universitas
MALIKUSSALEH

PENGEMBANGAN APLIKASI PENGETAHUAN BAHASA PEMOGRAMAN DASAR DAN LANJUT BERBASIS *ANDROID* MENGGUNAKAN METODE *GAME DEVELOPMENT LIFE CYCLE*

SKRIPSI

Disusun Sebagai Syarat Memperoleh Gelar Sarjana Komputer
Prodi Sistem Informasi Fakultas Teknik
Universitas Malikussaleh

DISUSUN OLEH:

NAMA : BAGAS AULIA ALFASYAM
NIM : 190180087
PRODI : SISTEM INFORMASI

JURUSAN ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS MALIKUSSALEH
LHOKSEUMAWE

2023

LEMBAR PERNYATAAN ORISINALITAS

Saya yang bertanda tangan tangan dibawah ini:

Nama : Bagas Aulia Alfasyam

NIM : 190180087

Fakultas/Jurusan : Teknik / Teknik Elektro

Dengan ini menyatakan skripsi yang berjudul:

Pengembangan Aplikasi Pengetahuan Bahasa Pemograman Dasar Dan Lanjut Berbasis *Android* Menggunakan Metode *Game Development Life Cycle*

adalah hasil kerja tulisan saya sendiri didampingi dosen pembimbing bukan hasil plagiat dari karya tulis ilmiah orang lain.

Demikian surat pernyataan ini saya buat dengan sebenarnya, jika dikemudian hari ternyata terbukti bahwa skripsi yang saya tulis adalah plagiat, maka saya bersedia menerima sanksi sesuai aturan yang berlaku, dan saya bertanggung jawab secara mandiri tidak ada sangkut pautnya dengan Dosen Pembimbing dan kelembagaan Fakultas Teknik Universitas Malikussaleh.

Lhokseumawe, 30 September 2023

Penulis,

Bagas Aulia Alfasyam

NIM. 190180087

LEMBAR PENGESAHAN PEMBIMBING

Judul Tugas Akhir	:Pengembangan Aplikasi Pengetahuan Bahasa Pemograman Dasar Dan Lanjut Berbasis *Android* Menggunakan Metode *Game Development Life Cycle*.
Nama Mahasiswa	: Bagas Aulia Alfasyam
NIM	: 190180087
Tanggal Sidang	: 12 Oktober 2023

Bukit Indah, 07 November 2023

Penulis,

Bagas Aulia Alfasyam
NIM. 190180087

Menyetujui,-

Pembimbing Utama	Pembimbing Pendamping
	
Mochamad Ari Saptari, S.Kom., M.Kom	**Himmatur Rijal, S.T., M.Sc**
NIP. 198001052008121001	**NIP. 198810092022031006**

LEMBAR PENGESAHAN KOMISI PENGUJI

Telah disidangkan pada

Tanggal 12 Oktober 2023

KOMISI PENGUJI TUGAS AKHIR

Pembimbing Utama :
Mochamad Ari Saptari, S.Kom., M.Kom
NIP. 198001052008121001 ( )

Pembimbing Pendamping :
Himmatur Rijal, S.T., M.Sc
NIP. 198810092022031006 ( )

Penguji I :
Muthmainnah, S.Kom., M.Kom
NIP. 197711252006042007 ( )

Penguji II :
Zalfie Ardian, S.Kom., M.Eng
NIP. 198612032022031002 ( )

LEMBAR ACC CETAK

Pembimbing Utama :
Mochamad Ari Saptari, S.Kom., M.Kom
NIP. 198001052008121001 ( )

Pembimbing Pendamping :
Himmatur Rijal, S.T., M.Sc
NIP. 198810092022031006 ( )

Penguji I :
Muthmainnah, S.Kom., M.Kom
NIP. 197711252006042007 ( )

Penguji II :
Zalfie Ardian, S.Kom., M.Eng
NIP. 198612032022031002 ( )

LEMBAR PENGESAHAN SKRIPSI

Judul Skripsi	:	Pengembangan Aplikasi Pengetahuan Bahasa Pemograman Dasar Dan Lanjut Berbasis Android Menggunakan Metode Game Development Life Cycle
Nama Mahasiswa	:	Bagas Aulia Alfasyam
NIM	:	190180087
Program Studi	:	S1 Sistem Informasi
Jurusan	:	Teknik Elektro
Fakultas	:	Teknik
Perguruan Tinggi	:	Universitas Malikussaleh
Pembimbing Utama	:	Mochamad Ari Saptari, S. Kom., M.Kom
Pembimbing Pendamping	:	Himmatur Rijal, S.T., M.Sc
Ketua Penguji	:	Muthmainnah, S.Kom., M.Kom
Anggota Penguji	:	Zalfie Ardian, S.Kom., M.Eng

Lhokseumawe, 4 Desember 2023
Penulis,

Bagas Aulia Alfasyam
NIM 190180087

Menyetujui:

Pembimbing Utama,

Mochamad Ari Saptari, S.Kom., M.Kom
NIP 198001052008121001

Pembimbing Pendamping,

Himmatur Rijal, S.T., M.Sc
NIP 198810092022031006

Mengetahui:

Ketua Jurusan

Prof. Dr. Ir. Dahlan Abdullah, S.T., M.Kom., IPU., ASEAN Eng
NIP 197602282002121005

Koordinator Program Studi,

Rizky Putra Fhonna, S.T., M.Kom
NIP 197705062005012003

ABSTRAK

Dalam konteks banyaknya pembelajaran pemrograman melalui *website* atau *bootcamp*, muncul inovasi "*Learn and Battle about Code*," sebuah aplikasi pembelajaran bahasa pemrograman berbasis *Android* menggunakan *Unity 3D*. Aplikasi "*Learn and Battle about Code*" merupakan *game* pembelajaran bahasa pemrograman berbasis *Android* yang inovatif dan interaktif. Tujuan utama aplikasi ini adalah menyediakan pengalaman belajar yang menyenangkan dan kompetitif dalam mempelajari bahasa pemrograman dasar dan lanjut. Melalui kombinasi elemen permainan dan pembelajaran, diharapkan mampu meningkatkan minat dan efektivitas pembelajaran bahasa pemrograman. Pengembangan aplikasi menggunakan metode *Game Development Life Cycle* (GDLC) untuk memastikan efisiensi dan kualitas program. Hasil uji coba menunjukkan bahwa game berjalan dengan baik pada resolusi layar tertentu dan mendapat penilaian positif dari pengguna dengan skor rata-rata 72 pada *System Usability Scale* (SUS), menandakan penerimaan yang baik, khususnya dari mahasiswa yang memiliki pengetahuan tentang pemrograman. Kesimpulannya, aplikasi ini berhasil menciptakan solusi pembelajaran bahasa pemrograman yang inovatif, efektif, dan dapat diterima dengan baik oleh pengguna.

Kata Kunci: *game*, pembelajaran pemograman, *GDLC, Unity Engine 3D*, SUS

ABSTRACT

In the context of many programming lessons through websites or bootcamps, there is an innovation "Learn and Battle about Code," an Android-based programming language learning application using Unity 3D. "Learn and Battle about Code" is an innovative and interactive Android-based programming language learning game. The main objective of this app is to provide a fun and competitive learning experience in learning basic and advanced programming languages. Through the combination of game and learning elements, it is expected to increase the interest and effectiveness of learning programming languages. The application development uses the Game Development Life Cycle (GDLC) method to ensure the efficiency and quality of the program. The test results show that the game runs well on certain screen resolutions and received positive ratings from users with an average score of 72 on the System Usability Scale (SUS), signaling good acceptance, especially from students who have knowledge of programming. In conclusion, this application succeeded in creating an innovative, effective, and well-accepted programming language learning solution for users.

Keywords: game, learning programming, GDLC, Unity Engine 3D, SUS

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh.

Alhamdulillahirobbil Alamin puji dan syukur penulis panjatkan kehadirat Allah Subhanahu Wa Ta'ala atas rahmat dan hidayah nya sehingga penulis dapat menyelesaikan proposal tugas akhir ini. Shalawat bertangkaikan salam penulis sanjung saji kan kepada junjungan alam Nabi Besar Muhammad SAW, yang telah membawa ummat dari alam kegelapan menuju alam yang terang benerang dengan ilmu pengetahuan.

Dalam kesempatan ini, penulis ingin mengucapkan terima kasih kepada semua pihak yang telah berkenan membantu pada tahap penyusunan proposal tugas akhir ini khususnya kepada.

1. Bapak Prof. Dr. Herman Fithra, ST., MT., IPM selaku Rektor Universitas Malikussaleh.
2. Bapak Dr. Muhammad, ST., M.Sc selaku Dekan Fakultas Teknik Universitas Malikussaleh.
3. Bapak Rizky Putra Phonna, ST., M.Kom selaku Ketua Prodi Sistem Informasi Fakultas Teknik Universitas Malikussaleh.
4. Bapak Mochamad Ari Saptari, S. Kom., M.Kom selaku pembimbing I
5. Bapak Himmatur Rijal, S.T., M.Sc selaku pembimbing II
6. Seluruh Dosen dan Staf Jurusan Sistem Informasi Universitas Malikussaleh.
7. Orang tua tercinta yang mendoakan, menyemangati, mendukung dan memotivasi saya.
8. Permata Wulandari selaku partner penulis yang telah membantu dan memberikan semangat kepada penulis dalam menyelesaikan tugas akhir ini
9. Muhammad Alana Fauzan, Erlangga Dwi Putra, Alif Fazilah Aziz selaku Sahabat penulis yang telah membantu dan memberikan semangat kepada penulis.

10. Seluruh teman-teman mahasiswa jurusan Sistem Informasi yang telah memberikan semangat dan motivasi. Serta beberapa pihak yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari laporan proposal tugas akhir ini tidak luput dari berbagai kekurangan. Penulis mengharapkan saran dan kritik demi kesempurnaan laporan proposal tugas akhir ini dapat memberikan manfaat bagi semua pihak yang membutuhkan. Akhir kata, semoga Tuhan Yang Maha Esa Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu.

Wassalamu'alaikum Warahmatullahi Wabarakatuh.

Lhokseumawe, September 2023

Penulis

Bagas Aulia Alfasyam
190180087

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengembangan aplikasi "*Learn and Battle about Code*" berbasis *Android* menggunakan *Unity 3D* didasari oleh kebutuhan akan media pembelajaran yang inovatif dan menyenangkan dalam mempelajari bahasa pemrograman dasar dan lanjut. Di era digital ini, keterampilan bahasa pemrograman menjadi krusial dalam dunia kerja. Namun, pembelajaran bahasa pemrograman sering dianggap membosankan dan sulit bagi pemula. Oleh karena itu, untuk mengatasi tantangan ini, pengembangan game "*Learn and Battle about Code*" diinisiasi sebagai solusi pembelajaran yang menyenangkan dan interaktif. Permainan ini menggabungkan elemen pembelajaran bahasa pemrograman dengan kompetisi yang menantang, mendorong pemain untuk belajar dan meningkatkan kemampuan mereka dengan semangat yang tinggi.

