and sunglasses  AI-generated content may be incorrect. |
| M12 | Consumen\_woman1 | Adobe | maximo.com | Februari 2025 | A person with her arms out  AI-generated content may be incorrect. |
| M13 | Consumen\_woman2 | Adobe | maximo.com | Februari 2025 | A person with her arms out  AI-generated content may be incorrect. |
| M14 | Cashier\_table | JudeFelix | https://www.cgtrader.com/free-3d-models/exterior/cityscape/convenience-store-f3032ffa-c5e7-4af4-af97-d9c912c7718a | Februari 2025 |  |
| M14 | Barcode\_remote | JudeFelix | https://www.cgtrader.com/free-3d-models/exterior/cityscape/convenience-store-f3032ffa-c5e7-4af4-af97-d9c912c7718a | Februari 2025 |  |

1. Aset SFX meliputi aset suara baik untuk suara efek setelah aksi atau musik utama ketika aplikasi sedang dimainkan. Tabel 3.3 adalah daftar aset SFX yang dibutuhkan aplikasi.

Tabel 3. Daftar Aset SFX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kode Aset | Nama Aset | Pembuat | Pranalar | Unduh |
| S00 | Suara tombol menu | Not\_Amasingrock | https://pixabay.com/id/sound-effects/video-game-menu-click-sounds-148373/ | Februari 2025 |
| S01 | BGM Ketika bermain | nengjemping | https://pixabay.com/id/music/ketukan-lofi-music-background-hits-352712/ | Februari 2025 |

1. Aset animasi meliputi animasi yang digunakan oleh model tiga dimensi selama permainan berlangsung. Tabel 3.4 adalah daftar aset animasi yang dibutuhkan aplikasi.

Tabel 3. Daftar Aset Animasi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kode Aset | Nama Aset | Pembuat | Pranalar | Unduh |
| A00 | Animasi tunggu M10 | Adobe | maximo.com | Februari 2025 |
| A01 | Animasi jalan M10 | Adobe | maximo.com | Februari 2025 |
| A02 | Animasi tunggu M11 | Adobe | maximo.com | Februari 2025 |
| A03 | Animasi jalan M11 | Adobe | maximo.com | Februari 2025 |
| A04 | Animasi tunggu M12 | Adobe | maximo.com | Februari 2025 |
| A05 | Animasi jalan M12 | Adobe | maximo.com | Februari 2025 |
| A06 | Animasi tunggu M13 | Adobe | maximo.com | Februari 2025 |
| A07 | Animasi jalan M13 | Adobe | maximo.com | Februari 2025 |

1. *Technical Aspects*

*Technical aspects* digambarkan menggunakan UML (*Unified Modeling Language*) untuk perancangan sistem menggunakan *use case diagram, activity diagram,* dan *flowchart.*

1. *Use Case Diagram*

*Use Case Diagram* digunakan untuk menggambarkan interaksi *user* ketika pertama kali menggunakan aplikasi dan kemungkinan fitur yang akan digunakan oleh user. Berikut *use case diagram* dari permainan yang akan dikembangkan pada Gambar 3.9.



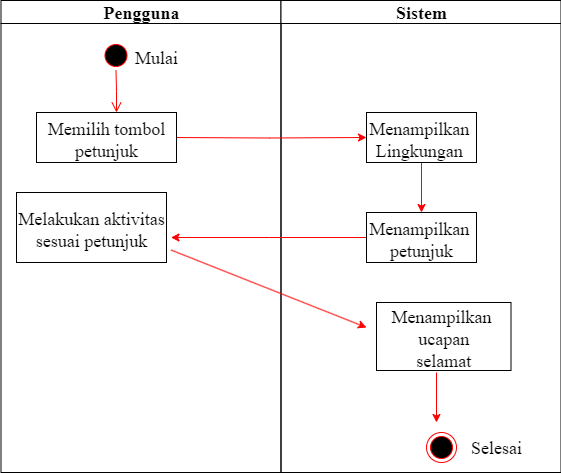
Gambar 3. *Use case diagram* VR Pelatihan Kasir

1. *Activity Diagram*

*Activity diagram* menggambarkan alur kerja atau proses dalam aplikasi permainan. Berikut adalah *activity diagram* dari aplikasi yang dikembangkan.

* 1. *Activity Diagram* Petunjuk Aplikasi

*Activity Diagram* petunjuk aplikasi menjelaskan alur kerja aplikasi ketika memberikan arahan atau tata cara menggunakan aplikasi dan proses kerja kasir. Permainan akan menampilkan visual untuk menjelaskan tata cara suatu aktivitas dilakukan. Ketika aktivitas terkait selesai dilaksanakan, maka pemain akan diarahkan untuk melaksanakan aktivitas berikutnya. Hal ini dilakukan untuk menjelaskan semua interaksi dan tata cara bermain yang jelas bagi pemain. Gambar 3.10 adalah *Activity Diagram* Petunjuk Aplikasi.

**

Gambar 3. *Activity Diagram* Petunjuk Aplikasi

* 1. *Activity Diagram* Utama

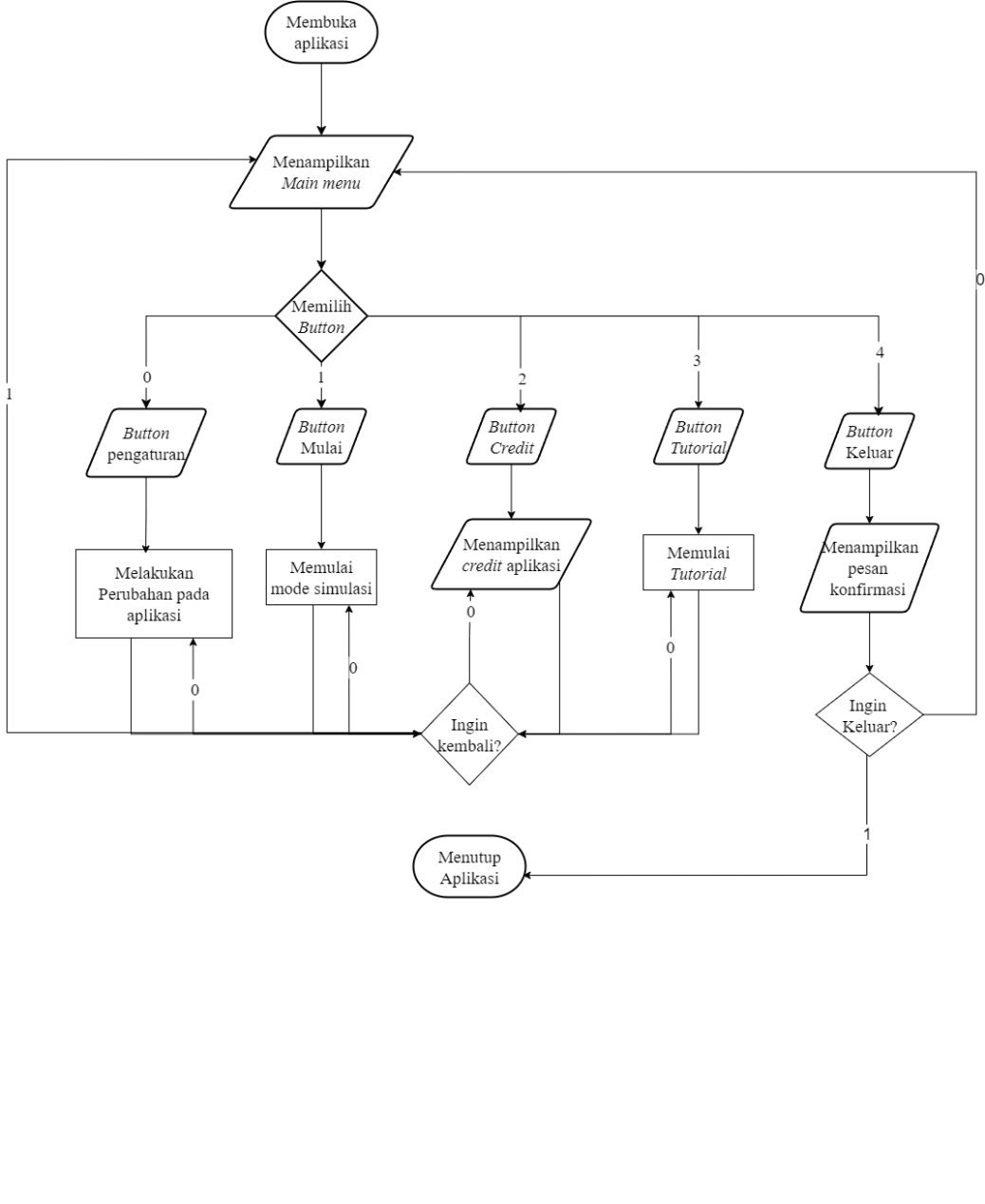
*Activity Diagram* Utama berisikan diagram yang menjelaskan alur permainan pada bagian permainan utama ketika sudah terdapat konsumen dan target untuk melayani tiap konsumen. Gambar 3.11 menampilkan *activity diagram* utama.



Gambar 3. *Activity Diagram* Utama

1. *Flowchart*

*Flowchart* berisikan alur yang dilalui oleh pengguna mulai dari membuka aplikasi sampai menutup aplikasi. Pengguna dapat memilih beberapa tombol yang akan menghasilkan perilaku yang berbeda pada aplikasi. *Flowchart* menggambarkan alur aplikasi berdasarkan tombol yang diklik pengguna. Terdapat tiga tombol utama pengaturan, bermain, *credit,* dan keluar Gambar 3.12 adalah *flowchart* aplikasi.



Gambar 3. *Flowchart* Aplikasi

# **BAB IV**

**HASIL DAN PEMBAHASAN**

1. Production

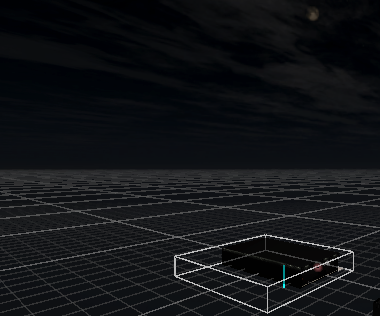
Tahap ini dilaksanakan dengan proses pembuatan kode progam dan implementasi aset beserta integrasi antara kode program dan aset. Implementasi yang dilakukan meliputi pembuatan lingkungan, pembuatan UI, penambahan efek visual, penambahan suara, dan penambahan video. Tahapan ini juga meliputi implementasi interasksi VR dengan lingkungan virtual.

1. Aset dan Lingkungan

Pembuatan lingkungan pada aplikasi diimplementasikan sesuai pada perencanaan pada sebelumnya. Aset dan lingkungan yang digunakan diambil dari internet aset dimanfaatkan untuk lingkungan, karakter, dan objek untuk memberikan kesan imersif yang lebih. Aset yang diambil adalah aset 3D yang *open source* sehingga memiliki hak cipta yang dapat digunakan.

1. Pembuatan Lingkungan Toko

Pembuatan lingkungan toko diawali dengan menambahkan *skybox* malam untuk memberikan kesan realistis. Selain itu, *skybox* berperan penting dalam pantulan warna dan pencahayaan lingkungan toko yang akan menjadi latar utama permainan.



Gambar 4. Tampilan Gedung Toko dan *Skybox*

Setelah itu, lingkungan toko dapat dimasukkan untuk dan dilakukan penyesuaian pencahayaanya. Produk yang terdapat pada rak lingkungan toko dikurangin dari sumber awal aset untuk meningkatkan peforma aplikasi pada *Headset* VR. Penyesuaian meja kasir juga dilakukan untuk meningkatkan kenyamanan pemain.



Gambar 4. Lingkungan Toko dari dalam

1. Pembuatan Lingkungan *Scene* Menu

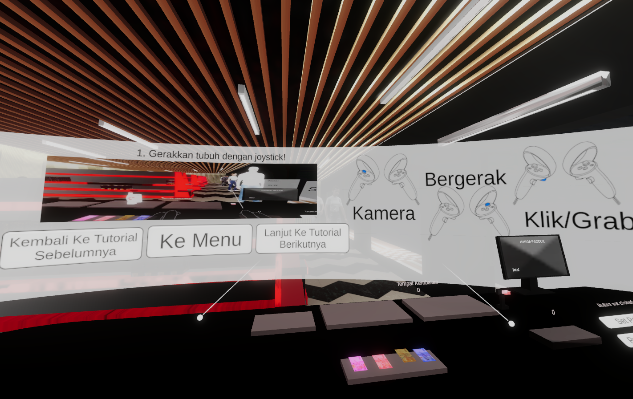
Scene menu berada pada lingkungan toko yang disesuikan dengan menghilangkan beberapa objek seperti rak barang agar memfokuskan pemain kepada button UI. *Ray* pada controller pemain diatur agar lebih panjang untuk menjangkau *button* pada menu sehingga pemain tidak harus bergerak maju mendekatkan ke *button* pada menu untuk menekan menu. *Main Menu* berisikan lima *button, button tutorial* yang mengalihkan langsung ke *scene tutorial, button bermain* untuk beralih ke *scene simulasi utama*, dan keluar untuk keluar dari aplikasi.