Pemilihan *platform* pengembangan *Unity 3D* sebagai basis aplikasi didasari oleh kemampuannya dalam menciptakan *game 2D* dan *3D*, serta dukungan yang luas untuk berbagai sistem operasi seperti Android. Hal ini memungkinkan pengembangan *game* yang kompatibel dengan beragam perangkat mobile, memperluas jangkauan pemain potensial. Keseluruhan pengembangan aplikasi "*Learn and Battle about Code*" di *Android* menggunakan *Unity 3D* dirancang untuk menyediakan solusi pembelajaran inovatif dan menyenangkan dalam mempelajari bahasa pemrograman dasar dan lanjut, dengan memberikan pengalaman belajar yang bisa dimainkan bersaing dengan pemain lain.

Dalam rangka mengatasi permasalahan pembelajaran bahasa pemrograman yang dianggap sulit dan kurang menarik, aplikasi "*Learn and Battle about Code*" diwujudkan sebagai solusi yang interaktif dan menghibur. Dalam permainan ini, aspek pembelajaran bahasa pemrograman diintegrasikan dengan tantangan kompetitif, menciptakan dorongan bagi pemain untuk belajar dan meningkatkan kemampuan mereka dengan cara yang lebih terlibat.

Untuk memperkuat dasar-dasar riset dan pengembangan, analisis kebutuhan pengguna dan preferensi dalam pembelajaran bahasa pemrograman telah dilakukan. Survei dan wawancara dengan calon pengguna, baik yang memiliki sedikit pengetahuan tentang pemrograman maupun yang sudah mahir, telah memberikan wawasan penting yang membimbing desain pengalaman pembelajaran dalam aplikasi ini.

Melalui penggabungan pembelajaran bahasa pemrograman dengan elemen kompetitif dalam suatu *game*, aplikasi ini diharapkan mampu meningkatkan minat dan efektivitas pembelajaran bahasa pemrograman, terutama bagi mereka yang merasa sulit memahami konsep-konsep tersebut. Dalam penelitian ini, kami akan mengevaluasi performa dan kompatibilitas aplikasi ini di berbagai perangkat *Android* serta menganalisis tingkat keterlibatan dan motivasi pemain. Hasil dari penelitian ini akan membantu menyempurnakan dan mengembangkan lebih lanjut aplikasi "*Learn and Battle about Code*", sehingga dapat memberikan pengalaman pembelajaran yang lebih baik dan lebih bermanfaat bagi para pengguna.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas yang telah di uraikan maka dapat dirumuskan permasalahannya antara lain :

1. Bagaimana merancang dan membangun mengunakan metode *Game Development Life Cycle* pada *game* pengetahuan bahasa pemograman "*Learn and Battle About Code*" ?
2. Bagaimana menerapi *game* Pengetahuan bahasa pemograman "*Learn and Battle About Code*" sebagai alat bantu dalam proses bermain dan belajar memahami bahasa pemograman dasar dan lanjut?
3. Bagaimana mengimplementasikan *game* Pengetahuan bahasa pemograman dasar dan lanjut "*Learn and Battle About Code*" berbasis android sehingga mampu dengan mudah di mengerti dan difahami?

1.3 Batasan Masalah

Melihat dari permasalahannya, maka penelitian ini dibuat beberapa asumsi dengan tujuan agar pembahasan menjadi lebih terarah serta membatasi permasalahan. Batasan masalah dari penelitian ini yaitu antara lain:

1. Aplikasi *game* pengetahuan Bahasa pemograman ini dimainkan oleh dua orang (*Multiplayer*).
2. Aplikasi *game* ini ditujukan untuk programmer pemula atau lanjut.
3. Aplikasi *game* ini memiliki dimensi yang berbeda yaitu tiga dimensi sebagai pembelajaran dan dua dimensi sebagai kompetisi kuis.
4. Aplikasi *game* ini menggunakan *Blender* sebagai desain karakter dan dibangun menggunakan *Unity 3D.*
5. *Game* ini dapat di unduh melalui *google playstore* dan *Itch.io*.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah diatas didapat beberapa tujuan yang ingin dicapai dalam pembuatan *game* Pengetahuan bahasa pemograman "*Learn and Battle About Code*" yaitu antara lain :

1. Membangun dan merancang *game* pengetahuan bahasa pemrograman "*Learn and Battle About Code*" dengan metode *Game Development Life Cycle* yang terdiri dari enam tahapan.
2. Mengukur efektivitas dan kualitas *game* pengetahuan bahasa pemrograman "*Learn and Battle About Code*" sebagai media belajar bahasa pemrograman dasar dan lanjut.
3. Mengimplementasikan *game* pengetahuan bahasa pemrograman dasar dan lanjut "*Learn and Battle About Code*" berbasis *android* sehingga mampu dengan mudah dimengerti dan difahami oleh pemain.

1.5 Manfaat Penelitian

Manfaat dari penelitian mengenai *game* Pengetahuan bahasa pemograman dasar dan lanjut yaitu antara lain:

1.5.1 Manfaat bagi peneliti

Dapat Melatih peneliti dalam menerapkan ilmu dengan membuat *game* pengetahuan Bahasa pemograman dasar dan lanjut.

1.5.2 Manfaat bagi Pengguna

a. Mengetahui dan mengulang kembali bahasa pemograman dasar dan lanjut.
b. Mengembangkan keterampilan pemecahan masalah, logika, dan kreativitas.
c. Meningkatkan aksesibilitas pembelajaran bahasa pemrograman.

1.6 Sistematika Penulisan

Sistematika penulisan berfungsi untuk lebih terarahnya penulisan laporan ini, maka sistematika penulisan laporan ini dibagi menjadi lima bab dan masing-masing bab menjadi sub-sub bab yang saling berhubungan. Penulisan masing-masing bab dapat dilihat sebagai berikut ini:

BAB I PENDAHULUAN

Pada bab ini penulis membahas tentang latar belakang, definisi masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

BAB II TINJAUAN KEPUSTAKAAN

Bab ini menjelaskan landasan teori dan penelitian terdahulu yang digunakan dalam pengolahan masalah penelitian.

BAB III METODOLOGI PENELITIAN

Bab ini berisikan tentang tempat dan jadwal penelitian, teknik pengumpulan data, alat dan bahan, metode pengembangan sistem, prosedur alur penelitian, dan gambaran perencanaan singkat *system UI* dalam pembuatan *game* Pengetahuan bahasa pemograman dasar dan lanjut.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini penulis akan menerapkan rancangan dan hasil dari metode yang dibuat pada penelitian.

BAB V KESIMPULAN DAN SARAN

Pada bab ini penulis akan menjelaskan kesimpulan dan saran dari penelitian yang dilakukan.

BAB II
TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 Pengertian *Game*

Game atau permainan memiliki dua pengertian menurut (Agustina et al., 2015) Berikut ialah :

1. Permainan adalah suatu aktifitas bermain secara murni yang di tujukan untuk mencari kebahagiaan tidak untuk kalah atau menang
2. Permainan adalah suatu aktifitas bermain yang di tujukan untuk kesenangan akan tetapi di dalam nya ada sebuah kemenangan dan kekalahan

Berdasarkan pengertian permainan ada 4 teori permainan yang dapat di klasifikasikan kepada beberapa bagiannya, Yaitu :

1. Permainan papan yang hanya terbatas untuk dua pemain, yang biasanya memakai sistem pencarian langkah yang di sebut *Number of Players*.
2. Pemain yang mendapatkan giliran, sebagai lapisan dalam suatu permainan dan melakukan pergiliran dalam suatu ronde di sebut *Plies, Move and Turns*.
3. Mendapatkan kemenangan dan pemenang hanya di dapatkan oleh satu pemain dengan kata lain 1 pemain yang menang pemain lain nya adalah kekalahan (*zero-sum*) di sebut *The Goal of the game.*
4. Pemain mengetahui semua aturan dalam permainan, contohnya adalah Catur. Setiap langkah yang di ambil akan terpengaruhi pada sesudah nya oleh karena itu pemain harus mengetahui aturan aturan dari *game* yang di sebut *Information* .

2.1.2 Permainan Laga

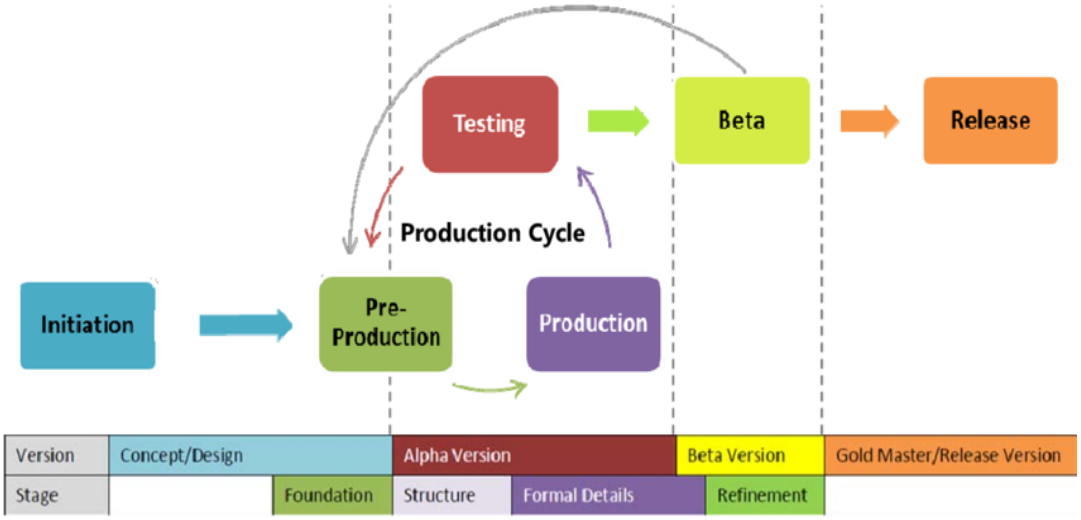
Menurut (Najuah et al., 2022) *Game* laga atau *action* adalah *game* yang memungkinkan pemain melakukan aksi laga pada karakter *game* nya, seperti melompat, menari, memukul, bertarung, menembak, yang berkaitan dengan menaklukan musuh atau menaklukan tantangan *game* tersebut.

Sejarah permainan laga dapat ditelusuri kembali ke masa pra-sejarah, di mana manusia sudah melakukan berbagai bentuk permainan yang mengandung unsur laga. Pada abad ke-19, permainan laga mulai dikembangkan secara sistematis dan digunakan sebagai sarana untuk meningkatkan kondisi fisik dan mental seseorang. Teori permainan laga mencakup berbagai aspek, seperti pengembangan keterampilan, pembelajaran motorik, pengembangan kesehatan, pembentukan karakter, dan pembentukan identitas sosial. Metodologi permainan laga meliputi berbagai jenis latihan dan teknik yang digunakan untuk meningkatkan keterampilan dan kondisi fisik seseorang. Ini termasuk latihan fisik, latihan teknik, dan latihan taktik. Aplikasi permainan laga meliputi berbagai bidang, seperti pendidikan, rekreasi, olahraga, dan pembentukan karakter. Permainan laga dapat digunakan untuk meningkatkan keterampilan fisik dan mental seseorang, serta untuk meningkatkan kesejahteraan dan kesehatan.

2.1.3 *Game Development Life Cycle*

Menurut (Andriyat Krisdiawan, 2019) Metode pengembangan sistematis yang digunakan dalam perancangan perangkat lunak ini menggunakan metodologi kerja *GDLC* (*Game Development Life Cycle*). *GDLC* adalah proses permainan yang menerapkan pendekatan iterative yang terdiri dari enam fase pengembangan dimulai dengan inisiasi/konsep, praproduksi, produksi, pengujian dan rilis. Keenam langkah tersebut dapat dikelompokan menjadi tiga proses utama, yaitu :

a. Proses inisiasi yang terdiri dari konsep dan desain.
b. Proses produksi terdiri dari praproduksi, produksi dan pengujian (*alpha* dan *beta*).
c. Publikasi



Gambar 2.1 Fase dan Proses *Game Development Life Cycle*

Metode GDLC yang digunakan dalam perancangan perangkat lunak ini bisa dipecahkan menjadi enam fase pengembangan, yaitu:

1. Inisiasi

Fase ini adalah fase awal yang bertujuan untuk menentukan ide dan visi permainan. Tim pengembang melakukan *brainstorming*, riset pasar, dan analisis kompetitor untuk menghasilkan konsep permainan yang menarik dan unik.

2. Praproduksi

Fase ini adalah fase persiapan yang bertujuan untuk membuat prototipe permainan yang menunjukkan konsep dan mekanik dasar permainan. Tim pengembang menggunakan alat-alat seperti *Unity*, *Unreal Engine*, atau *Game Maker* untuk mengembangkan prototipe yang dapat dimainkan dan diuji. Fase ini juga fase lanjutan dari konsep yang bertujuan untuk merancang spesifikasi teknis dan artistik permainan. Tim pengembang membuat dokumen desain permainan yang berisi detail tentang alur cerita, karakter, mekanik, grafik, suara, dan antarmuka permainan.

3. Produksi

Fase ini adalah fase pembuatan yang bertujuan untuk membangun aset permainan seperti *model 3D,* tekstur, animasi, musik, efek suara, dan kode. Tim pengembang menggunakan alat-alat seperti *Blender*, *Photoshop, Audacity*, atau *Visual Studio* untuk membuat aset yang berkualitas dan sesuai dengan desain permainan.

4. Pengujian.

Pengujian dalam konteks ini berarti bahwa pengujian internal dan eksternal dilakukan untuk menguji kegunaan *game*. Metode pengujian khusus untuk setiap fase prototipe.

a. *Alpha Testing*

Setelah produksi selesai, pengujian ini dijalankan oleh peneliti untuk melihat apakah masih ada *bug* dan apakah ada peluang untuk mengurangi atau menambah fitur. Jika ada *bug/error* atau penambahan fitur, peneliti akan memperbaikinya.

b. *Beta Testing*

Setelah *game* selesai dibuat, bukan berarti *game* Anda akan diterima oleh pemain. Peneliti melakukan pengujian eksternal yang dikenal sebagai pengujian *beta* yang dilakukan peneliti untuk menguji penerimaan *game* dan mendeteksi berbagai *bug* dan keluhan yang diajukan oleh penguji pihak ketiga. *Beta* sudah keluar dari siklus produksi, namun jika hasil pengujian ini masih menunjukkan potensi kesalahan, peneliti akan mengulang siklus produksi.

5. Rilis

Fase ini adalah fase peluncuran yang bertujuan untuk menyiapkan permainan untuk diluncurkan ke pasar. Tim pengembang melakukan tes akhir untuk memastikan tidak ada bug atau masalah yang tersisa. Tim pengembang juga membuat materi promosi seperti *trailer*, poster, dan situs web untuk menarik minat pemain. Tim pengembang juga mendapatkan persetujuan dari *platform* distribusi

seperti *Steam*, *Google Play Store*, atau *Apple App Store* untuk mempublikasikan permainan mereka.

2.1.4 Bahasa Pemograman

Bahasa Pemograman yang digunakan untuk memberitahu komputer apa yang harus dilakukan. Terdapat berbagai jenis bahasa pemrograman, seperti *C++*, *Java*, *Python*, dan lainnya, yang digunakan untuk menulis perangkat lunak dan aplikasi yang berbeda. Setiap bahasa pemrograman memiliki sintaks dan aturan yang berbeda-beda dan digunakan untuk tujuan yang berbeda-beda pula.

Menurut (Rochmawati et al., 2023) Bahasa pemrograman adalah instruksi standar untuk menginstruksikan komputer untuk mencapai fungsi yang diinginkan. Bahasa pemrograman ini adalah seperangkat aturan sintaksis dan semantik yang digunakan untuk mendefinisikan program komputer/laptop. Bahasa ini memungkinkan pemrogram untuk menentukan dengan tepat data apa yang akan digunakan dan diproses oleh komputer, bagaimana data itu akan disimpan dan dikirim, dan jenis tindakan apa yang akan dilakukan dalam sistem dalam situasi yang berbeda.

2.1.5 Sistem Operasi *Android*

Menurut (Frialdo et al., 2023) *Android* adalah sistem operasi telepon pintar atau *tablet* yang memiliki banyak fitur untuk mempermudah kehidupan manusia dan merupakan sistem operasi berbasis *Linux* yang terus berkembang dan dimodifikasi untuk perangkat seluler yang terdiri dari sistem operasi, *middleware*, dan aplikasi utama.

Sistem operasi berbasis *Linux* yang dikembangkan oleh *Google* untuk perangkat *mobile* seperti *smartphone* dan *tablet*. *Android* dirilis sebagai perangkat lunak sumber terbuka dan saat ini menjadi salah satu sistem operasi *mobile* yang paling populer di dunia. Pada awalnya dikembangkan untuk perangkat berbasis ARM, namun saat ini juga tersedia untuk perangkat lain seperti x86 dan MIPS.

Android menyediakan antarmuka pengguna yang intuitif dan mudah digunakan serta dukungan untuk aplikasi yang dapat diunduh dari toko aplikasi

resmi *Google*, *Google Play Store*. Selain itu, *Android* juga menyediakan dukungan untuk berbagai jenis sensor seperti *GPS*, kamera, dan sensor gerak, yang memungkinkan pengembangan aplikasi yang inovatif.

Android juga menyediakan dukungan untuk berbagai jenis konektivitas seperti *WiFi*, *Bluetooth*, dan *NFC*, yang memungkinkan perangkat untuk terhubung dengan perangkat lain atau akses internet dengan mudah. Sistem operasi ini juga menyediakan dukungan untuk berbagai jenis media seperti musik, vidio, dan gambar, yang memungkinkan pengguna untuk menikmati konten *multimedia* dengan mudah.

Secara umum, *Android* dikenal sebagai sistem operasi yang fleksibel dan mudah digunakan dengan dukungan yang luas untuk aplikasi dan perangkat keras. Ini juga memiliki komunitas pengembang yang aktif dan terus berkembang, yang memungkinkan adanya pengembangan aplikasi yang inovatif dan terus meningkatnya fitur-fitur yang tersedia di sistem operasi.

2.1.6 *Unity* 3D

Unity 3D perangkat lunak *game engine* yang digunakan untuk membuat dan mengembangkan permainan *2D* dan *3D*. *Unity* dapat digunakan pada berbagai *platform* seperti *Windows*, *MacOS*, *iOS*, *Android*, dan lainnya. Perangkat lunak ini menyediakan berbagai fitur seperti *rendering*, *physics*, *scripting*, dan animasi yang memungkinkan pengembang untuk membuat permainan yang menarik dan interaktif. *Unity* juga menyediakan integrasi dengan berbagai perangkat keras, seperti *VR* dan *AR*, yang memungkinkan pengembang untuk membuat permainan yang *immersif*. Selain itu, *Unity* memiliki komunitas yang besar dan aktif yang menyediakan bantuan dan sumber daya untuk pengembang.

Menurut (Nugroho et al., 2017) *Unity* adalah mesin *game universal*. *Unity* dapat dirilis sebagai *standalone* (*.exe*), berbasis *web*, berbasis *web*, *Android*, *iOS-iphone*, *XBOX* dan *PS3*. Meskipun *Unity* dapat dipublikasikan di berbagai *platform*, *Unity* memerlukan lisensi penerbitan untuk *platform* tertentu. Tetapi *Unity* menawarkannya untuk pengguna gratis dan dapat dipublikasikan dalam format *standalone* (*.exe*) dan *web*. *Unity* saat ini sedang dikembangkan berbasis i

(*Augmented Reality*). *Unity* memerlukan lisensi untuk mengaktifkan lisensi. Misalnya, jika Anda ingin mengaktifkan pengguna gratis, langkah pertama adalah mengunduh perangkat lunak secara gratis dari *www.unity3d.com*.

2.1.7 *Blender*

Perangkat lunak pemodelan, animasi, dan pembuatan *film* gratis dan sumber terbuka yang digunakan dalam industri hiburan, desain produk, dan pembuatan *video game* yang di sebut *Blender*. *Blender* memiliki fitur yang lengkap untuk pemodelan *3D*, *texturing*, animasi, simulasi, pembuatan *film*, dan *compositing*. *Blender* juga mendukung berbagai *format file* dan dapat diintegrasikan dengan perangkat lunak lainnya seperti *Photoshop* dan *After Effects*. Selain itu, *Blender* memiliki komunitas yang aktif yang menyediakan dukungan, *tutorial*, dan sumber daya untuk membantu pengguna dalam menggunakan perangkat lunak ini.