Gambar 4. Tampilan *Main Menu*

1. Pembuatan Lingkungan *Scene* Tutorial

Lingkungan *scene tutorial* menempatkan pengguna dibelakang meja kasir dengan perintah dan simulasi video. Pemain dapat bebas melanjutkan atau kembali pada perintah sebelumnya dan melihat video yang ada. *Scene tutorial* juga memberikan simulasi konsumen yang datang. Konsumen akan terus datang dan memberikan barang untuk dilayani sampai pemain merasa cukup. Apabila pemain sudah merasa cukup, pemain dapat memilih *button* Kembali ke main menu untuk kembali ke *main menu.*

**

Gambar 4. Tampilan Tutorial

1. Pembuatan Lingkungan Scene *Main game*

Scene *main game* menempatkan pemain di belakang meja kasir. Meja kasir disesuaikan dengan ditempatkannya beberapa UI dan objek yang akan digunakan oleh pemain seperti monitor, keyboard, UI, *barcode scanner*, dan plastik. Pemain akan ditempatkan persis dibelakang monitor dan meja kasir akan diberikan *collider* pembatas sehingga pemain tidak bisa keluar dari lingkungan kasir. Meja kasir juga disesuaikan dengan tinggi pemain pada VR *headset* agar simulasi kasir dikerjakan dengan lebih nyaman. Pada *scene main game*, terdapat batasan waktu dan informasi score pengguna.



Gambar 4. Tampilan *Main Game*

1. Pembuatan Efek Suara

Terdapat dua jenis suara yang ditambahkan pada permainan. Suara *background* dan suara *sound effect.* Suara *background* akan terus dimainkan tanpa berhenti selama permainan dimainkan sedangkan, sound effect hanya akan muncul ketika pemain berhasil memilih *button*. Kedua fungsionalitas ini memiliki implementasi yang berbeda.

*Background Music* (BGM)

Langkah pertama dalam menambahkan BGM dilakukan pada *scene* *main menu* karena *scene* ini adalah *scene* yang akan pertama kali pengguna masuki ketika pertama kali membuka aplikasi. Sehingga, implementasi BGM yang akan terus dimainkan dari awal permainan dilaksanakan pada *scene* ini. Pada *scene* ini akan dibuat *gameobject* baru Bernama BGMManager yang akan berisikan implementasi BGM. Pada BGMManager akan ditambahkan dua *game component*, *audio source* dan *script BGMManager.cs.* *Game component* *audio source* berisikan pengaturan untuk sumber audio yang akan diputar. Sendangkan *script BGMManager.cs* berisikan logika untuk memastikan suara akan terus dimainkan ketika berganti *scene.*

*A screenshot of a computer

AI-generated content may be incorrect.*

Gambar 4. Pengaturan *Game Component* Audio *Source* dan *Script BGM Manager*

Berikut adalah *script BGMManager.cs*

|  |
| --- |
| using UnityEngine;  public class BGMManager : MonoBehaviour  {      public static BGMManager Instance;      public AudioSource bgmSource;      private void Awake()      {  *// Pastikan hanya ada satu BGMManager di semua scene*          if (Instance != null && Instance != this)          {              Destroy(gameObject);              return;          }          Instance = this;          DontDestroyOnLoad(gameObject); *// BGM tidak hancur ketika ganti scene*      }      public void PlayBGM(AudioClip clip, bool loop = true)      {          if (bgmSource.clip == clip && bgmSource.isPlaying) return; *// jangan restart kalau sama*          bgmSource.clip = clip;          bgmSource.loop = loop;          bgmSource.Play();      }      public void StopBGM()      {          bgmSource.Stop();      }      public void SetVolume(float volume)      {          bgmSource.volume = Mathf.Clamp01(volume);      }  } |

*Sound effect* dan Perpindahan *scene*

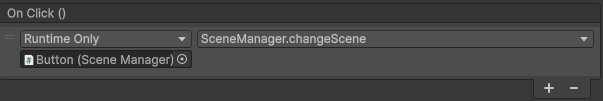
Implementasi *Sound Effect* dilakukan pada *script UISoundManager.cs*. Fungsi AttachSFX() pada *script UISoundManager.cs* akan mencari seluruh *button* yang ada di *scene* yang sedang dibuka. Setelah itu, akan ditambahkan fungsi PlaySFX untuk memainkan SFX setelah di click. Berikut adalah implementasi AttachSFX dan PlaySFX() pada *script UISoundManager.cs.*

|  |
| --- |
| void AttachSFX()  {  Button[] buttons = FindObjectsOfType<Button>(true);  foreach (Button btn in buttons)  {  // cegah double SFX  btn.onClick.RemoveListener(PlaySFX);  // tambahkan SFX  btn.onClick.AddListener(PlaySFX);  }  }  void PlaySFX()  {  if (clickSFX != null)  audioSource.PlayOneShot(clickSFX, volume);  } |

Untuk melakukan perpindahan pada *scene,* digunakan *script SceneManager.cs* yang ditempatkan pada *gameobject button. Script* ini memiliki *field* input teks yang berisikan nama *scene* yang akan dibuka oleh aplikasi setelah melakukan klik pada *button.*

|  |
| --- |
| using UnityEngine;  using UnityEngine.SceneManagement;  public class SceneManager : MonoBehaviour  {  *// Start is called once before the first execution of Update after the MonoBehaviour is created*      public string sceneName;      public void changeScene()      {          UnityEngine.SceneManagement.SceneManager.LoadScene(sceneName);      }      public void quitApp()      {          Application.Quit();      }  } |

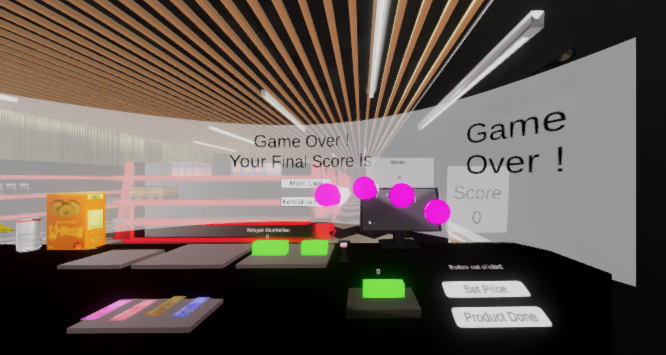
Fungsi changeScene() pada *SceneManager.cs* akan ditempatkan pada *gamecomponent* pada button di bagian *OnClick()* yang berfungsi untuk mendefinisikan hal apa saja yang harus aplikasi lakukan setelah klik *button*. Dalam konteks ini, memainkan suara efek menekan tombol dan berpindah *scene*. Gambar 4.7 adalah tampilan *game component* OnClick() pada *button* yang memiliki fungsi untuk berpindah *scene.*



Gambar 4. Tampilan *Game Component* *Button* Pindah *Scene*

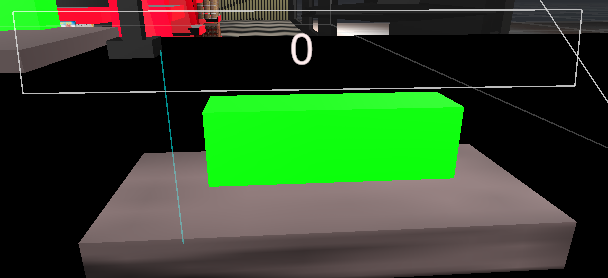
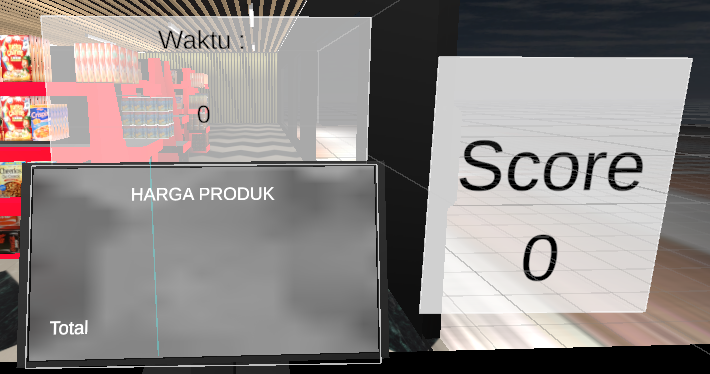
1. Pembuatan *User Interface*

Tedapat dua jenis UI yang digunakan pada aplikasi. UI yang memiliki tampilan *canvas* melengkung dan datar. UI dengan tampilan *canvas* melengkung digunakan untuk memberikan fungsional penglihatan keliling dan memastikan *user* melihat UI tersebut dimanapun posisi user. UI melengkung di implementasikan pada *canvas*  informasi tutorial, *main menu,* dan *game over.* Selain itu, terdapat UI datar yang tidak berbentuk melengkung tetapi *flat*/datar. UI datar berfungsi untuk memberikan informasi yang tidak begitu banyak, seperti *button gameplay, monitor, kalkulator, score,* waktu permainan,tempat kembalian, dan penyimpan uang. Gambar 4.8 dan 4.9 adalah kumpulan implementasi *UI Curved* dan *UI Flat.*

**

**

Gambar 4. Kumpulan UI *Curved*

**

Gambar 4. Kumpulan UI Datar

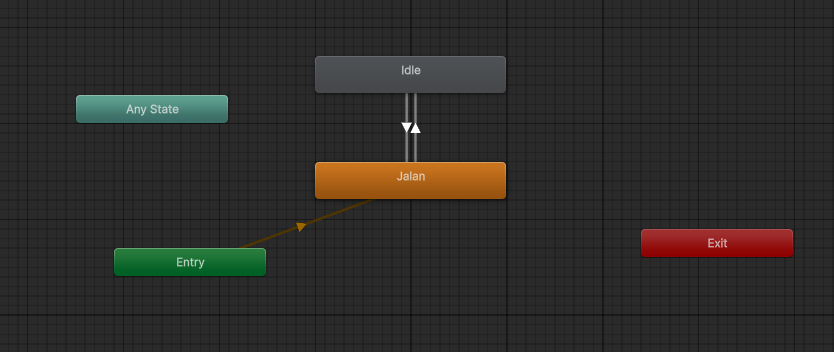
1. Pembuatan Konsumen

Proses penambahan konsumen diawali dengan memindahkan sumber file 3D konsumen dengan ekstensi *FBX* ke dalam Unity. Model 3D dengan ekstensi file *FBX* memiliki dukungan yang cukup baik di unity, sehingga model karakter 3D yang dimasukkan kedalam proyek lebih mudah untuk di kutomisasi. Karakter 3D yang dimasukkan adalah empat buah karakter pembeli. Gambar 4.10 berisikan hasil implementasi menempatkan model tiga dimensi di Unity.



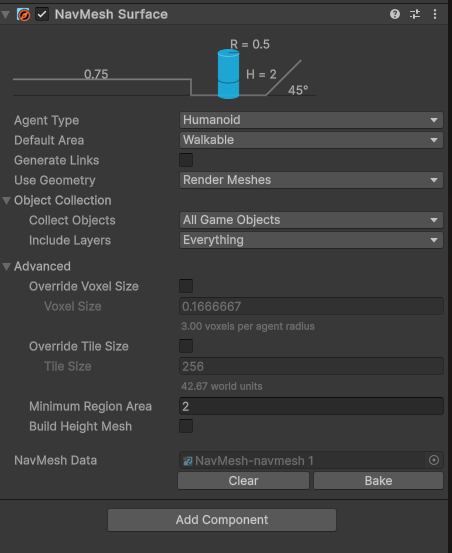
Gambar 4. Tampilan Model 3D Konsumen di Unity

Animasi menjadi aspek yang cukup penting dalam aplikasi karena memberikan kesan simulasi yang lebih realistis. Masing masing empat karakter memiliki empat animasinya masing – masing. Implementasi diawali dengan menambahkan Animator pada masing masing *gameobject* konsumen. Setelah itu, *animation clip* masing masing konsumen dimasukkan kedalam animator. Pada Animator, ditambahkan *state* Jalan dan *idle* dengan animasinya masing masing dengan transisi berisikan variable *Boolean* yang mengecek informasi apakah konsumen harus berjalan atau berdiam. Gambar 4.11 berisikan alur *state* dan transisi tiap animasi.



Gambar 4. *Animator* masing masing konsumen

Implementasi AI dalam konsumen untuk dapat berpindah pada satu tempat ke tempat lainnya menggunakan penambahan komponen *navmesh agent* pada *gameobject* konsumen dan *navmesh surface* pada lingkungan toko. Implementasi ini memungkinkan konsumen dapat bergerak dan berkeliling di lingkungan toko. Gambar 4.12 berisikan *game component* *nav mesh agent* dan *surface*.