Menurut (Mulyono et al., 2012) *Blender* adalah perangkat lunak pemodelan animasi *3D* dengan mesin *game*. *Blender* awalnya dikembangkan oleh perusahaan animasi Belanda *NeoGeo* sebagai program animasi *in-house*. *Blender* telah tumbuh dan berkembang dengan proyek-proyek yang dikerjakan oleh *NeoGeo*. Tak lama setelah versi gratisnya dirilis di internet, *NeoGeo* dihentikan. Sekitar waktu ini, Ton Roosendaal *programmer* utama *Blender*, mendirikan sebuah perusahaan bernama *NOT a Number* untuk mengembangkan *Blender* lebih lanjut. *Blender* dapat berupa produk, versi gratis *Blender* bukanlah versi *demo* tetapi berfungsi penuh dan lisensi memungkinkan penggunaan tak terbatas untuk produksi komersial.

2.1.8 *Adobe Audition*

Adobe Audition adalah perangkat lunak pengedit *audio digital* yang dikembangkan oleh *Adobe Systems*. Ini banyak digunakan untuk mengedit, mencampur, dan menguasai *audio* untuk *video*, *podcast*, musik, dan kebutuhan produksi *audio* lainnya. Ini menawarkan berbagai alat dan fitur seperti pengeditan *multitrack*, analisis frekuensi dan berbagai efek dan opsi pemrosesan. Ini juga termasuk dukungan untuk *plugin VST* dan *AU*, serta integrasi dengan perangkat lunak *Adobe* lainnya seperti *Premiere Pro*. Audisi digunakan oleh para profesional di industri *audio*, serta penghobi dan amatir (Ramdhan et al., 2019).

2.2 Penelitian Terdahulu

Dalam penelitian ini, penulis juga meninjau beberapa penelitian sebelumnya untuk perbandingan dan referensi. Selanjutnya, hipotesis kesamaan dengan penelitian ini dihindari. Oleh karena itu, dalam kajian dokumen ini, penulis mencantumkan hasil penelitian terdahulu sebagai berikut:

Tabel 2.1 Penelitian Terdahulu

No	Peneliti	Tahun	Judul	Hasil Penelitian
1.	Intan Suraya	2022	Media Pembelajaran Komputer Dan Jaringan Berbasis *Game Android* Menggunakan Model Interaktif Untuk Meningkatkan Daya Tarik Dan Minat Siswa (Studi Kasus: SMKN 1 Stabat)	Hasil penelitian menunjukkan bahwa penggunaan media pembelajaran berbasis *game Android* dengan model interaktif dapat efektif meningkatkan daya tarik dan minat siswa dalam pembelajaran komputer dan jaringan. Siswa lebih tertarik dan terlibat dalam proses pembelajaran karena metode pembelajaran yang menyenangkan dan interaktif. Selain itu, media pembelajaran berbasis *game Android* juga memiliki kelebihan seperti mudah digunakan dan dapat diakses kapan saja dan di mana saja.
2.	Olivia Purnama	2022	Game Edukasi *Puzzle* Dan Tebak Gambar Menggunakan Metode *Fisher Yates Shuffle* Berbasis *Android*	Penelitian tentang "Game Edukasi Puzzle Dan Tebak Gambar Menggunakan Metode Fisher *Yates Shuffle* Berbasis Android" menunjukkan bahwa metode *Fisher-Yates* Shuffle merupakan pilihan yang baik untuk meningkatkan kualitas game edukasi *puzzle* dan tebak gambar berbasis Android. Algoritma acak membuat setiap sesi permainan unik dan menantang bagi pengguna, sehingga meningkatkan tingkat keterlibatan dan kepuasan. Hasil penelitian ini didapat melalui analisis literatur dan uji aplikasi.

3.	Doni Riadi	2022	Perancangan *Game RPG* (*Role Playing Game*) "*Signa*" Berbasis *Android*	Penelitian tentang *game* ini bertujuan untuk memberikan pengalaman hiburan bagi pemain dan pada saat yang sama juga mengedukasi tentang Corona *Virus Disease*. Aplikasi ini menggunakan metode *forward chaining* sebagai pedoman bagi pemain untuk membuka tantangan berikutnya. Selain itu, metode *finite state machine* juga digunakan untuk mengatur perilaku musuh dalam *game*, memberikan pengalaman bermain yang lebih realistis dan menantang bagi pemain.
4.	M. Herdiansyah Putra P.	2021	Pembuatan *Game Tower Defense* "*Heroes Conquest*" Menggunakan *Unity*	*Heroes Conquest* adalah *game Tower Defense* untuk desktop menggunakan *Unity 3D*. *Game* ini memiliki 2 *level* dan hasil pengujian menunjukkan permainan berfungsi dengan baik. Tingkat kelayakan permainan sebesar 61,33% (cukup layak) berdasarkan hasil analisis kuesioner. *Game* ini merupakan pilihan yang baik bagi pemain yang mencari *game Tower Defense* menyenangkan dan layak.
5.	Mega Budiwansyah, Malabay	2023	Pembuatan *Game Zombie Smasher* dengan *Unity* berbasis *Android*	Pembuatan *game Zombie Smasher* untuk *platform Android* memerlukan tingkat respon dan refleks yang sangat tinggi dari pemain. *Game* ini akan mudah dimengerti dan tidak akan membuat pemain mengalami kesulitan saat memainkannya. Skor yang tinggi dalam permainan sangat bergantung pada tingkat refleks dan respon pemain ketika menghadapi beberapa *zombie* yang muncul secara acak dan berjalan dengan sangat cepat. Dengan demikian, permainan ini akan memberikan pengalaman yang menantang bagi pemain dan membuat

				mereka merasa terlibat dalam permainan. Pemain harus memiliki tingkat respon dan refleks yang sangat baik agar dapat memainkan permainan ini dengan baik dan mencapai skor yang tinggi. Oleh karena itu, *game* ini sangat cocok bagi pemain yang mencari pengalaman permainan yang menantang dan menyenangkan.

2.3 Perbandingan Penelitian Terdahulu

Tabel 2.2 Tabel Perbandingan Penelitian Terdahulu

No	Judul	Perbandingan Penelitian
1.	Media Pembelajaran Komputer Dan Jaringan Berbasis *Game Android* Menggunakan Model Interaktif Untuk Meningkatkan Daya Tarik Dan Minat Siswa (Studi Kasus: SMKN 1 Stabat)	Keduanya memiliki kesamaan dalam hal pengembangan media pembelajaran berbasis *game Android*, namun berbeda dalam bidang yang difokuskan. Penelitian ini menitikberatkan pada pembelajaran komputer dan jaringan dengan menerapkan model interaktif untuk meningkatkan daya tarik dan minat siswa. Sementara penelitian "*Learn and Battle About Code*" lebih menekankan pada pembelajaran bahasa pemrograman dasar dan lanjut dengan menggunakan teknologi *Unity 3D*. Studi kasus pada penelitian ini dilakukan di SMKN 1 Stabat, sementara pada penelitian "*Learn and Battle About Code*" tidak memiliki lokasi tertentu.
2.	*Game* Edukasi *Puzzle* Dan Tebak Gambar Menggunakan Metode *Fisher Yates Shuffle* Berbasis *Android*	Penelitian ini memiliki fokus pada pengembangan *game* edukasi berbasis *Android* dengan tujuan untuk membuat *game* yang menyenangkan dan membantu pembelajaran. *Game* edukasi yang mengkombinasikan *puzzle* dan tebak gambar dikembangkan dengan menggunakan metode *Fisher Yates Shuffle* untuk membuat *game* lebih menyenangkan dan tidak monoton. Sementara *game* Pengetahuan bahasa pemrograman dasar dan lanjut dikembangkan dengan menggunakan *Unity 3D* untuk membuat *game* lebih menyenangkan dan memiliki grafik yang baik. Kedua *game* edukasi ini memiliki tujuan yang sama, yaitu untuk membantu pembelajaran dan dimainkan oleh pengguna pada perangkat *mobile*.

3.	Perancangan *Game RPG* (*Role Playing Game*) "*Signa*" Berbasis *Android*	Perbandingan dapat dilihat dari jenis *game*, basis teknologi, tema *game*, dan target pengguna. Jenis *game* yang difokuskan pada salah satu adalah *RPG*, sedangkan pada yang lain adalah *game* Pengetahuan bahasa pemrograman. Kedua penelitian sama-sama berbasis *Android* dan menggunakan *Unity 3D* sebagai *engine* untuk pembuatannya. Tema dari salah satu penelitian adalah *Signa*, sedangkan tema dari yang lain adalah *Learn and Battle About Code*. Target pengguna dari salah satu penelitian mungkin lebih cocok untuk penggemar *RPG*, sedangkan yang lain mungkin lebih cocok untuk penggemar *game* Pengetahuan dan pemrograman.
4.	Pembuatan Game Tower Defense "Heroes Conquest" Menggunakan Unity	Pembuatan *game tower defense* "*Heroes Conquest*" dan pengembangan *game* Pengetahuan bahasa pemograman dasar dan lanjut "*Learn and Battle About Code*" telah dilakukan menggunakan *Unity* sebagai *platform* pengembangan. *Game tower defense* "*Heroes Conquest*" dibuat dengan fokus pada pembuatan *game tower defense*, sedangkan *game* Pengetahuan bahasa pemrograman dasar dan lanjut "*Learn and Battle About Code*" dikembangkan dengan fokus pada pengembangan game Pengetahuan bahasa pemrograman dasar dan lanjut untuk *platform Android*. Kedua *game* tersebut dikembangkan dengan menggunakan *Unity* sebagai *platform* pengembangan.
5.	Pembuatan *Game Zombie Smasher* dengan *Unity* berbasis *Android*	Penelitian ini memiliki perbedaan dan keunikan masing-masing. Satu lebih spesifik pada pembuatan *game*, sedangkan yang lain lebih fokus pada pengembangan *game* yang memiliki fitur dan elemen yang lebih kompleks. Kedua penelitian menggunakan *platform* yang sama dan memiliki tujuan untuk pengembangan game pada sistem operasi yang sama. Walaupun memiliki perbedaan, kedua penelitian ini memiliki fokus yang sama yaitu pengembangan *game* untuk sistem operasi tersebut. Oleh karena itu, kedua penelitian ini dapat dibandingkan dalam hal fokus dan tujuan penelitian.

BAB III
METODOLOGI PENELITIAN

Pada bab ini, peneliti menjelaskan metode penelitian yang digunakan untuk menganalisis data dan rancangan membangun sebuah *game*. Bab ini dimulai dengan metode pengumpulan data, metodelogi pengembangan sistem, Prosedur alur penelitian, skema sistem, Teknik pengumpulan data, Analisa kebutuhan sistem, lokasi dan jadwal penelitian.

3.1 Metode Pengumpulan Data

1. Observasi
   Melaksanakan peninjauan secara langsung di kalangan programmer pemula atau lanjut untuk mendapatkan gambaran tentang pengetahuan Bahasa pemograman teratas.
2. Studi Pustaka
   Mencari data yang berhubungan dengan topik penelitian melalui buku literatur, artikel internet, majalah, maupun karya ilmiah.