Gambar 4. *Game Component nav mesh agent* dan *surface*

Perilaku konsumen dalam berbelanja di implementasikan menggunakan kode program C#. Masing – masing aktivitas yang akan dilakukan oleh konsumen dibagi menjadi beberapa *state*. Tiap *state* ini berisikan implementasi dari aktivitas yang akan dilakukan oleh pelanggan. Dengan menambahkan *state*, konsumen dapat berpindah ke satu aktivitas dan aktivitas lainnya. Kumpulan dari *state* konsumen terdapat pada Tabel 4.1.

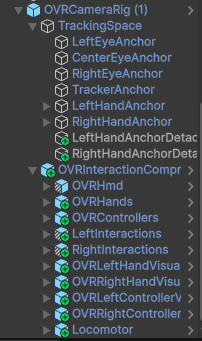
Tabel 4. *State AI*  Konsumen

| Nomor | Nama *state* | Keterangan |
| --- | --- | --- |
| 1 | *Entering* | Konsumen memasuki toko. |
| 2 | *Browsing* | Konsumen menjelajahi toko. |
| 3 | *Queuing* | Konsumen memasukin antrian kasir. |
| 4 | *WaitingInQueue* | Konsumen menunggu di antrian. |
| 5 | *GivingProduct* | Konsumen memberikan barang belanjaan. |
| 6 | *WaitingForPrice* | Konsumen menunggu harga barang. |
| 7 | *Paying* | Konsumen memberikan uang untuk membeli barang. |
| 8 | *Receiving* | Konsumen menerima barang yang sudah ditempatkan diplastik |
| 9 | *Leaving* | Konsumen meninggalkan toko. |

Perpindahan antara satu state dengan state lainnya diatur berdasarkan beberapa hal meliputi interaksi dengan pengguna, waktu yang sudah diatur, dan interaksi dengan karakter atau konsumen lainnya. Setelah konsumen meninggalkan toko atau sampai pada *state leaving*, maka uang dan barang yang diberikan oleh konsumen terkait, akan dihancurkan dari aplikasi agar aplikasi lebih ringan. Implementasi lengkap *script*  AI untuk konsumen pada Lampiran 2. Script tersebut akan menjadi *game component* dengan value yang disesuaikan dengan kebutuhan aktivitas konsumen.

1. Pembuatan Interaksi

Langkah awal dalam pembentukan interaksi pada VR adalah dengan menambahkan *gameobject* OVRCameraRig kedalam proyek. *Game object* ini berisikan implementasi kamera pada VR, pengaturan *remote,* dan pengaturaninteraksi yang akan dilakukan oleh VR. Selain itu, terdapat juga *game object Locomotor* didalam *gameobject* OVRCameraRig yang memberikan pengaturan gerak dan tinggi pengguna. Gambar 4.13 adalah struktur *game object* OVRCameraRig.



Gambar 4. Struktur *Game Object* OVRCameraRig pada Proyek

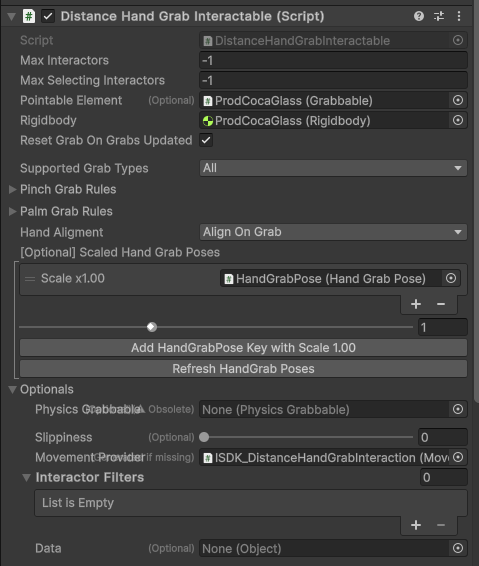
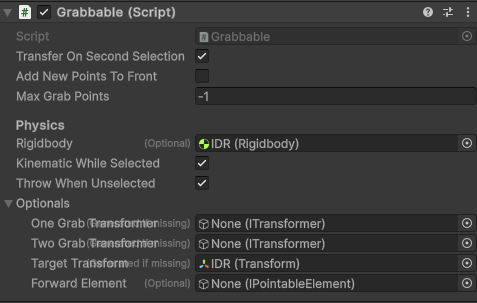
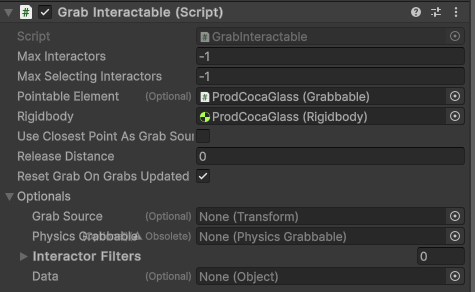
Terdapat tiga interaksi menggunakan VR *headset* dan *remote VR* pada aplikasi yaitu, *Grab interactable, Ray Interactable,* dan *Snap Interactable. Grab Interactable* adalah interaksi yang memungkinkan pengguna memegang objek 3D pada aplikasi menggunakan *remote VR.* Terdapat dua jenis implementasi interaksi *grab* pada aplikasi. Yaitu, *distance grab interactable* dan *grab interactable. Distance grab interactable* memungkinkan pengguna untuk mengambil objek dari jarak jauh sedangkan *grab interactable* mengharuskan pengguna untuk mendekatkan diri ke objek. Interaksi yang kedua adalah *ray interactable* yang digunakan untuk fungsi *UI Click* menggunakan *remote VR. Ray Interactable* adalah interaksi menggunakan visual *laser ray* pada aplikasi. *Snap interactable* memungkinkan untuk memindahkan barang yang telah dipegang untuk secara otomatis berpindah ke titik tertentu ketika selesai dipegang.

Tabel 4. Daftar *Interaksi* VR pada Objek

| Nomor | Nama Barang | Foto Interaksi | Jenis Interaksi |
| --- | --- | --- | --- |
| 1 | idr\_100k |  | *Distance grab* |
| 2 | Idr\_50k |  | *Distance grab* |
| 3 | Idr\_10k |  | *Distance grab* |
| 4 | Idr\_5k |  | *Distance grab* |
| 5 | Plastic\_bag |  | *Distance grab* |
| 6 | Product\_chips |  | *Grab* dan *distance grab* |
| 7 | Product\_energydrink |  | *Grab* dan *distance grab* |
| 8 | Product\_coke |  | *Grab* dan *distance grab* |
| 9 | Product\_cereal |  | *Grab* dan *distance grab* |
| 10 | Barcode\_remote |  | *Grab* dan *distance grab* |
| 11 | *UI button* |  | *Ray Interactable* |

1. *Grab* *Interactable* Dan *Distance Grab Interactable*

Implementasi *grab interactable* dilakukan dengan menambahkan beberapa *game component* yang dapat dipegang oleh pengguna. *Game component* yang pertama adalah *grabbable. Game Component grabbable* memberikan informasi pada aplikasi bahwa *game object* dapat dipegang dan mengontrol objek hasil dari interaksi *grab* yang dilakukan pada objek. Setelah itu, barulah dimasukkan *game component* dengan nama *grab interactable* untuk interaksi sederhana dan *distance grab interactable*. Masing masing dari dua *game component* ini dimasukkan kepada *child game object* dari *game object* masing masing produk, sehingga tiap *produk* memiliki *interaksi grab.* Gambar 4.14 adalah tampilan *game component* ini.

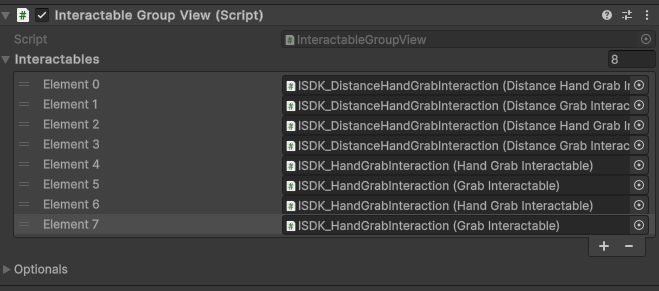
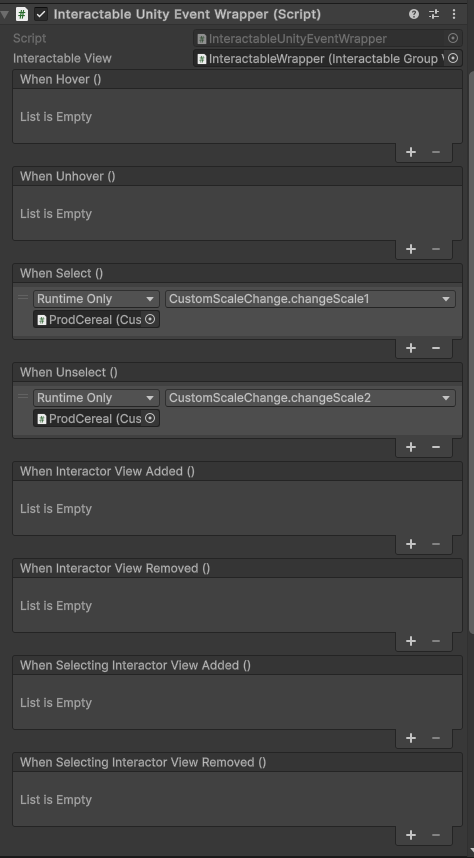
Gambar 4. *Game Component* untuk *Grab* dan *Distance Grab Interactable*

Pada *game component Distance Grab Interactable* dan *Grab Interactable,* terdapat pengaturan *scaled hand grab pose* untuk memberikan pengaturan bentuk tangan ketika memegang objek. Implementasi ini dilakukan dengan memindahkan masing masing posisi X, Y, dan Z pada tiap jari untuk tangan kanan dan kiri. Sehingga, ketika pengguna berhasil memegang objek, bentuk tangan akan otomatis menyesuaikan dengan bentuk objek yang dipegang. Gambar 4.15 adalah contoh objek produk yang sudah diberikan pengaturan pose tangan dan kiri ketika dipegang.



Gambar 4. Pose tangan Ketika Memegang Objek Minuman *Cola*

Implementasi perilaku yang dilakukan ketika pengguna memegang atau melepas objek di implementasikan dengan menambahkan *game component Interactable Unity Event Wrapper*  dan *Interactable Group View. Interactable Group View* digunakan untuk mengelompokkan interaksi pada objek kepada satu kelompok. Kelompok ini kemudian akan di berikan kepada *game component interactable unity event wrapper* untuk memberikan perilaku apa saja yang harus dilakukan kelompok interaksi ini ketika mereka memegang atau tidak memegang objek. Untuk memudahkan pengguna, implementasi ini dilakukan untuk mengecilkan dan membesarkan objek ketika sedang memegang atau tidak memegang objek. Gambar 4.16 merupakan gambar konfigurasi dua *game component* tersebut.



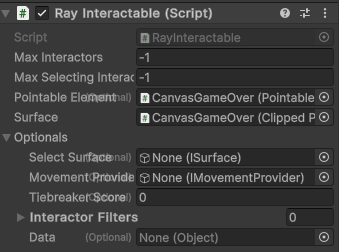
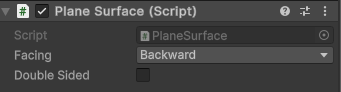
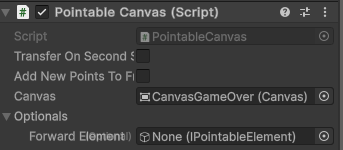
Gambar 4. *Game Component Interactable Unity Event Wrapper* dan *Interactable Group View* untuk interaksi *grab*

Modifikasi ukuran objek dilakukan menggunakan *script* yang membuat dua fungsi untuk ukuran satu dan dua. Fungsi ukuran satu akan dipanggil ketika memegang objek dan fungsi ukuran dua akan dipanggil ketika melepas objek. Implementasi kode terdapat pada script *CustomScaleChange.cs* berikut:

|  |
| --- |
| using Unity.Mathematics;  using UnityEngine;  using UnityEngine.UIElements;  public class CustomScaleChange : MonoBehaviour  {      public Vector3 scale1, scale2;      public Vector3 rotation1, rotation2;      public Transform objectTransform;      void Start()      {          if (objectTransform == null)          {              objectTransform = this.gameObject.GetComponent<Transform>();          }      }      public void changeScale1()      {          objectTransform.localScale = scale1;      }      public void changeScale2()      {          objectTransform.localScale = scale2;      }      public void changeRotation1()      {          objectTransform.localRotation = Quaternion.Euler(rotation1);      }      public void changerotation2()      {          objectTransform.localRotation = Quaternion.Euler(rotation2);      }  } |