3.2 Metodologi Pengembangan Sistem

Metodologi pengembangan *game* yang digunakan oleh penulis dalam melakukan penelitian ini adalah *Game Development Life Cycle* (*GDLC*)

Game Development Life Cycle merupakan sebuah metode perancangan dan pengembangan *game* yang melibatkan beberapa tahapan, mulai dari menentukan ide dan visi *game* hingga menyiapkan *game* untuk diluncurkan ke pasar. Tahapan-tahapan tersebut adalah inisiasi, pra-produksi, produksi, pengujian, alpha, pengujian beta, dan rilis. Dua tahap pertama yaitu inisiasi dan pra-produksi merupakan bagian dari perancangan *game*.

3.3 Inisiasi

Tahap inisiasi dimulai dengan merancang konsep dari permainan yang akan dibuat, termasuk cara di mana permainan akan dikembangkan. Hasil dari proses inisiasi ini adalah konsep permainan lengkap beserta deskripsi yang mendetail.

Pada tahap inisiasi, akan diuraikan skenario utama yang akan dihadirkan dalam permainan, karakter-karakter yang akan ada, serta cerita yang akan membentuk dasar permainan tersebut. Selain itu, tahap inisiasi juga mencakup penetapan target pemain yang dituju, platform di mana permainan akan dimainkan, dan mesin permainan (*game engine*) yang akan digunakan untuk mengembangkan permainan tersebut.

3.3.1 *Genre*

Genre yang akan diusung dalam permainan yang sedang direncanakan adalah perpaduan antara edukasi dan laga. Kombinasi *genre* ini memiliki tujuan utama untuk menyajikan pengalaman bermain yang tidak hanya menghibur, tetapi juga memberikan nilai edukatif kepada para pemainnya. Melalui permainan ini, pemain akan diberikan kesempatan untuk belajar dan memperoleh pengetahuan serta informasi yang bermanfaat.

Dalam menjalankan tujuan edukatif ini, permainan akan menghadirkan beragam elemen interaktif yang dirancang secara cerdas. Elemen-elemen tersebut akan membantu pemain untuk memahami konsep-konsep tertentu melalui tantangan-tantangan laga yang dihadirkan dalam permainan. Dengan demikian, pemain tidak hanya dapat merasakan kegembiraan dari aspek permainan laga, tetapi juga memperluas wawasan mereka dalam berbagai bidang pengetahuan.

Perlu dipastikan bahwa elemen laga dalam permainan tidak hanya bersifat hiburan semata, tetapi juga memiliki tujuan untuk mengintegrasikan konten edukatif secara halus ke dalam pengalaman bermain. Dengan cara ini, permainan akan berhasil mencapai keseimbangan antara kesenangan dan pembelajaran, menjadikan pemain lebih terlibat dan termotivasi untuk terus bermain sekaligus memperoleh pengetahuan baru.

Dalam merancang *genre* ini, aspek-aspek penting seperti skenario permainan, karakter-karakter yang terlibat, serta cerita yang dibawakan akan menjadi bagian integral dari pengalaman edukasi dan laga yang dihadirkan. Selain itu, penentuan target pemain yang tepat, pilihan platform yang sesuai, dan

penggunaan mesin permainan yang mendukung akan menjadi langkah-langkah strategis dalam menjalankan konsep ini dengan sukses.

3.3.2 Konsep

Konsep utama dari permainan ini melibatkan dua aspek yang saling terkait, yaitu Edukasi/Pembelajaran dan laga. Kedua konsep ini diintegrasikan dalam pengalaman bermain, menciptakan pengalaman yang unik dan bermanfaat bagi para pemain.

Permainan ini memusatkan perhatian pada pemahaman Bahasa pemrograman dan laga. Saat pemain memasuki ruangan yang berkaitan dengan Bahasa pemrograman tertentu, mereka akan menemui 6 ruangan yang berbeda. Setiap ruangan ini menyajikan *NPC* yang bertindak sebagai penghubung untuk pertempuran *online* antar pemain, serta perpustakaan yang dirancang khusus untuk proses pembelajaran Bahasa pemrograman.

Game ini menawarkan pilihan dari enam Bahasa pemrograman yang dapat dipelajari dan dikuasai, termasuk *Csharp, PHP, Golang, Kotlin, Javascript,* dan *Python*. Setiap Bahasa pemrograman memiliki tantangan dan misi yang sesuai dengan karakteristiknya sendiri. Melalui tantangan ini, pemain tidak hanya belajar mengenai Bahasa pemrograman yang mereka pilih, tetapi juga memiliki kesempatan untuk mengaplikasikan pengetahuan tersebut dalam pertempuran melawan pemain lain secara *online*.

Konsep ini menciptakan hubungan yang kuat antara pembelajaran dan pengalaman bermain. Pemain akan merasakan keseimbangan antara meningkatkan pengetahuan teknis dalam Bahasa pemrograman dan menghadapi aksi laga yang seru dalam pertempuran online. Dengan adanya elemen perpustakaan dan *NPC* penghubung, pemain juga dapat mengakses sumber daya edukatif secara langsung dan memperdalam pemahaman mereka tentang Bahasa pemrograman yang dipilih.

Dengan menggabungkan kedua konsep ini, permainan ini tidak hanya menghibur, tetapi juga memberikan nilai edukatif yang tinggi kepada pemainnya. Pemain akan merasakan pertumbuhan dalam keterampilan Bahasa pemrograman mereka seiring dengan perjalanan dalam dunia *laga* online yang penuh tantangan.

3.3.3 Ruang Lingkup

Ruang lingkup pengembangan *game* mencakup beberapa aspek yang saling terkait dan memiliki peran penting. Berikut adalah penjelasan mengenai masing-masing aspek tersebut:

1. *Autentikasi*

   Aspek *autentikasi* mengacu pada proses identifikasi dan validasi pemain sebelum mereka dapat mengakses *game*. Ini melibatkan pembuatan akun pengguna, pengelolaan kata sandi, dan mekanisme keamanan lainnya. *Autentikasi* adalah langkah awal yang penting untuk memastikan bahwa pemain memiliki izin dan hak akses yang tepat saat berinteraksi dengan game.
2. Panduan Permainan

   panduan permainan adalah elemen yang memberikan bantuan dan arahan kepada pemain dalam memahami game dan cara bermainnya. Ini bisa berupa tutorial interaktif, petunjuk visual, atau informasi yang membantu pemain untuk memahami mekanika permainan, kendali, dan tujuan utama. Panduan ini memastikan bahwa pemain memiliki pengalaman bermain yang lancar dan menyenangkan.
3. Pembelajaran

   Aspek pembelajaran mengacu pada bagaimana *game* mengintegrasikan elemen edukatif ke dalam pengalaman bermain. Ini bisa meliputi pemahaman tentang konsep, keterampilan, atau pengetahuan tertentu yang ingin disampaikan kepada pemain. Dalam konteks ini, pemain dapat belajar sambil bermain, meningkatkan pemahaman mereka tentang topik tertentu melalui tantangan atau skenario yang disediakan dalam permainan.
4. Laga

   Aspek laga merujuk pada unsur aksi atau pertempuran yang dihadirkan dalam *game*. berupa pertarungan melawan pemain lain secara *online*, Pertempuran ini dapat menjadi fokus utama dalam gim atau sekadar salah satu elemen yang mendukung pengalaman bermain. Aspek laga memberikan elemen adrenalin dan keseruan dalam permainan.

Kesemuanya ini bekerja bersama untuk membentuk pengalaman bermain yang holistik dan menarik. *Autentikasi* memberikan keamanan bagi pemain, panduan permainan membantu mereka memahami mekanika, pembelajaran memberikan nilai edukatif, dan aspek laga memberikan tantangan dan hiburan. Dengan mengintegrasikan elemen-elemen ini dengan cermat, pengembang dapat menciptakan *game* yang menggabungkan aspek-aspek yang berbeda untuk menciptakan pengalaman yang kaya dan memuaskan bagi para pemainnya.

3.3.4 Target Pemain

Target audiens untuk permainan ini adalah para programmer, baik yang pemula maupun yang sudah memiliki tingkat lanjut. Walaupun demikian, fokus utama *game* ini difokuskan khusus pada para programmer. Dalam pengembangan *game* ini, pengalaman dan pengetahuan dalam pemrograman menjadi aset yang sangat berharga. *Game* dirancang untuk memberikan tantangan serta peluang pembelajaran yang relevan dengan dunia pemrograman.

Para programmer pemula akan mendapatkan manfaat dari panduan dan konten pembelajaran yang disediakan dalam *game*. Panduan ini akan membantu mereka memahami dasar-dasar Bahasa pemrograman yang berbeda serta memberikan panduan tentang cara memecahkan masalah dan menerapkan konsep-konsep pemrograman secara praktis.

Di sisi lain, programmer yang telah memiliki pengalaman lanjut juga akan menemukan nilai dalam *game* ini. Mereka dapat memanfaatkan *game* ini sebagai sarana untuk meningkatkan pemahaman tentang Bahasa pemrograman yang mungkin belum dikuasai sebelumnya atau untuk mengasah keterampilan *eksisting* mereka melalui tantangan-tantangan yang disediakan.

Karena *game* ini difokuskan pada para programmer, elemen-edukatif dalam *game* akan lebih mendalam dan berfokus pada pemrograman yang praktis. Ini akan membantu para programmer memperluas keterampilan mereka sambil tetap terlibat dalam pengalaman bermain yang menyenangkan dan menantang.

Dengan demikian, *game* ini mengakomodasi kedua kelompok audiens, baik pemula maupun yang lebih berpengalaman, dengan memberikan konten yang relevan dan menarik dalam konteks pemrograman.

3.3.5 *Platform*

Platform yang dipilih untuk permainan ini adalah *platform mobile Android*. Keputusan ini didasarkan pada pertimbangan bahwa penggunaan *smartphone* telah menjadi semakin umum dan meningkat dalam kehidupan sehari-hari. Fenomena ini mengakibatkan tingginya permintaan akan permainan yang dapat dimainkan melalui *smartphone* di Indonesia, yang pada gilirannya meningkatkan tingkat unduhan permainan melalui *platform* tersebut.

Dengan memilih *platform mobile Android*, permainan ini dapat mencapai basis pemain yang luas dan beragam di seluruh spektrum usia. *Fleksibilitas* dan kemudahan *aksesibilitas* dari perangkat *Android* memungkinkan permainan untuk diakses oleh berbagai lapisan masyarakat. Selain itu, *platform* ini juga memberikan ruang kreatif yang cukup besar bagi pengembang untuk menghadirkan konsep permainan edukasi dan laga dengan sentuhan inovatif yang sesuai dengan karakteristik perangkat *mobile*.