1. *Ray* *Interactable*

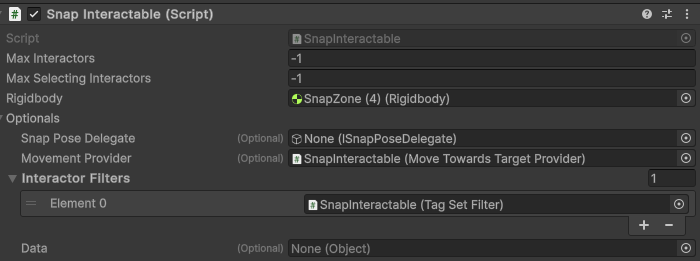
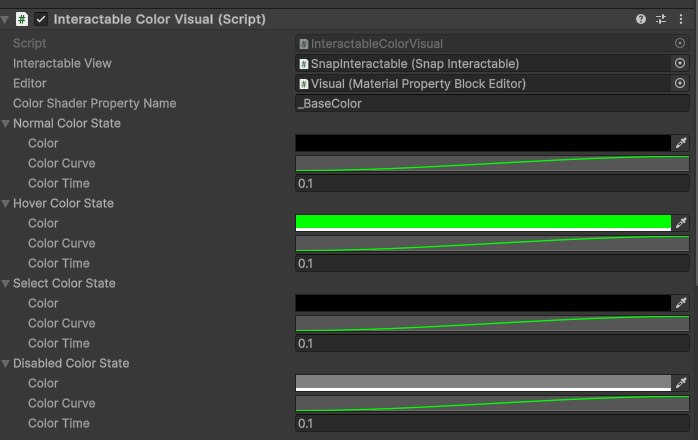
*Ray Interactable* digunakan untuk interaksi *button* pada *canvas. Gameobject* *canvas* ditambahkan *game component pointable canvas* dan *plane surface.* *Pointable canvas* untuk mendefinisikan bahwa *game object ini* dapat di klik sedangkan *place surface* berfungsi untuk memberikan informasi area yang dapat diklik pada *canvas* area mana saja. Terakhir, ditambahkan *game component Ray Interactable* untuk memberikan informasi bahwa *canvas* memiliki interaksi *ray interactable.*

**

Gambar 4. *Game Component Ray Interactable, Pointable Canvas,* dan *Plane Surface*

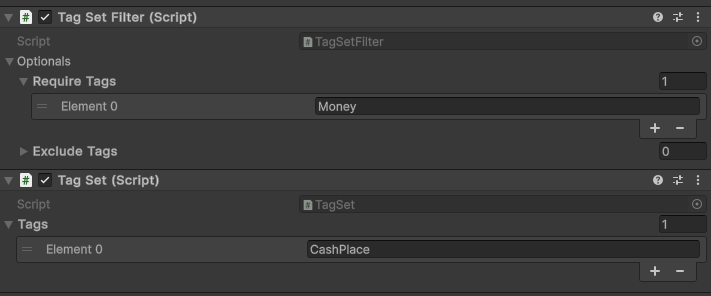
1. *Snap Interactable*

Snap Interactable diimplementasikan dengan menambahkan *game component snap interactable*. *Game Component* ini akan mendeteksi bahwa *game object ini* memiliki perilaku *snap.* Setelah itu, ditambahkan juga *game component interactable color visual* untuk memberikan informasi warna, ketika *game object* sedang di *hover*. Gambar 4.18 adalah implementasi pengaturan masing masing *game component* ini.

**

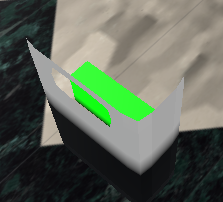
Gambar 4. *Game Component* *Snap Interactable* dan *Interactable Color Visual*

Terdapat juga implementasi *filter* untuk melakukan seleksi objek apa saja yang bisa di tempatkan pada *game object* dengan interaksi *snap.* Hal ini dilakukan dengan menambahkan *game component tag set filter* dan *tag set. Tag set* memberikan informasi *game object* dengan *tag* apa saja yang bisa *snap* di tempat tersebut dan *tag set* memberikan informasi tempat nama *tag* dari tempat *snap* terkait. Implementasi ini dilakukan untuk membedakan tempat kembalian dan tempat penyimpanan uang oleh konseumen. Gambar 4.19 adalah contoh implementasi *filter* untuk penyimpanan uang dari konsumen.



Gambar 4. *Filter* Lokasi *Snap* Uang Konsumen

Terdapat tujuh lokasi *snap interactable.*Tigatempat Penyimpanan objek, tempat penyimpanan uang, tempat kembalian, tempat pemberiak produk yang sudah dikemas, dan lokasi *snap* produk didalam plastik. Masing masing tempat ini memiliki implementasi warna hijau apabila di*hover* dan tidak ada warna apabila tidak di hover. Apabila tidak dihover maka lokasi objek ini akan tidak memiliki warna. Ketika terdapat objek didalamnya, objek akan tersimpan di lokasi tersebut dan objek lain yang ingin di tempakan di lokasi tersebut akan saling tertumpuk dengan objek sebelumnya. Gambar 4.20 adalah masing masing implementasi penempatan tujuh lokasi tersebut.



Gambar 4. Lokasi *snap* Tiga tempat Penyimpanan objek, tempat penyimpanan uang, tempat kembalian, tempat pemberian produk yang sudah dikemas, dan lokasi snap produk didalam plastik.

1. Pembuatan Sistem Kasir

Sistem kasir memiliki beberapa tahapan. Tahapan *scan barcode,* Penetapan harga, penyimpanan dan perhitungan uang yang diberikan konsumen, pemberian uang kembalian dan pengemasan barang. Tiap tahapan ini saling memiliki keterikatan satu sama lain dan diimplementasikan menggunakan *script* C#. Berikut adalah implementasi tahap tahap tersebut.

1. *Scan Barcode*

Implementasi perilaku *barcode* diawali dengan membuat *gameobject* baru berupa laser barcode yang menjadi anak dari *barcode remote.* Hal ini dilakukan agar ketika *barcode remote* digerakkan, maka laser juga akan ikut bergerak. Gamer 4.21 adalah gamber dari *laser barcode remote.*



Gambar 4. Laser *barcode remote*

Setiap *gameobject* produk memiliki *script* ProdData.cs yang menyimpan informasi produk berupa nama produk, harga, dan informasi apakah produk tersebut sudah di *scan* atau belum. Hal ini dilakukan untuk memastikan produk yang sama tidak di*scan* dua kali. *Script* ProdData.cs sebagai berikut:

|  |
| --- |
| using UnityEngine;  public class ProdData : MonoBehaviour  {      public string objectName = " ";      public int objectPrice = 0;      public bool isScanned = false;      public bool inCart = false;    } |

Pada *gameobject* laser *barcode* memiliki *script* BarcodeController.cs yang mendeteksi produk tersebut. Proses ini dilakukan dengan membuat laser tersentuh dengan produk. *Script* tersebut juga melakukan pengecekan apakah produk tersebut sudah di*scan* atau belum. Pada *script* tersebut juga terdapat implementasi untuk menyimpan hasil scan pada *text* yang ada monitor. *Script* BarcodeController.cs adalah sebagai berikut.

|  |
| --- |
| void OnTriggerEnter(Collider other)      {                if ( other.gameObject.CompareTag("Product")){                  if (other.gameObject.GetComponentInChildren<ProdData>().isScanned == false)                  {                      barcodeValue = other.gameObject.GetComponentInChildren<ProdData>().objectName;                      barcodePrice = other.gameObject.GetComponentInChildren<ProdData>().objectPrice.ToString();                      totalHarga += other.gameObject.GetComponentInChildren<ProdData>().objectPrice;                      textProductGrabbed.text += '\n' + barcodeValue + "\t\t" + barcodePrice;                      textPriceTotal.text = totalHarga.ToString();                      other.gameObject.GetComponentInChildren<ProdData>().isScanned = true;                      Debug.Log("Barcode value set: " + barcodeValue);                  }              }        } |

1. Penetapan Harga

Penetapan harga dilakukan dengan melakukan klik *button* dengan *text ‘Set Price’* setelah melakukan *klik,* makan akan dipanggil *fungsi* SetPrice() yang berfungsi untuk mengubah *action index* sehingga *state* pada AI akan berganti. Selain itu, tampilan monitor juga akan berubah menjadi tampilan *monitor* input uang yang diberikan. Berikut adalah fungsi SetPrice() pada *script GameManager.cs.*

|  |
| --- |
| public void setPrice()      {          if(AIinFront == null) return;          else if (AIinFront.GetComponent<AIAgentMovement>().state == ConsumerState.WaitingForPrice)          {              price = barcode.GetComponent<BarcodeController>().totalHarga;              actionIndex = 1;              monitor.GetComponent<MonitorController>().activateLayar2();              fontTest.text = "Price Set";          }      } |

1. Penyimpanan Dan Perhitungan Uang

Uang yang diberikan konsumen memiliki Tag *MoneyIn.* Hal ini dilakukan untuk membedakan uang dari tempat kembalian dan uang yang diberikan oleh konsumen. Setelah uang berhasil disimpan di tempat *snap* penyimpanan uang, maka *text* penyimpanan uang akan bertambah. Hal ini diimplementasikan pada *script* CashInvertory.cs sebagai berikut.

|  |
| --- |
| using TMPro;  using UnityEngine;  public class CashInventory : MonoBehaviour  {      public int moneyValueTotal = 0; *// This can be set to the initial value of money*      public TMP\_Text moneyText;      void OnTriggerEnter(Collider other)      {          if (other.gameObject.CompareTag("MoneyIn"))          {              MoneyData moneyData = other.gameObject.GetComponent<MoneyData>();              if (moneyData != null)              {                  moneyValueTotal += moneyData.moneyValue; *// Add the money value from the MoneyData component*  *// Update the money value in the UI*                  moneyText.text = moneyValueTotal.ToString();  *// Optionally, destroy the money object after collecting it*  *//Destroy(other.gameObject);*              }               other.gameObject.GetComponent<MoneyData>().placedInCashInventory = true;          }      }  } |

1. Pemberian uang kembalian dan pengemasan barang

Pengguna akan menghitung terlebih dahulu uang yang diberikan oleh pengguna menggunakan *keyboard UI* dan melihat hasilnya di monitor. Hal ini dilakukan pada *script* NumberKeyboardController.cs. *Script* ini akan mendeteksi input yang diberikan berdasarkan. *Script* ini memberikan nilai pada masing masing tombol yang diklik dengan memberikan nilai pada variabel *string text. Script* ini juga melakukan perhitungan berapa jumlah kembalian yang harus diberikan. Berikut adalah implementasi *script* NumberKeyboardController.cs

|  |
| --- |
| using System;  using TMPro;  using UnityEngine;  using UnityEngine.UI;  public class NumberKeyboardController : MonoBehaviour  {      public TMP\_Text inputField;      public string displayText = "";  *// Start is called once before the first execution of Update after the MonoBehaviour is created*      public void setDisplaytext(string text)      {          displayText = text;      }      public void onButtonKeyboardClick()      {          GameObject buttonGO = UnityEngine.EventSystems.EventSystem.current.currentSelectedGameObject;          if (inputField.text.Length < 0 && displayText == "0")          {  *// Do nothing if the input field is empty and the button is "0"*              return;          }          else if (inputField.text.Length < 6)          {              inputField.text += displayText;          }          else          {  *// Optionally, you can handle the case where the input exceeds a certain length*              Debug.Log("Input exceeds maximum length.");          }      }      public void deleteCharButton()      {          if (inputField.text.Length > 0)          {              inputField.text = inputField.text.Substring(0, inputField.text.Length - 1);          }      }  } |

Setelah mengetahui uang yang diberikan, pengguna akan mengambil uang kembalian dari tempat penyimpanan uang kembalian. Uang kembalian memiliki Tag *MoneyOut*, sehingga tidak bertabrakan dengan uang yang diberikan oleh konsumen. Setelah menempatkan uang kembalian pada lokasi *snap* uang kembalian, maka *text* jumlah uang kembalian akan berubah sesuai dengan uang yang ditempatkan pada lokasi *snap* uang kembalian. Implementasi ini dilakukan pada *script* PlaceProduct.cs yang juga menyimpan informasi apakah plastik sudah ditempatkan pada lokasi *snap* plastic atau belum. Berikut adalah fungsi OntriggerEnter() pada *script* PlaceProduct.cs.