Dalam mengoptimalkan pengalaman permainan di *platform Android*, penting untuk mempertimbangkan berbagai aspek seperti antarmuka pengguna yang ramah perangkat sentuh. Dengan demikian, permainan dapat memberikan pengalaman bermain yang mulus dan memuaskan bagi para pemain yang memilih untuk menjelajahi dunia edukasi dan laga yang telah dirancang dengan cermat.

3.3.6 *Game Engine*

Dalam pengembangan permainan ini, dipilihlah *Unity Engine* sebagai *game engine* utama. Keputusan ini diambil berdasarkan beberapa pertimbangan yang mendasar. Pertama, *Unity Engine* menggunakan bahasa pemrograman *C#* dalam pembuatan permainan, yang merupakan bahasa yang cukup populer dan memiliki sintaks yang jelas, membuatnya lebih mudah dipahami oleh para pengembang, baik mereka yang baru belajar atau yang sudah berpengalaman.

Selain itu, *Unity Engine* juga menonjolkan antarmuka pengembangan yang intuitif dan *user-friendly*. Hal ini sangat penting, terutama dalam proyek permainan yang mencakup *genre* edukasi dan laga, karena pengembang dapat lebih fokus pada konten permainan dan konsep edukatif daripada menghabiskan waktu berlebihan untuk mengatasi kendala teknis dalam pengembangan.

Dalam keseluruhan, penggunaan *Unity Engine* sebagai *game engine* dalam proyek ini diharapkan dapat menghasilkan permainan yang memukau secara visual, berfungsionalitas tinggi, dan efektif dalam menyampaikan nilai edukatif melalui kombinasi *genre* edukasi dan laga yang unik.

3.4 *Pra-produksi*

3.4.1 Perancangan *User Interface* (UI)

Pada subjudul ini peneliti menjelaskan gambaran singkat tentang perencanaan sistem *UI* dari *game* yang dibangun pada *platform Android*, berikut beberapa gambaran singkat tentang perencanaan antarmuka pengguna:

1. *Splashscreen*

Gambar 3.1 Halaman *Splashscreen*

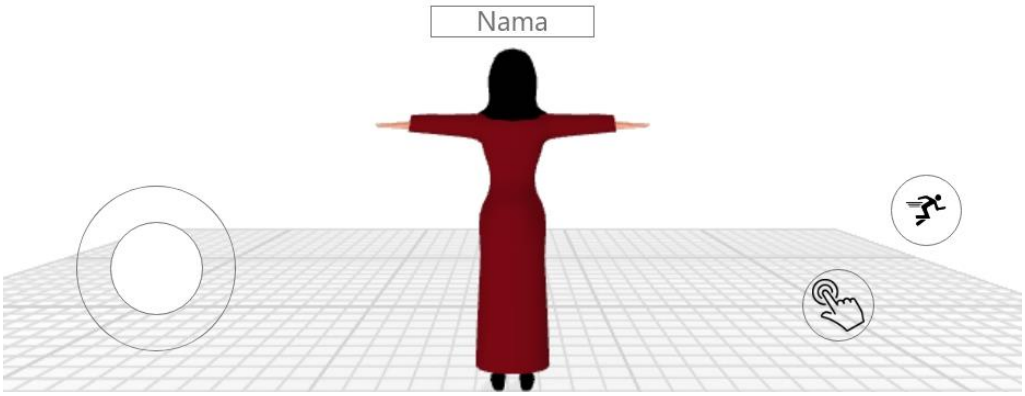
Dengan adanya *splashscreen* ini, pemain dapat mengetahui identitas pembuat game dan membantu meningkatkan *brand awareness* dari Universitas Malikussaleh.

2. Halaman *Menu*

Gambar 3.2 Halaman *Menu*

Berikut adalah halaman menu yang di isikan judul dari *game* Pengetahuan ini yaitu *Learn and Battle About Code* sebelum dimulai isikan nama terlebih dahulu untuk memasuki *open world*.

3. Halaman *User* Dunia Terbuka



Gambar 3.3 Halaman *User* Dunia Terbuka

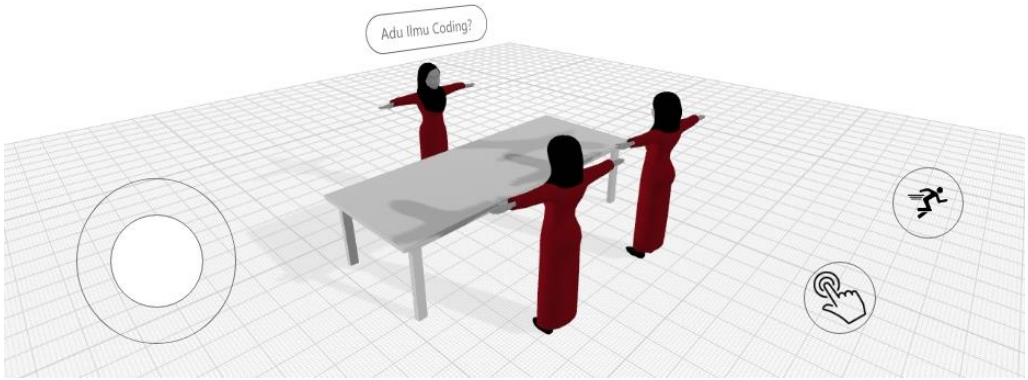
Berikut adalah halaman *user open world* dengan *controller* yang sederhana yaitu lari dan aksi. Yang digunakan untuk berkelana pada dunia *3D* untuk mencari ilmu bahasa pemograman.

4. Halaman Tentang Belajar Bahasa Pemograman

Gambar 3.3 Halaman Belajar Bahasa Pemograman

Dengan halaman ini, pemain dapat mempelajari tentang ilmu pemrograman seperti *Python* secara interaktif. Buku dalam permainan dapat memberikan penjelasan dan contoh dalam bentuk *3D*, sehingga pemain dapat memvisualisasikan dan memahami materi dengan lebih baik. Ini merupakan cara yang menyenangkan dan efektif untuk mempelajari ilmu pemrograman dan bisa membantu pemain meningkatkan kemampuan mereka dalam hal ini pemrograman *Python*.

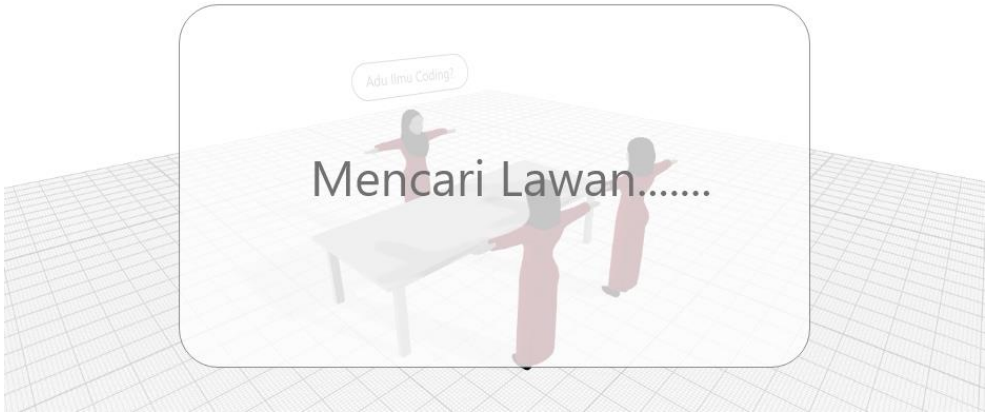
5. Halaman Pertandingan
    a. *Lobby* Pertandingan



Gambar 3.4 Halaman *lobby* pertandingan

Berikut adalah halaman *lobby* pertandingan yang di awali oleh *NPC* yang berguna untuk memulai pertandingan dan memasuki pencarian lawan

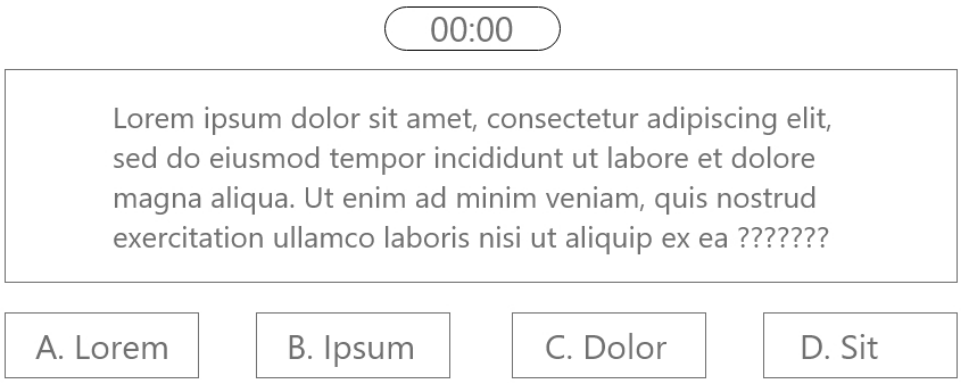
b. Pencarian Lawan



Gambar 3.5 Halaman pencarian lawan

Halaman pencarian lawan dalam *game*. Saat bermain, pemain harus menunggu lawan untuk memulai pertandingan. Jika lawan tidak ada, maka pemain dapat menunggu atau menutup halaman pencarian lawan.

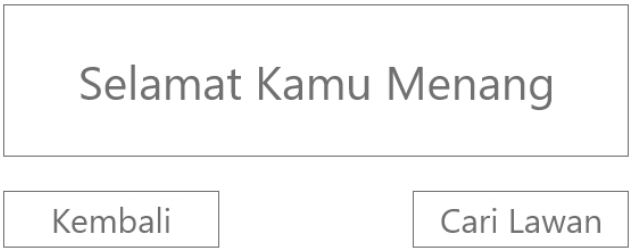
c. Pertandingan



Gambar 3.6 Halaman pertandingan

halaman pertandingan yang menawarkan pilihan ganda sebagai cara untuk mempermudah proses menjawab bagi pengguna.

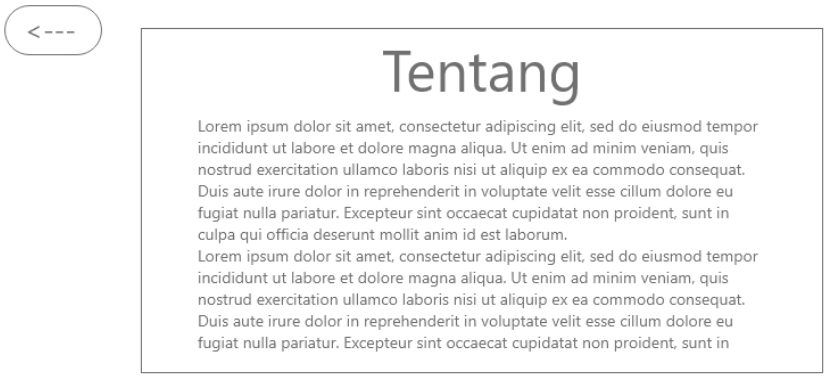
d. *Condition Win/Lose*



Gambar 3.7 Halaman kondisi menang atau kalah

Halaman tentang kondisi dari perlawanan yang sengit dan akan di hasili menang atau kalah, maka dari itu *user* harap mempelajari materi terlebih dahulu untuk mencapai kemenangan.