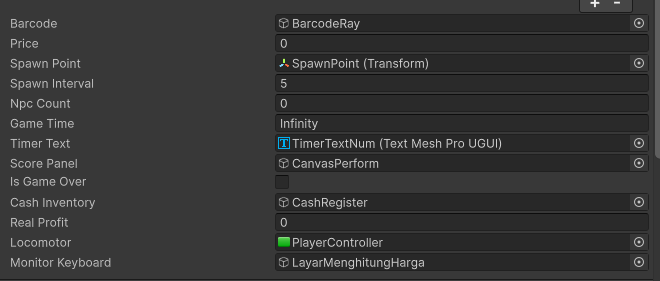
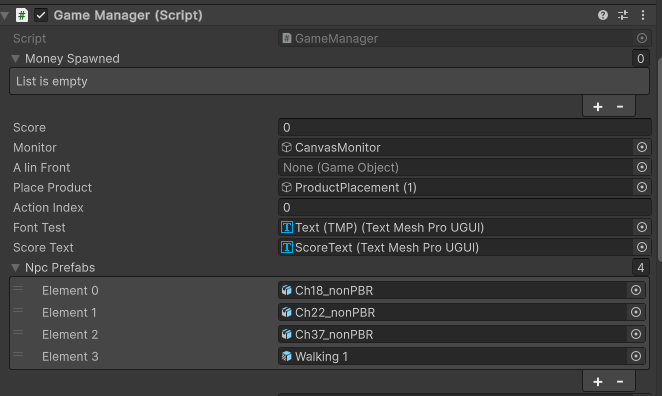
|  |
| --- |
| void OnTriggerEnter(Collider other)      {          if (other.gameObject.CompareTag("Plastic"))          {              plastic = other.gameObject;          }            if (other.gameObject.CompareTag("MoneyOut"))          {              money.Add(other.gameObject);              totalHarga += other.GetComponent<MoneyData>().moneyValue;              changePlacementText.text = totalHarga.ToString();          }      } |

Apabila produk sudah dimasukkan pada lokasi *snap* didalam plastik, maka informasi tersebut akan disimpan dalam suatu *List gameobejct* yang diimplementasikan pada *script* ProductInPlastic.cs. Apabila produk dikeluarkan dari plastik, maka produk tersebut akan dikeluarkan dari plastik. Berikut adalah *script* ProductInPlastic.cs

|  |
| --- |
| using System.Collections.Generic;  using Unity.VisualScripting;  using UnityEngine;  public class ProductInPlastic : MonoBehaviour  {      public List<GameObject> productsInCart = new List<GameObject>();      void OnTriggerEnter(Collider other)  {          if (other.gameObject.CompareTag("Product"))          {              if (!productsInCart.Contains(other.gameObject))              {                  productsInCart.Add(other.gameObject);              }          }  }     void OnTriggerExit(Collider other)  {          if (other.gameObject.CompareTag("Product"))          {              if (!productsInCart.Contains(other.gameObject))              {                  productsInCart.Remove(other.gameObject);          }  }  } |

1. Pembuatan *Game Manager*

Game Manager memberikan implementasi dari alur aplikasi *MainGame* dan *Tutorial. Game Manager* memberikan pengaturan untuk waktu permainan, jumlah konsumen, waktu durasi, interval konsumen datang, pengaturan skor, dan mekanisme *gameover.*Hal ini dilakukan untuk memudahkan implementasi mode *Tutorial* dan *MainGame* dan menjaga modularitas aplikasi apabila ada penambahan fitur dimasa depan*.* Implementasi *GameManager* terdapat pada Lampiran 3. Gambar 4.21 adalah tampilan *game component game manager.*



Gambar 4. Tampilan *game component game manager*

Perbedaan implementasi *game manager* mode *MainGame* dan *Tutorial* terletak pada waktu dimana mode *tutorial* memiliki waktu tidak terbatas dan *maingame* memiliki waktu terbatas. Selain itu, mode tutorial memiliki *script manager* khusus untuk menampilkan tampilan tutorial pada TutorialManager.cs *script* ini akan menampilkan *text* beserta *video* demonstrasi untuk memudahkan pengguna. Berikut adalah *script* TutorialManager.cs

|  |
| --- |
| public class TutorialManagerer : MonoBehaviour  {      [Header("Tutorial UI")]      public TMP\_Text instructionText;      *// Text di canvas*  *//public Image instructionImage;        // Tambahkan Image di Canvas untuk tampilkan gambar*      private int currentStep = 0;      public Button buttonNext;      public Button buttonPrev;      public Texture[] tutorialTextures;      public RawImage tutorialRawImage;      public VideoPlayer videoPlayer;      public VideoClip[] tutorialVideos;      [Header("Tutorial Steps")]      [TextArea(2, 5)]      private string[] tutorialSteps = new string[]      {          "1. Gerakkan tubuh dengan joystick!",*//done*          "2. Grab dan tempatkan produk di meja kasir.",          "3. Scan barang dengan barcode scanner.", *//done*          "4. Tekan tombol 'Set Price' di monitor.", *//done*          "5. Ambil uang dan taruh di cash register.", *//done*          "6. Ketik nominal uang pembeli di keyboard.", *//done*          "7. Ambil kembalian dan tempatkan di kotak.", *//done*          "8. Lakukan Packing barang dan klik product done",*//done*  *//done bbutuh di pisah*          "Tutorial Selesai !"      };      [Header("Tutorial Images")]      public Sprite[] tutorialImages; *// Drag & drop semua gambar sesuai urutan step ke sini di Inspector*      void Start()      {  *// Pastikan teks & gambar awal tampil*          ShowInstruction();  *//ShowImage();*        }      public void NextStep()      {          if (currentStep < tutorialSteps.Length - 1)          {              currentStep++;              ShowInstruction();  *//ShowImage();*          }      }      public void PrevStep()      {          if (currentStep > 0)          {              currentStep--;              ShowInstruction();          }      }      private void ShowInstruction()      {          instructionText.text = tutorialSteps[currentStep];          videoPlayer.clip = tutorialVideos[currentStep];        }      void Update()      {  *// Tombol keyboard untuk navigasi*          if (Input.GetKeyDown(KeyCode.D))          {              buttonNext.onClick.Invoke();          }          if (Input.GetKeyDown(KeyCode.A))          {              buttonPrev.onClick.Invoke();          }      }  } |

1. Testing

*Testing* atau pengujian dilakukan melalui dua tahap, yaitu pengujian internal menggunakan *black box testing* dengan *equivalent partition* yang membagi pengujian menjadi beberapa kelompok. Kelas dari pengujian internal dibentuk berdasarkan kebutuhan fitur dari aplikasi. Terdapat 11 kelas dengan kasus pengujiannya masing masing. Seluruh kasus pengujian ini akan diuji satu persatu didalam *VR Headset* Meta Quest 2*.* Kelas kelas dari pengujian internal meliputi *Locomotion, Main Menu, Grab, Distance Grab, Snap Interaction, AI Konsumen, Sistem kasir, Sound Effects (SFX), Tutorial, Main Game,* dan, *Game Manager.*

1. Pengujian Internal

Pengujian internal berfungsi untuk memastikan fitur - fitur utama pada aplikasi sudah terimplementasikan dengan baik. Pengujian internal dibagi menjadi beberapa kelas atau kelompok yang sesuai dengan kebutuhan aplikasi. Kelompok ini adalah *Locomotion, Main Menu, Grab, Distance Grab, Snap* Barang, Konsumen AI, Sistem Kasir, *Sound Effect (SFX), Tutorial, Main Game, Game Manager*. Pengujian internal dilakukan dengan cara pengecekan masing masing kasus pada pengujian apakah sesuai dengan hasil yang diharapkan atau tidak. Apabila sesuai maka akan ditulis ‘VALID’ namun apabila tidak sesuai, maka akan ditulis ‘INVALID’. Pengujian langsung dilakukan dengan menggunakan headset *meta quest* 2 pada hasil *build* aplikasi dari Unity. Sehinga, diharapkan hasil pengujian dapat lebih akurat dan sesuai dengan keperluan aplikasi sebelum dilakukan pengujian oleh ahli. Berikut adalah hasil dari masing masing kelompok pengujian.

1. *Locomotion* (Pergerakan dalam dunia VR)

Kelas pengujian *Locomotion* adalah kelompok pengujian yang menguji implementasi pergerakan pengguna dalam dunia VR. Pengujian ini dilakukan secara langsung di *headset* Meta Quest 2 dengan menggunakan *remote* VR sebagai media menggerakkan pengguna. Terdapat dua *test* untuk menguji cara pergerakan dan akses pergerakan pengguna untuk tidak dapat menembus tembok dan melewati tembok sehingga dapat bergerak ke area yang tidak dapat dilewati oleh pengguna. Hasil dari masing masing *case* adalah VALID. Tabel 4.3 berisikan detail dan hasil dari masing masing pengujian tiap studi kasus kelas *Locomotion* yang diberikan

Tabel 4. Kelas Pengujian Internal *Locomotion*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Deskripsi | Hasil | Ekspektasi Hasil | Kesimpulan |
| TA01 | Pengguna berjalan menggunakan joystick pada controller | Ketika menggerakkan Joystick, pengguna dapat bergerak dan berpindah tempat sesuai dengan arah yang diinginkan | Pengguna dapat berjalan sesuai dengan arah input joystick | VALID |
| TA02 | Pengguna melewati area yang dapat dilewati dan tidak untuk sebaliknya | Pengguna tidak melewati meja kasir dan tembus dari tembok. | Pengguna akan berhenti ketika ingin lewat dari meja kasir | VALID |

1. *Main Menu*

Kelas pengujian *Main Menu* adalah kelompok pengujian yang menguji implementasi fitur fitur pada *main menu.* Utamanya, pengujian yang dilakukan adalah proses klik *button* atau tombol yang terdapat pada *main menu.* Hasil dari masing masing *test case* adalah VALID. Tabel 4.4 berisikan detail dan hasil dari masing masing pengujian tiap studi kasus kelas *Main Menu* yang diberikan.

Tabel 4. Kelas Pengujian Internal *Main Game*

| ID | Deskripsi | Hasil | Ekspektasi Hasil | Kesimpulan |
| --- | --- | --- | --- | --- |
| TB01 | Pengguna tidak dapat jalan di main menu | Ketika analog digerakkan, pengguna tetap pada tempatnya. | Ketika pengguna bergerak dengan analog, pemain tetap tidak dapat bergerak. | VALID |
| TB02 | Klik button mulai | Setelah selesai di klik, loading sebentar dan langsung dipindahkan ke *scene Main Game.* | Ketika pengguna klik, langsung dipindahkan ke scene permainan utama. | VALID |
| TB03 | Klik button tutorial | Setelah selesai klik, pengguna langsung berpindah ke *scene tutorial*. | Ketika pengguna klik, langsung dipindahkan ke scene tutorial | VALID |
| TB04 | Klik button keluar | Seteleah klik, keluar dari aplikasi dan berpindah ke layar *Home* *Meta.* | Ketika pengguna klik, langsung keluar dari aplikasi | VALID |

1. *Grab*

Kelas pengujian *Grab* adalah kelompok pengujian yang menguji interaksi *grab* standar pada aplikasi. Pengujian ini dilakukan pada masing masing *game object* yang telah direncanakan memiliki *fitur grab* pada objek tersebut. *Remote* akan didekatkan ke objek untuk menguji apakah objek dapat dipegang. Hasil dari masing masing *test case* adalah VALID. Tabel 4.5 berisikan detail dan hasil dari masing masing pengujian tiap studi kasus kelas *Grab* yang diberikan.

Tabel 4. Kelas Pengujian Internal *Grab*

| ID | Deskripsi | Hasil | Ekspektasi Hasil | Kesimpulan |
| --- | --- | --- | --- | --- |
| TC01 | *Grab* Product\_  chips | Ketika remote didekatkan ke produk, produk akan terpegang dan mengikuti Gerakan *remote.* | Berhasil g*rab* Product\_chips | VALID |
| TC02 | *Grab* Product\_  energy  drink | Ketika remote didekatkan ke produk, produk akan terpegang dan mengikuti Gerakan *remote.* | Berhasil g*rab* Product\_  energydrink | VALID |
| TC03 | *Grab* Product\_  coke | Ketika remote didekatkan ke produk, produk akan terpegang dan mengikuti Gerakan *remote.* | Berhasil g*rab* Product\_coke | VALID |
| TC04 | *Grab* Product\_  cereal | Ketika remote didekatkan ke produk, produk akan terpegang dan mengikuti Gerakan *remote.* | Berhasil g*rab* Product\_cereal | VALID |
| TC05 | *Grab* Barcode\_  remote | Ketika remote didekatkan ke produk, produk akan terpegang dan mengikuti Gerakan *remote.* | Berhasil g*rab* Barcode\_  remote | VALID |

1. *Distance Grab*

Kelas pengujian *Distance Grab* adalah kelompok pengujian yang meguji interaksi *grab* dari jarak yang jauh pada objek yang direncakanan memiliki interasi tersebut. Pengujian ini dilakukan pada masing masing *game object* yang memiliki interaksi *distance grab* (*DG*) dengan mengarahkan *remote* ke objek tersebut. Tabel 4.6 berisikan detail dan hasil dari masing masing pengujian tiap studi kasus kelas *Distance Grab* yang diberikan.