6. Halaman Tentang



Gambar 3.8 Halaman Tentang

Berikut adalah halaman tentang yang di isikan tentang identitas dari pembuat permainan Pengetahuan bahasa pemograman.

7. Halaman Keluar

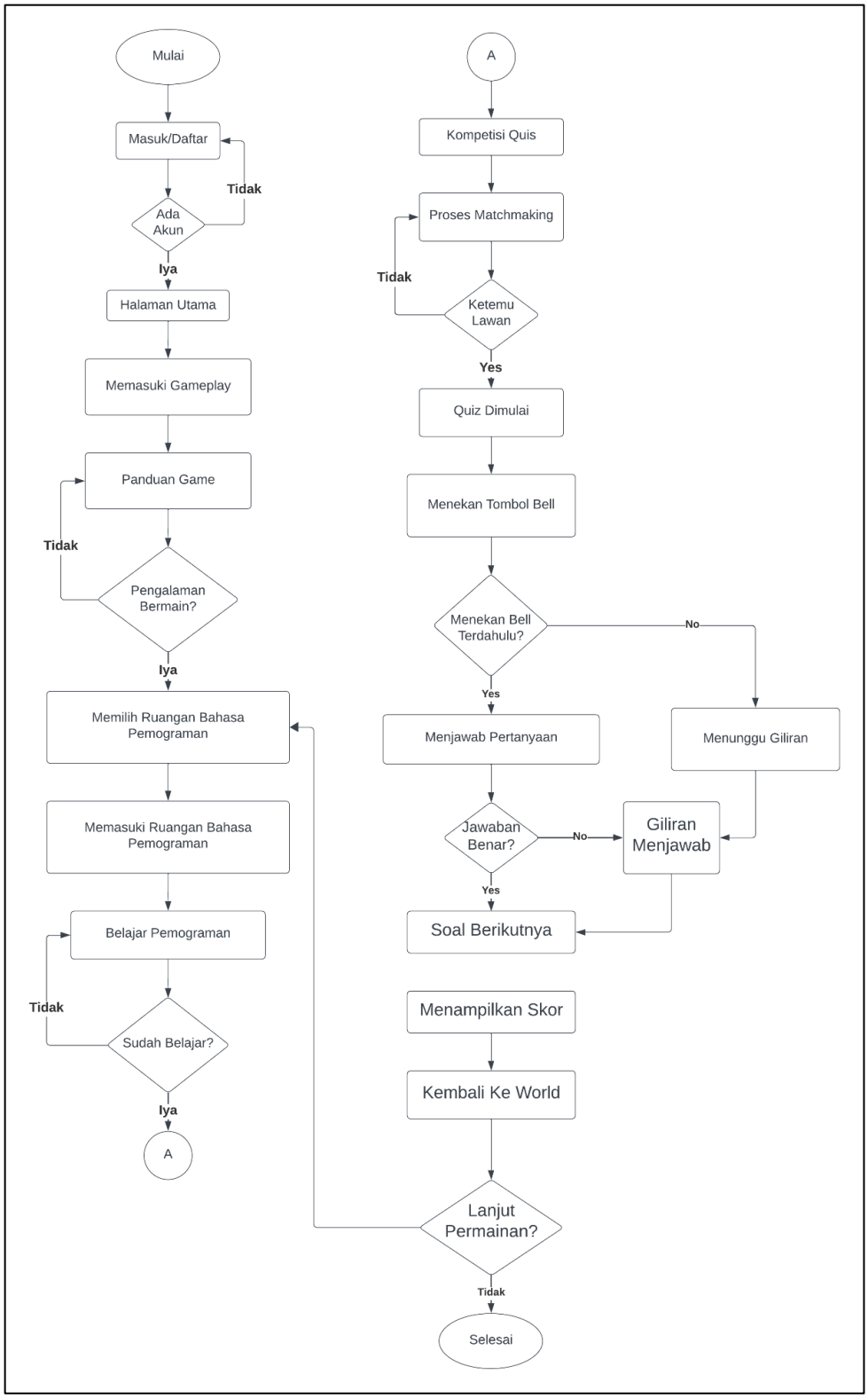
Gambar 3.9 Halaman keluar

Halaman Keluar merupakan halaman yang akan muncul ketika *user* menekan tombol keluar pada halaman *menu*. Terdapat dua tombol yaitu "iya" untuk keluar dari aplikasi dan tombol "tidak" untuk tetap pada aplikasi.

3.4.2 *Gameflow*

Gambar di bawah ini menjelaskan alur dari permainan ini. Permainan dimulai dengan langkah Masuk/Daftar, di mana pemain harus masuk ke akun yang sudah ada atau membuat akun baru. Jika akun sudah ada, maka pemain akan lanjut ke halaman beranda. Jika akun belum ada, maka pemain harus daftar terlebih dahulu. Di halaman utama, pemain dapat memilih untuk memasuki ruangan bahasa pemrograman yang diinginkan. Di dalam ruangan, pemain dapat memilih untuk belajar atau berkompetisi dalam kuis. Jika pemain sudah berpengalaman, maka pemain dapat langsung berkompetisi. Jika pemain belum berpengalaman, maka pemain harus belajar terlebih dahulu untuk berkompetisi, pemain harus mencari lawan melalui sistem *matchmaking*. Jika lawan sudah ditemukan, maka kuis akan dimulai. kuis terdiri dari beberapa soal yang harus dijawab oleh dua pemain secara bergantian. Pemain yang lebih cepat menekan bel akan mendapatkan kesempatan untuk menjawab terlebih dahulu. Jika jawaban benar, maka pemain akan mendapatkan poin. Jika jawaban salah, maka lawan akan mendapatkan kesempatan untuk menjawab. Setelah semua soal selesai, hasil dari kuis akan ditampilkan.

Pemain dapat kembali ke ruangan bahasa pemrograman untuk belajar atau berkompetisi lagi, atau keluar dari permainan.



Gambar 3.10 *Gameflow* Permainan

3.4.3 Aset

Untuk membangun sebuah *game*, diperlukan berbagai macam aset yang akan digunakan untuk menciptakan visualisasi *game* tersebut. Aset-aset ini meliputi karakter utama yang akan dimainkan oleh pemain, karakter *non-playable (NPC)* yang ada di dalam dunia game, peta atau *NPC*, *Texture*, *Material*, serta beragam objek pendukung yang akan memperkaya pengalaman bermain *game*. Selain itu, aspek *audio* juga sangat penting dalam *game*, termasuk suara-suaranya, seperti suara opening dan ending *game*, suara tombol, serta suara langkah kaki karakter.

Aset-aset yang mencakup karakter utama, karakter *NPC*, peta, dan objek seperti yang disebutkan di atas, semuanya dapat ditemukan dan digunakan dari sumber daya yang tersedia di *Unity*, sebuah *platform* pengembangan game yang populer. Rincian terkait aset karakter dan pergerakan karakter, tombol-tombol yang digunakan dalam game, objek-objek interaktif, serta desain peta dan desain kompetisi kuis, dapat ditemukan dalam Tabel XX sampai Tabel XX dalam dokumentasi proyek pengembangan game ini.

1. Character

Tabel 3.1 Asset Karakter

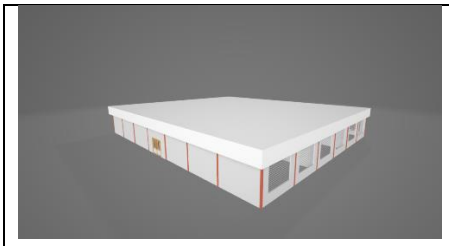
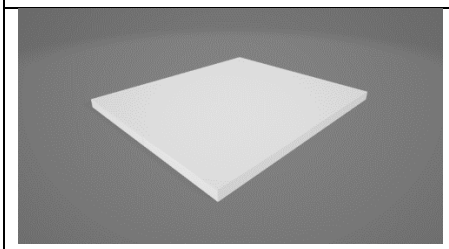
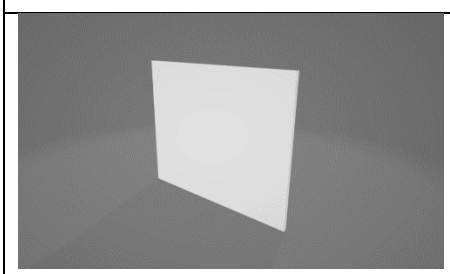
Gambar	Nama	Keterangan
	*Player*	*3D model* karakter utama

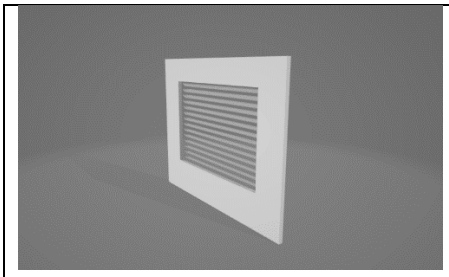
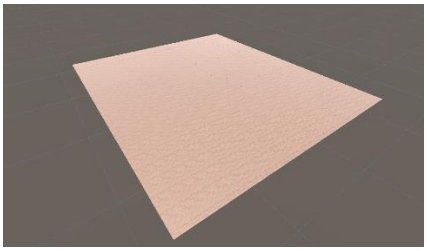
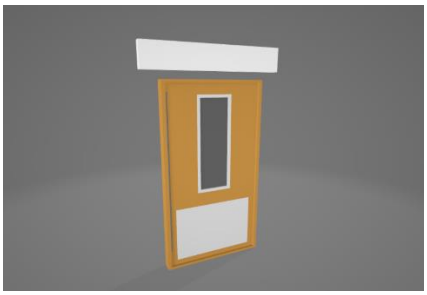
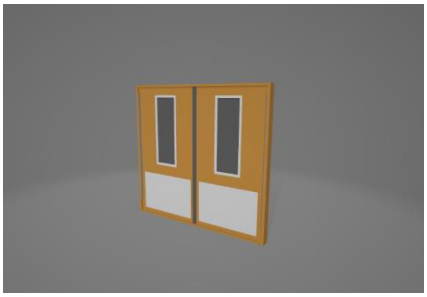
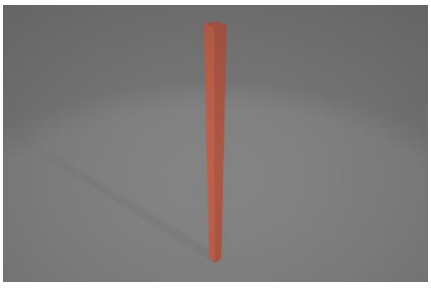
	*NPC Guide*	*NPC* yang pengarah *game*
	*NPC Python*	*NPC* yang membawa *player* ke kompetisi kuis pemograman *python*
	*NPC PHP*	*NPC* yang membawa player ke kompetisi kuis pemograman *PHP*
	*NPC Kotlin*	*NPC* yang membawa *player* ke kompetisi kuis pemograman *Kotlin*
	*NPC Javascript*	*NPC* yang membawa *player* ke kompetisi kuis pemograman *Javascript*