Tabel 4. Kelas Pengujian Internal *Distance Grab*

| ID | Deskripsi | Hasil | Ekspektasi Hasil | Kesimpulan |
| --- | --- | --- | --- | --- |
| TD01 | *DistanceGrab* idr\_100k | Ketika *laser* remote didekatkan ke *objek* muncul bayangan produk dan Ketika di *klik,* produk akan bergerak ke *remote.* | Berhasil *DG* idr\_100k | VALID |
| TD02 | *DistanceGrab* Idr\_50k | Ketika *laser* remote didekatkan ke *objek* muncul bayangan produk dan Ketika di *klik,* produk akan bergerak ke *remote.* | Berhasil  *DG* Idr\_50k | VALID |
| TD03 | *DistanceGrab* Idr\_10k | Ketika *laser* remote didekatkan ke *objek* muncul bayangan produk dan Ketika di *klik,* produk akan bergerak ke *remote.* | Berhasil *DG* Idr\_10k | VALID |
| TD04 | *DistanceGrab* Idr\_5k | Ketika *laser* remote didekatkan ke *objek* muncul bayangan produk dan Ketika di *klik,* produk akan bergerak ke *remote.* | Berhasil *DG* Idr\_5k | VALID |
| TD05 | *DistanceGrab* Plastic\_bag | Ketika *laser* remote didekatkan ke *objek* muncul bayangan produk dan Ketika di *klik,* produk akan bergerak ke *remote.* | Berhasil *DG* Plastic\_bag | VALID |
| TD06 | *DistanceGrab* Product\_chips | Ketika *laser* remote didekatkan ke *objek* muncul bayangan produk dan Ketika di *klik,* produk akan bergerak ke *remote.* | Berhasil *DG* Product\_chips | VALID |
| TD07 | *DistanceGrab* Product\_energydrink | Ketika *laser* remote didekatkan ke *objek* muncul bayangan produk dan Ketika di *klik,* produk akan bergerak ke *remote.* | Berhasil *DG* Product\_energydrink | VALID |
| TD08 | *DistanceGrab* Product\_coke | Ketika *laser* remote didekatkan ke *objek* muncul bayangan produk dan Ketika di *klik,* produk akan bergerak ke *remote.* | Berhasil *DG* Product\_coke | VALID |
| TD09 | *DistanceGrab* Product\_cereal | Ketika *laser* remote didekatkan ke *objek* muncul bayangan produk dan Ketika di *klik,* produk akan bergerak ke *remote.* | Berhasil *DG* Product\_cereal | VALID |
| TD10 | *DistanceGrab* Barcode\_remote | Ketika *laser* remote didekatkan ke *objek* muncul bayangan produk dan Ketika di *klik,* produk akan bergerak ke *remote.* | Berhasil *DG* Barcode\_remote | VALID |

1. *Snap Interaction*

Kelas pengujian *Snap Interaction* adalah kelompok pengujian yang menguji interaksi *snap* pada lokasi objek yang ditentukan. Pengecekan ini dilakukan dengan mendekatkan objek yang memiliki pengaturan dapat di*snap* ke Lokasi tersebut. Seluruh hasil pengujian menghasilkan Kesimpulan ‘VALID’. Tabel 4.7 berisikan detail dan hasil dari masing masing pengujian tiap studi kasus kelas *Snap Interaction* yang diberikan.

Tabel 4. Kelas Pengujian Internal *Snap Interaction*

| ID | Deskripsi | Hasil | Ekspektasi Hasil | Kesimpulan |
| --- | --- | --- | --- | --- |
| TE01 | Lokasi *snap* tempat Penyimpanan objek pertama | Lokasi *snap* berubah warna Ketika objek di *hover* dan objek *snap* ke lokasi | Objek *Snap* ke lokasi | VALID |
| TE02 | Lokasi *snap* tempat Penyimpanan objek kedua | Lokasi *snap* berubah warna Ketika objek di *hover* dan objek *snap* ke lokasi | Objek *Snap* ke lokasi | VALID |
| TE03 | Lokasi *snap* tempat Penyimpanan objek ketiga | Lokasi *snap* berubah warna Ketika objek di *hover* dan objek *snap* ke lokasi | Objek *Snap* ke lokasi | VALID |
| TE04 | *Snap* tempat penyimpanan uang | Lokasi *snap* berubah warna Ketika uang konsumen di *hover* dan uang *snap* ke Lokasi. | Uang dari konsumen *snap* ke lokasi | VALID |
| TE05 | *Snap* tempat kembalian | Lokasi *snap* berubah warna Ketika uang konsumen di *hover* dan uang *snap* ke Lokasi. Selain itu, objek lain tidak akan tersnap | Uang kembalian *snap* ke lokasi | VALID |
| TE06 | tempat *snap* pemberian produk yang sudah dikemas | Lokasi *snap* berubah warna Ketika plastik konsumen di *hover* dan plastik *snap* ke Lokasi. Selain itu, objek lain tidak akan tersnap | Plastik *snap* ke lokasi | VALID |
| TE07 | Lokasi *snap* produk didalam plastik | Lokasi *snap* berubah warna Ketika produk konsumen di *hover* dan produk *snap* ke Lokasi. Selain itu, objek lain tidak akan tersnap | Produk *snap* ke dalam plastik dan mengikuti pergerakan plastik | VALID |

1. *AI* Konsumen

Kelas pengujian AI Konsumen berfokus pada perilaku konsumen dari awal memasuki toko sampai keluar dari toko. Pengujian dilakukan dengan melihat masing masing *state* AI dan kesesuainnya masing masing. Seluruh empat konsumen akan diuji kesesuainnya dengan perilaku seharusnya pada *state* terkait. Seluruh studi kasus pengujian memberikan hasil Kesimpulan ‘VALID’. Tabel 4.8 berisikan detail dan hasil dari masing masing pengujian tiap studi kasus kelas *Snap Interaction* yang diberikan.

Tabel 4. Kelas Pengujian Internal *Snap Interaction.*

| ID | Deskripsi | Hasil | Ekspektasi Hasil | Kesimpulan |
| --- | --- | --- | --- | --- |
| TF01 | Pembeli *consumen\_man1* – Perilaku tiap *state* | Konsumen berhasil mengikuti alur masing masing *state* secara berurut dan dengan *trigger* yang sesuai | Konsumen mengikuti alur masing masing *state* secara berurut dan dengan *trigger* yang sesuai | VALID |
| TF02 | Pembeli *consumen\_man1* – *animation* | Konsumen berhasil memainkan animasi dikondisi *state*nya. | Konsumen memainkan animasi sesuai dengan kondisi *state* atau saatnya. | VALID |
| TF03 | Pembeli *consumen\_man2* – Perilaku tiap *state* | Konsumen berhasil mengikuti alur masing masing *state* secara berurut dan dengan *trigger* yang sesuai | Konsumen mengikuti alur masing masing *state* secara berurut dan dengan *trigger* yang sesuai | VALID |
| TF04 | Pembeli *consumen\_man2* – *animation* | Konsumen berhasil memainkan animasi dikondisi *state*nya. | Konsumen memainkan animasi sesuai dengan kondisi *state* atau saatnya. | VALID |
| TF05 | Pembeli *consumen\_woman1* – Perilaku tiap *state* | Konsumen berhasil mengikuti alur masing masing *state* secara berurut dan dengan *trigger* yang sesuai | Konsumen mengikuti alur masing masing *state* secara berurut dan dengan *trigger* yang sesuai | VALID |
| TF06 | Pembeli *consumen\_woman1* – *animation* | Konsumen berhasil memainkan animasi dikondisi *state*nya. | Konsumen memainkan animasi sesuai dengan kondisi *state* atau saatnya. | VALID |
| TF07 | Pembeli *consumen\_woman1* – Perilaku tiap *state* | Konsumen berhasil mengikuti alur masing masing *state* secara berurut dan dengan *trigger* yang sesuai | Konsumen mengikuti alur masing masing *state* secara berurut dan dengan *trigger* yang sesuai | VALID |
| TF08 | Pembeli *consumen\_woman1* –*animation* | Konsumen berhasil memainkan animasi dikondisi *state*nya. | Konsumen memainkan animasi sesuai dengan kondisi *state* atau saatnya. | VALID |

1. Sistem Kasir

Kelas pengujian sistem kasir merupakan pengujian yang memastikan bahwa proses yang harus dilakukan pengguna dalam bekerja sebagai kasir berjalan dengan sesuai dan mudah digunakan. Seluruh kasus pengujian menghasilkan hasil ‘VALID’. Tabel 4.9 berisikan detail dan hasil dari masing masing pengujian tiap studi kasus kelas system kasir yang diberikan.

Tabel 4. Kelas Pengujian Internal Sistem Kasir*.*

| ID | Deskripsi | Hasil | Ekspektasi Hasil | Kesimpulan |
| --- | --- | --- | --- | --- |
| TG01 | *Scan* Produk - Deteksi | Setelah *barcode ray* didekatkan ke produk, nama dan harga produk muncul di monitor, produk yang sudah ketika di*scan* lagi, tidak muncul di monitor | Ketika *barcode ray* didekatkan ke produk, produk akan muncul di monitor, produk yang sudah di*scan* tidak akan muncul di monitor | VALID |
| TG02 | Klik *button set price* | Uang *spawn* di Lokasi dekat konsumen. | setelah klik *button set price*, konsumen akan memberikan uang. | VALID |
| TG03 | Penempatan uang | Uang dari konsumen akan memberika *update* pada *text* tempat penyimpanan uang. | Indikator uang tersimpan hanya akan bertambah ketika uang dari konsumen berhasil diberikan | VALID |
| TG04 | Pengemasan barang | Ketika selesai barang ditempatkan diplastik, barang akan mengikuti pergerakan plastik | Barang dapat ditempatkan diplastik dan mengikuti pergerakan plastik | VALID |
| TG05 | Input uang konsumen | Setiap *button* keyboard akan muncul dimonitor dengan sesuai dan perhitungannya pun otomatis | Konsumen dapat klik UI keyboard calculator dan inputan user akan muncul di layar monitor | VALID |
| TG06 | Kembalian | Ketika mengambil uang kembalian, uang kembalian *spawn* lagi di tempat kembalian | User dapat mengambil uang kembalian berapapun jumlahnya | VALID |
| TG07 | Klik button product done | Barang, Plastik, dan kosumen hilang dari *scene* | User dapat klik button product done. Setelah klik, barang, plastik, dan uang konsumen hilang dari scene dan juga konsumen keluar dari toko | VALID |

1. *Sound Effects* (SFX)

Kelas pengujian SFX adalah kelompok pengujian untuk efek suara. Terdapat pengujian untuk music *background* dan *SFX* setelah klik *button.* Pengujian dilakukan dengan mendengarkan suara yang dikeluarga oleh VR ketika menggunakan aplikasi dan klik SFX untuk *button.* Suara dari VR muncul langsung dari VR *Headset* dan volume *VR Headset* disesuaikan agar memudahkan proses pengujian.Tabel 4.10 berisikan detail dan hasil dari masing masing pengujian tiap studi kasus kelas SFX yang diberikan.

Tabel 4. Kelas Pengujian Internal *SFX*

| ID | Deskripsi | Hasil | Ekspektasi Hasil | Kesimpulan |
| --- | --- | --- | --- | --- |
| TH01 | *Background music* | *Background music* selalu dimainkan mulai dari aplikasi pertama kali dibuka | Background music selalu dimainkan selama aplikasi berjalan | VALID |
| TH02 | *Button SFX* | Muncul suara setelah button di click. | Setiap button di click akan muncul suara SFX | VALID |

1. *Tutorial*

Kelas pengujian *tutorial* berfokus pada fitur paduan pengguna dalam menggunakan aplikasi. Pengujian dilakukan dengan secara langsung membuka mode *tutorial* dalam aplikasi dan menyesuaikan hasilnya pada kasus yang diuji. Hasil dari pengujian tiap studi kasus kelas Tutorial adalah VALID. Tabel 4.11 berisikan detail dan hasil dari masing masing pengujian tiap studi kasus kelas tutorial yang diberikan.

Tabel 4. Kelas Pengujian Internal *Tutorial*

| ID | Deskripsi | Hasil | Ekspektasi Hasil | Kesimpulan |
| --- | --- | --- | --- | --- |
| TI01 | *Video Play 1* dan *Text 1* | Video dan teks muncul indeks yang sesuai muncul | Memunculkan video dan teks di indeks pertama | VALID |
| TI02 | *Video Play 2* dan *Text 2* | Video dan teks muncul indeks yang sesuai muncul | Memunculkan video dan teks di indeks kedua | VALID |
| TI03 | *Video Play 3* dan *Text 3* | Video dan teks muncul indeks yang sesuai muncul | Memunculkan video dan teks di indeks ketiga | VALID |
| TI04 | *Video Play 4* dan *Text 4* | Video dan teks muncul indeks yang sesuai muncul | Memunculkan video dan teks di indeks keempat | VALID |
| TI05 | *Video Play 5* dan *Text 5* | Video dan teks muncul indeks yang sesuai muncul | Memunculkan video dan teks di indeks kelima | VALID |
| TI06 | *Button Next* | Setelah *klik* button video dan teks indeks berikutnya muncul | Memunculkan konten tutorial di index setelah indeks yang sedang ditampilkan | VALID |
| TI07 | *Button Previous* | Setelah *klik* button video dan teks indeks sebelumnya muncul | Memunculkan konten tutorial di index sebelum indeks yang sedang ditampilkan | VALID |

1. *Main Game*

Kelas pengujian *Main Game* berfokus pada fitur khusus yang ada pada mode *MainGame.* Fitur ini berupa *Game over* dan *score* permainan. Dua fiturini dicoba dengan memainkan mode *MainGame* dan melihat kesesuaiannya dengan pengujian. Terdapat beberapa kasus juga yang mencoba untuk menguji kasus ketika tidak melakukan langkah yang diharuskan untuk mendapatkan *score*. Tabel 4.12 berisikan detail dan hasil dari masing masing pengujian tiap studi kasus kelas *MainGame* yang diberikan.

Tabel 4. Kelas Pengujian Internal *Main Game*

| ID | Deskripsi | Hasil | Ekspektasi Hasil | Kesimpulan |
| --- | --- | --- | --- | --- |
| TJ01 | Waktu Habis – Canvas *Game Over* | *Canvas game over* muncul saat waktu habis | Canvas game over muncul ketika waktu permainan habis. | VALID |
| TJ02 | Score – *Scan* Produk | *Score* bertambah satu poin, apabila tidak, tidak dapat satu poin | Apabila *scan* semua produk dari konsumen, akan mendapatkan satu poin, apabila tidak, tidak dapat satu poin | VALID |
| TJ03 | Score – Simpan Uang | *Score* bertambah satu poin, apabila tidak, tidak dapat satu poin | Apabila simpan semua uang dari konsumen, akan mendapatkan satu poin, apabila tidak, tidak dapat satu poin | VALID |
| TJ04 | Score – Uang Kembalian | *Score* bertambah satu poin, apabila tidak, tidak dapat satu poin | Apabilamenempatkan uang kembalian dengan tepat, akan mendapatkan satu poin, apabila tidak, tidak dapat satu poin | VALID |
| TJ05 | Score – Pengemasan Barang | *Score* bertambah satu poin, apabila tidak, tidak dapat satu poin | Apabila menempatkan seluruh barang pada plastic dan menempatkannya di meja kasir, akan mendapatkan satu poin, apabila tidak, tidak dapat satu poin | VALID |
| TJ06 | Score – Input Uang Konsumen | *Score* bertambah satu poin, apabila tidak, tidak dapat satu poin | Apabila dengan tepat *input* uang konsumen pada komputer, akan mendapatkan satu poin, apabila tidak, tidak dapat satu poin | VALID |

1. *Game Manager*

Kelas pengujian *Game Manager* berfokus pada hal hal yang berpengaruh pada alur permainan meliputi waktu permainan dan *spawn* konsumen. Pengujian dilakukan dengan memainkan mode *main game* dan *tutorial.* Beserta melihat jumlah konsumen yang berdatangan. Tabel 4.13 berisikan detail dan hasil dari masing masing pengujian tiap studi kasus kelas *Game Manager* yang diberikan.

Tabel 4. Kelas Pengujian Internal *Game Manager*

| ID | Deskripsi | Hasil | Ekspektasi Hasil | Kesimpulan |
| --- | --- | --- | --- | --- |
| TK01 | Waktu Permainan | Waktu habis dan konsumen berhenti berdatangan pada mode *main game* sedangkan tidak pada mode *tutorial.* | Mode tutorial tidak terbatas dan mode *main game* terbatas. | VALID |
| TK02 | *Spawn* Konsumen | *Spawn* sesuai dengan random dan tidak spontan. | *Spawn* konsumen secara random dan dengan interval tertenu, tidak secara spontan. | VALID |

1. Beta

Setelah tahapan *internal testing,* aplikasi telah memasuki fase Beta. Aplikasi memasuki fase beta karena seluruh fitur yang direncanakan difase perencanaan telah terimplementasikan di aplikasi sesuai hasil dari *internal testing.* Pada fase beta, dilakukan tahapan *external* testing untuk menguji aplikasi dari pihak eksternal.

1. External Testing

a.Target Evaluator

b.Skenario Pengujian

C.Hasil Pengujian

d.evaluasi

e.kendala

4.3release

# **DAFTAR PUSTAKA**

Shankar, V., Kalyanam, K., Setia, P., Golmohammadi, A., Tirunillai, S., Douglass, T., ... & Waddoups, R. (2021). How technology is changing retail. Journal of Retailing, 97(1), 13-27.

Boppana, V. R. (2022). Virtual Reality Applications in CRM Training and Support. EPH-International Journal of Business & Management Science, 8(3), 1-8.

Yang, C. J., & Wu, C. F. (2020). Study on Learning Effectiveness of Virtual Reality Technology in Retail Store Design Course. In Virtual, Augmented and Mixed Reality. Industrial and Everyday Life Applications: 12th International Conference, VAMR 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part II 22 (pp. 327-337). Springer International Publishing.

Ciekanowska, A., Kiszczak - Gliński, A., & Dziedzic, K. (2021). Comparative analysis of Unity and Unreal Engine efficiency in creating virtual exhibitions of 3D scanned models. Journal of Computer Sciences Institute, 20, 247–253. https://doi.org/10.35784/jcsi.2698

Ramadan & Widyani (2013). Game Development Life Cycle Guidelines. 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)

lbahari, J., & Albahari, B. (2020). C# 8.0 in a Nutshell. O'Reilly Media.

Burdea, G. C., & Coiffet, P. (2021). Virtual Reality Technology. Wiley.

Goldstone, W. (2020). Unity 2020 Game Development Cookbook. Packt Publishing.

Hernandez, M., et al. (2022). "Efficiency in Game Development with C#: A Unity Case Study." Journal of Game Development, 15(3), 245-259.

Johnson, L. (2020). "The Evolution of Cashier Systems in Modern Retail." Retail Technology Review, 12(4), 345-360.

Lee, H., & Kim, J. (2022). "Impact of Self-Checkout Systems on Retail Operations." International Journal of Retail Management, 18(2), 122-138.

Levy, M., & Weitz, B. A. (2020). Retailing Management. McGraw-Hill Education.

Rogers, S. (2020). Level Up! The Guide to Great Video Game Design. Wiley.

Smith, R., et al. (2021). "Data Analytics in Retail: A Case Study of Walmart." Journal of Business Analytics, 5(1), 67-80.

Thompson, L., & Nguyen, T. (2021). "Optimizing the Game Development Life Cycle for Indie Developers." Game Studies Journal, 10(4), 299-315.

Zhang, Y., & Liu, X. (2021). "Unity in Education: Enhancing Learning with VR Applications." Educational Technology Research, 29(3), 185-200.

Gonzalez, R., et al. (2023). "Virtual Reality in Surgical Training: A Comparative Study." Medical Education Review, 37(1), 112-126.

Lampiran

| *Scene* | Deskripsi |
| --- | --- |
| Halaman menu utama | Menu utama berisikan empat tombol mulai, pengaturan, credit, dan keluar. Pemain mengarahkan controller dan menekan tombol pada controller untuk memilih button. |
| Halaman memilih permainan | Deskripsi |
|  | Menu memilih mode permainain terdapat dua, tutorial dan simulasi. |
| Halaman tutorial | Deskripsi |
|  | Pemain diharuskan melakukan aktivitas sesuai dengan paduan pada kanvas petunjuk untuk melanjutkan proses tutorial. |
| Halaman Simulasi 1 | Deskripsi |

Tabel 3. Lanjutan

|  |  |
| --- | --- |
|  | Awal masuk mode simulasi |
| Halaman Simulasi 2 | Deskripsi |
|  | Konsumen berkeliling di toko |
| Halaman Simulasi 3 | Deskripsi |
|  | Konsumen mengantri di depan meja kasir |
| Halaman Simulasi 4 | Deskripsi |
|  | Konsumen memberikan barang belanjaan |
| Halaman Simulasi 5 | Deskripsi |
|  | Konsumen scan barang menggunakan barcode dan memencet keyboard untuk finalisasi harga barang |

Tabel 3. Lanjutan

|  |  |
| --- | --- |
| Halaman Simulasi 6 | Deskripsi |
|  | Konsumen memberikan uang ,menempatkan uang di laci, dan input nilai uang menggunakan keyboard untuk mendapatkan perhitungan kembalian |
| Halaman Simulasi 7 | Deskripsi |
|  | Pemain melakukan pengemasan barang dan memberikan kembalian apabila ada kembalian. Setelah itu, memencet tombol pada keyboard untuk menyelesaikan pelayanan. |
| Halaman Simulasi 8 | Deskripsi |
|  | Konsumen yang telah selesai dilayani akan keluar toko dan melanjutkan ke antrian berikutnya selama waktu belum habis. |
| Halaman Simulasi 9 | Deskripsi |
|  | Ketika waktu habis akan ditampilkan total score yang didapat dan terdapat tombol untuk Kembali ke menu utama |

|  |
| --- |
| using UnityEngine;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using UnityEngine.AI;  public enum ConsumerState  {  Entering,  Browsing,  Queuing,  WaitingInQueue,  GivingProduct,  WaitingForPrice,  Paying,  Receiving,  Leaving  }  public class AIAgentMovement : MonoBehaviour  {  public int totalPay = 0;  public GameObject gm;  bool hasPlayedIdle = false;  public GameObject[] productPrefabs; // Assign in Inspector  public Transform productSpawnPoint; // Assign in Inspector  public int minProductAmount = 1;  public int maxProductAmount = 3;  public NavMeshAgent agent;  public Transform storeEntry;  public Transform[] browsingPoints;  public GameObject[] spawnedObjects;  public Transform cashierPoint;  public Transform exitPoint;  public Transform[] queuePoints; // Points for queue positions  private int currentBrowseIndex = 0;  public ConsumerState state = ConsumerState.Entering;  public float browseDuration = 5f;  private float browseTimer;  public List<GameObject> money = new List<GameObject>();  private Animator anim;  public GameObject[] moneyPrefabs; // 0: 5000, 1: 10000, 2: 50000, 3: 100000  public Transform moneySpawnPoint;  // Static queue for all agents  private int myQueueIndex = -1;  private bool isMyTurn = false;  void Start()  {  browseTimer = browseDuration;  anim = GetComponent<Animator>();  agent = GetComponent<NavMeshAgent>();  agent.SetDestination(storeEntry.position);  foreach (Transform point in browsingPoints)  {  point.SetParent(null, true); // Ensure browsing points are not parented to this agent  }  foreach (Transform point in queuePoints)  {  point.SetParent(null, true); // Ensure queue points are not parented to this agent  }  cashierPoint.SetParent(null, true); // Ensure cashier point is not parented to this agent  exitPoint.SetParent(null, true); // Ensure exit point is not parented to this agent  storeEntry.SetParent(null, true); // Ensure store entry is not parented to this agent  productSpawnPoint.SetParent(null, true); // Ensure product spawn point is not parented to this agent  }    void Update()  {  if (gm.GetComponent<GameManager>().actionIndex == 1 )  {  SetPrice(gm.GetComponent<GameManager>().price);  }  if (gm.GetComponent<GameManager>().actionIndex == 2 )  {  //gm.GetComponent<GameManager>().productDone();    ProductDone();  }    switch (state)  {  case ConsumerState.Entering:  if (ReachedDestination())  {  state = ConsumerState.Browsing;  }  break;  case ConsumerState.Browsing:  browseTimer -= Time.deltaTime;  if (ReachedDestination())  {    Debug.Log("browse time = " + browseTimer);  if (browseTimer < 0)  {  state = ConsumerState.Queuing;  JoinQueue();  browseTimer = 0;  }  else  {  GoToNextBrowsingPoint();  }  }  break;  case ConsumerState.Queuing:  // Move to assigned queue point  if (ReachedDestination())  {  anim.Play("Idle"); // Idle when reached queue position  state = ConsumerState.WaitingInQueue;  }  break;  case ConsumerState.WaitingInQueue:  // Wait until it's my turn  if (isMyTurn)  {  //MoveToQueuePosition();  if (ReachedDestination())  {  state = ConsumerState.GivingProduct;  anim.Play("Idle");  StartCoroutine(GiveProductAtCashier());  }  }  else  {  // If not my turn and reached queue position, stay idle  if (ReachedDestination())  {  anim.Play("Idle");  }  }  break;  case ConsumerState.GivingProduct:  // Wait for coroutine to finish  gm.GetComponent<GameManager>().AIinFront = this.gameObject;  break;  case ConsumerState.WaitingForPrice:  // Wait for player/cashier to call SetPrice  break;  case ConsumerState.Paying:  // Wait for player/cashier to call ProductDone  break;  case ConsumerState.Leaving:    anim.Play("Jalan");  this.DestroyProducts();  if (ReachedDestination())  {  gm.GetComponent<GameManager>().actionIndex = 0;  gm.GetComponent<GameManager>().barcode.GetComponent<BarcodeController>().textPriceTotal.text = "0";  gm.GetComponent<GameManager>().barcode.GetComponent<BarcodeController>().textProductGrabbed.text = " ";  gm.GetComponent<GameManager>().barcode.GetComponent<BarcodeController>().totalHarga = 0;  LeaveQueue();  gm.GetComponent<GameManager>().npcCount--; ;  Destroy(gameObject);  }  break;  }  }  void GiveRandomProduct()  {  if (productPrefabs.Length == 0 || productSpawnPoint == null) return;  int productIndex = 0;  int amount = Random.Range(minProductAmount, maxProductAmount + 1);  spawnedObjects = new GameObject[amount];  for (int i = 0; i < amount; i++)  {  productIndex = Random.Range(0, productPrefabs.Length);  Vector3 offset = new Vector3(0.2f \* i, 0, 0); // Spread out products a bit  spawnedObjects[i] = Instantiate(productPrefabs[productIndex], productSpawnPoint.position + offset, Quaternion.identity);  spawnedObjects[i].SetActive(true);  }  }  bool ReachedDestination()  {  return !agent.pathPending && agent.remainingDistance <= agent.stoppingDistance;  }  void GoToNextBrowsingPoint()  {  currentBrowseIndex = Random.Range(0, browsingPoints.Length);  agent.SetDestination(browsingPoints[currentBrowseIndex].position);  }  void JoinQueue()  {  gm.GetComponent<GameManager>().queue.Enqueue(this);  myQueueIndex = gm.GetComponent<GameManager>().queue.Count - 1;  MoveToQueuePosition();  state = ConsumerState.Queuing;  StartCoroutine(CheckQueueTurn());  }  void MoveToQueuePosition()  {  if (queuePoints != null && myQueueIndex < queuePoints.Length)  {  agent.SetDestination(queuePoints[myQueueIndex].position);  anim.Play("Jalan"); // Play walking animation when moving to queue  }  // Only move to cashier when it's my turn  if (isMyTurn)  {  agent.SetDestination(cashierPoint.position);  anim.Play("Jalan"); // Play walking animation when moving to cashier  }  }  IEnumerator CheckQueueTurn()  {  while ( gm.GetComponent<GameManager>().queue.Peek() != this)  {  myQueueIndex = GetQueueIndex();  MoveToQueuePosition();  yield return new WaitForSeconds(0.5f);  }  isMyTurn = true;  MoveToQueuePosition();    }  int GetQueueIndex()  {  int idx = 0;  foreach (var agent in gm.GetComponent<GameManager>().queue)  {  if (agent == this) return idx;  idx++;  }  return -1;  }  IEnumerator GiveProductAtCashier()  {  // Step 4: Give product  Debug.Log("Giving product...");  anim.Play("GiveProduct"); // Optional animation  this.GiveRandomProduct();  yield return new WaitForSeconds(1.5f);  // Wait for cashier/player to set price  state = ConsumerState.WaitingForPrice;  Debug.Log("Waiting for cashier to set price...");  // Wait until SetPrice() is called  while (state == ConsumerState.WaitingForPrice)  yield return null;  }  // Call this from cashier/player when price is set  public void SetPrice(int price)  {  if (state == ConsumerState.WaitingForPrice)  {  Debug.Log("Price set by cashier. Paying...");  state = ConsumerState.Paying;  StartCoroutine(PayAndWaitForProduct(price));  }  }  IEnumerator PayAndWaitForProduct(int price)  {  anim.Play("Pay");  yield return new WaitForSeconds(1.0f);  GiveMoney(price);  Debug.Log("Waiting for product to be done...");  while (state == ConsumerState.Paying)  yield return null;  }  // filepath: [AIAgentMovement.cs](http://\_vscodecontentref\_/1)  void GiveMoney(int price)  {  int[] values = { 5000,10000,50000,100000 };  totalPay = 0;  int i = 0;  List<int> bills = new List<int>();  while (totalPay < price)  {  i = Random.Range(0, values.Length-1);  totalPay += values[i];  bills.Add(i);  if(bills.Count >= 5) // Limit to 5 bills  {  bills.Add(values[0]);  break;  }  }    foreach(int bill in bills)  {  if (bill < moneyPrefabs.Length)  {  Vector3 offset = new Vector3(0.1f \* bills.IndexOf(bill), 0, 0); // Spread out money a bit  GameObject spawnedMoney = Instantiate(moneyPrefabs[bill], moneySpawnPoint.position + offset, Quaternion.identity);  money.Add(spawnedMoney); }  }    }  public void DestroyProducts()  {  if (spawnedObjects != null)  {  foreach (var obj in spawnedObjects)  {  if (obj != null)  {  Destroy(obj);  }  }  }  }  IEnumerator PayAndWaitForProduct()  {  anim.Play("Pay"); // Optional animation  yield return new WaitForSeconds(1.5f);  // Wait for cashier/player to mark product as done  Debug.Log("Waiting for product to be done...");  // Wait until ProductDone() is called  while (state == ConsumerState.Paying)  yield return null;  }  // Call this from cashier/player when product is done  public void ProductDone()  {  if (state == ConsumerState.Paying)  {  // gm.GetComponent<GameManager>().checkInplace();  // gm.GetComponent<GameManager>().changeScore();  //destroyCustomerOwned();  state = ConsumerState.Leaving;  agent.SetDestination(exitPoint.position);  }  }  public void destroyCustomerOwned()  {  GameObject plastic = gm.GetComponent<GameManager>().PlaceProduct.GetComponent<PlaceProduct>().plastic;  if (plastic != null)  {  Destroy(plastic);  }  if( gm.GetComponent<GameManager>().PlaceProduct.GetComponent<PlaceProduct>().money != null)  {  foreach (GameObject m in gm.GetComponent<GameManager>().PlaceProduct.GetComponent<PlaceProduct>().money)  {  if (m != null)  {  Destroy(m);  }  }  }    }  void LeaveQueue()  {  if ( gm.GetComponent<GameManager>().queue.Count > 0 && gm.GetComponent<GameManager>().queue.Peek() == this)  {  gm.GetComponent<GameManager>().queue.Dequeue();  }  }    } |

|  |
| --- |
| using TMPro;  using UnityEngine;  using System.Collections;  using System.Collections.Generic;  using Unity.XR.CoreUtils;  using Unity.VisualScripting;  public class GameManager : MonoBehaviour  {   public List<GameObject> moneySpawned = new List<GameObject>();      public int score;      public GameObject monitor;      public GameObject AIinFront;      public GameObject PlaceProduct;      public int actionIndex = 0;      public TMP\_Text fontTest;      public TMP\_Text scoreText;  public GameObject[] npcPrefabs; *// Assign 4 AI prefabs in Inspector*      public GameObject barcode;      public int price;      public Transform spawnPoint; *// Assign spawn location in Inspector*      public float spawnInterval = 5f; *// Time in seconds between spawns*      public int npcCount = 0;      public float gameTime = 120f; *// waktu dalam detik (misal 2 menit)*      private float currentTime;      public TMP\_Text timerText; *// assign di Inspector untuk menampilkan waktu*      public GameObject scorePanel; *// assign panel UI untuk menampilkan skor akhir*      public bool isGameOver;      public GameObject cashInventory;      public Queue<AIAgentMovement> queue = new Queue<AIAgentMovement>();      public int realProfit;      public GameObject locomotor;      public GameObject monitorKeyboard;      void Start()      {          currentTime = gameTime;          StartCoroutine(SpawnNPCsRoutine());          scorePanel.SetActive(false); *// pastikan panel skor tidak aktif di awal*          this.gameObject.GetComponent<GameManager>().enabled = true;          isGameOver = false;      }            IEnumerator SpawnNPCsRoutine()      {          while (true)          {              if (npcCount < 4)              {                  SpawnNPC();                  yield return new WaitForSeconds(spawnInterval);              }              else              {                  yield return new WaitForSeconds(spawnInterval);                  ; *// Wait for the next frame if limit is reached*              }          }        }      void SpawnNPC()      {          if (npcPrefabs.Length == 0) return;      int randIndex = Random.Range(0, npcPrefabs.Length);      GameObject npc = Instantiate(npcPrefabs[randIndex], spawnPoint.position, spawnPoint.rotation);      npc.SetActive(true);      npcCount++;      }      void Update()      {          if (!isGameOver)          {              currentTime -= Time.deltaTime;              if (currentTime < 0)              {                  currentTime = 0;                  EndGame();              }              UpdateTimerUI();          }  *// if (Input.GetKeyDown(KeyCode.A))*  *// {*  *//     setPrice();*  *// }*  *// if (Input.GetKeyDown(KeyCode.B))*  *// {*  *//     productDone();*  *// }*      }      void UpdateTimerUI()      {          int minutes = Mathf.FloorToInt(currentTime / 60f);          int seconds = Mathf.FloorToInt(currentTime % 60f);          timerText.text = string.Format("{0:00}:{1:00}", minutes, seconds);      }  */// <summary>*  */// This function is called when the behaviour becomes disabled or inactive.*  */// </summary>*      void OnDisable()      {          StopAllCoroutines();           foreach (var npc in GameObject.FindGameObjectsWithTag("NPC"))          {              if (npc != null)              {                  Destroy(npc);              }          }        }      void EndGame()      {          if (cashInventory.GetComponent<CashInventory>().moneyValueTotal == realProfit)          {  *//score += 75;*          }          isGameOver = true;          scorePanel.SetActive(true); *// tampilkan panel skor*          scorePanel.GetComponentInChildren<TMP\_Text>().text = "Game Over !\n Your Final Score is\n" + score.ToString();  *// Nonaktifkan semua NPC movement (misal dengan tag "NPC")*  *//this.gameObject.GetComponent<GameManager>().enabled = false;*  *// Nonaktifkan player movement (misal script VR movement Anda)*  *// var player = GameObject.FindGameObjectWithTag("Player");*  *// if (player != null)*  *// {*  *//         var moveScript = locomotor;*  *//     if (moveScript != null)*  *//         moveScript.SetActive(false);*  *// }*      }          public void setPrice()      {          if(AIinFront == null) return;          else if (AIinFront.GetComponent<AIAgentMovement>().state == ConsumerState.WaitingForPrice)          {              price = barcode.GetComponent<BarcodeController>().totalHarga;              actionIndex = 1;              monitor.GetComponent<MonitorController>().activateLayar2();              fontTest.text = "Price Set";          }      }      public void productDone()      {          if(AIinFront == null) return;          else if (AIinFront.GetComponent<AIAgentMovement>().state == ConsumerState.Paying)          {              actionIndex = 2;  *//this.checkInplace();*  *//this.changeScore();*  *//AIinFront.GetComponent<AIAgentMovement>().destroyCustomerOwned();*              monitor.GetComponent<MonitorController>().activateLayar1();                fontTest.text = "Product Done";          }        }      public void checkInplace()      {          realProfit += AIinFront.GetComponent<AIAgentMovement>().totalPay;          int hargaKembalian = AIinFront.GetComponent<AIAgentMovement>().totalPay - price;          Debug.Log("Kembalian : " + hargaKembalian);          if (hargaKembalian == PlaceProduct.GetComponent<PlaceProduct>().totalHarga && price != 0)          {              score += 1;          }          if (PlaceProduct.GetComponent<PlaceProduct>().plastic != null)          {              GameObject[] customerProd = AIinFront.GetComponent<AIAgentMovement>().spawnedObjects;              List<GameObject> productsInCart2;              productsInCart2 = PlaceProduct.GetComponent<PlaceProduct>().plastic.GetNamedChild("SnapInteractable").GetComponent<ProductInPlastic>().productsInCart;              if (customerProd.Length == productsInCart2.Count)              {                  score += 1;              }          }          bool scannedProductsMatch = true;          GameObject[] customerProducts = AIinFront.GetComponent<AIAgentMovement>().spawnedObjects;          for (int i = 0; i < customerProducts.Length; i++)          {                if (customerProducts[i].GetComponentInChildren<ProdData>().isScanned == false)              {                  scannedProductsMatch = false;                  break;              }          }          if (scannedProductsMatch)          {              score += 1;          }          if (monitorKeyboard.GetComponent<CountScreenController>().money == AIinFront.GetComponent<AIAgentMovement>().totalPay && price != 0)          {              score += 1;          }            moneySpawned = AIinFront.GetComponent<AIAgentMovement>().money;          bool allMoneyPlaced = true;          if (moneySpawned.Count>0)          {                foreach (var money in moneySpawned)              {                  if (money != null)                  {                      Debug.Log(money.GetComponentInChildren<MoneyData>().moneyValue);                      if (money.GetComponentInChildren<MoneyData>().placedInCashInventory == false)                      {                          allMoneyPlaced = false;                          break;                      }                  }              }          }          if (allMoneyPlaced && moneySpawned.Count > 0 && price != 0)          {              score += 1;              moneySpawned.Clear();          }  *// else if (score == 0)*  *// {*  *//     score--;*  *// }*          PlaceProduct.GetComponent<PlaceProduct>().totalHarga = 0;          PlaceProduct.GetComponent<PlaceProduct>().changePlacementText.text = "";        }      public void changeScore()      {          scoreText.text = score.ToString();        }  } |