	*NPC Golang*	*NPC* yang membawa *player* ke kompetisi kuis pemograman *Golang*
	*NPC Csharp*	*NPC* yang membawa *player* ke kompetisi kuis pemograman *Csharp*

2. Perpustakaan

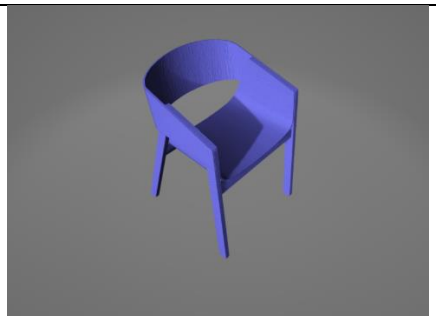
Tabel 3.2 Asset Bangunan Perpustakaan

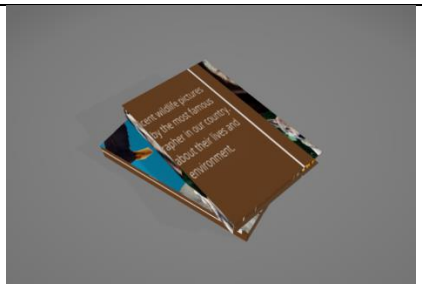
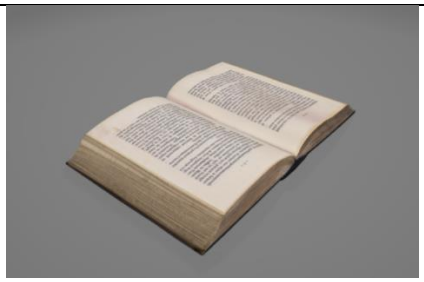
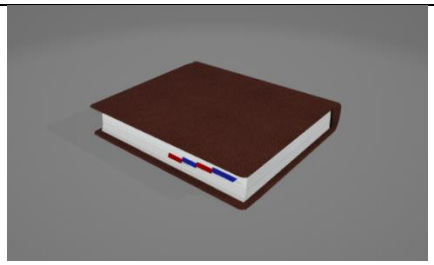
	Perpustakaan	Tampilan hasil Bangunan perpustakaan
	Atap perpustakaan	Atap yang menyeluruhi semua ruangan pemograman
	Dinding Perpustakaan	Dinding yang menyelimuti perpustakaan

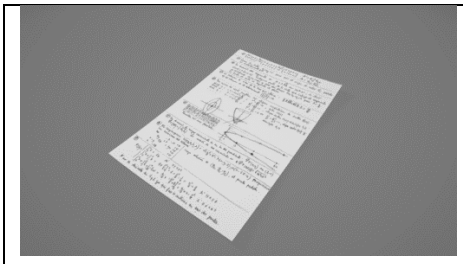
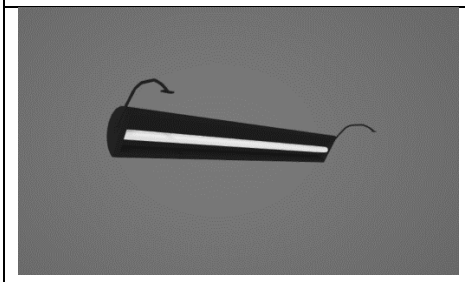
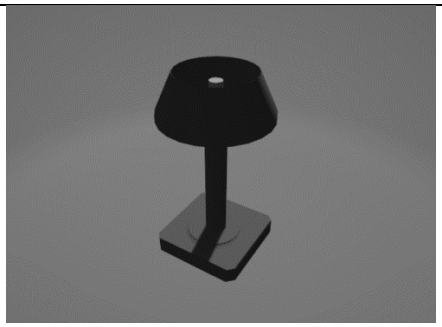
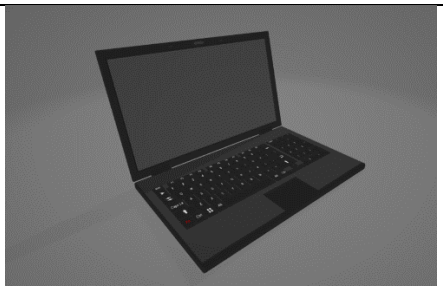
	Jendela Perpustakaan	Jendela yang terletak di sisi kanan kiri perpustakaan
	Lantai Perpustakaan	Lantai yang menyeluruhi satu perpustakaan
	Pintu 1 Perpustakaan	Pintu dan label nama Bahasa pemograman
	Pintu 2 Perpustakaan	Pintu utama perpustakaan
	Tiang Perpustakaan	Tiang sebagai selipan di antara dinding

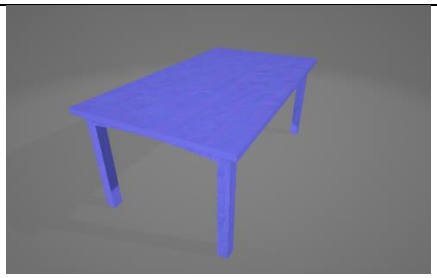
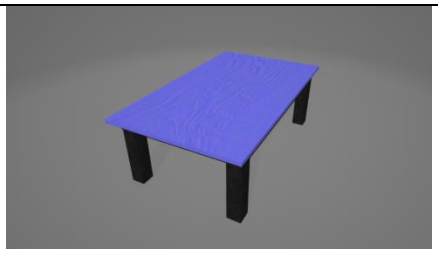
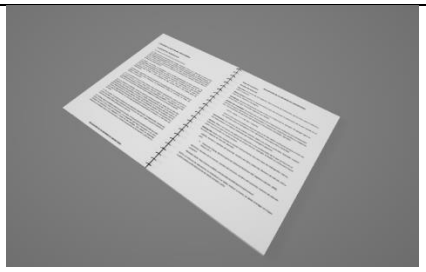
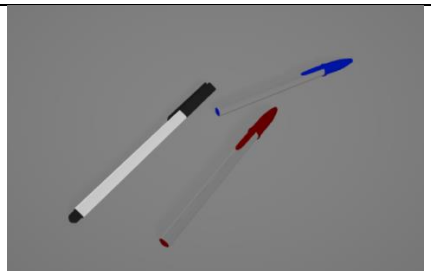
3. Komponen perpustakaan

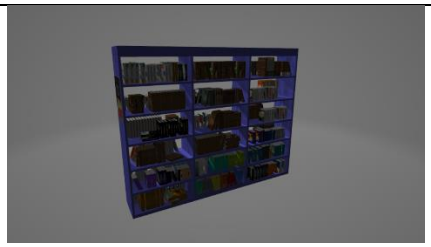
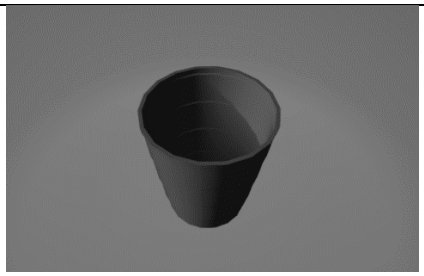
Tabel 3.3 Aset komponen perpustakaan

Gambar	Nama	Keterangan
	Kursi 1	Berupa *furniture* sebagai kursi untuk meja Panjang
	Kursi 2	Berupa *furniture* sebagai kursi tunggu
	Kursi Sofa	Berupa *furniture* sebagai kursi sofa untuk meja kecil
	Botol Minum	Benda seni yang terletak di meja Panjang

	Buku 1	Benda seni yang terletak di meja kecil
	Buku 2	Benda seni yang terletak di meja Panjang
	Buku 3	Benda seni yang terletak di meja kecil
	Buku 4	Benda seni yang terletak di meja kecil
	*Hand Sanitizer*	Benda seni yang terletak di meja Panjang
	Ponsel	Benda seni yang terletak di sebagian meja kecil

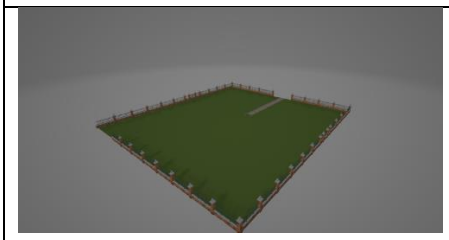
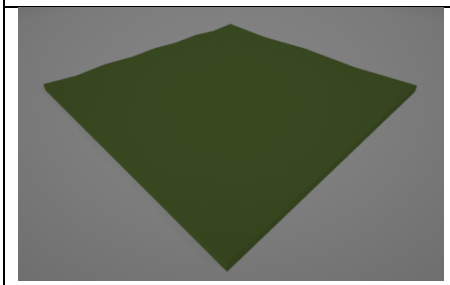
	Lembar Kertas	Benda seni yang terletak di meja kecil
	Koran	Benda seni yang terletak di meja Panjang
	Lampu lemari	Sebagai penerangan pencahayaan di rak lemari
	Lampu Meja	Penerangan cahaya untuk meja Panjang
	Laptop	Furniture sebagai hiasan meja Panjang dan meja NPC
	Buku Majalah	Perhiasan meja kecil

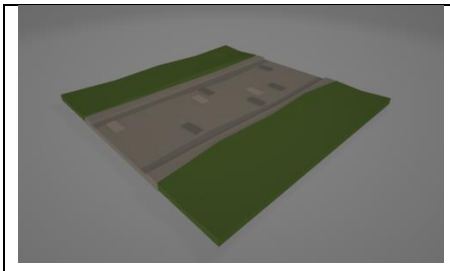
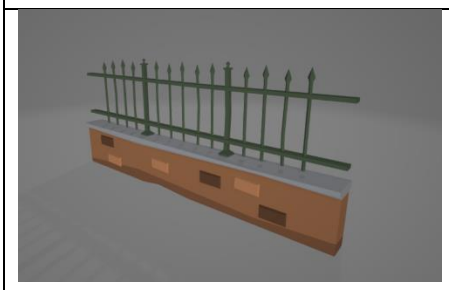
	Meja 1	Penampung furniture di tiap ruangan
	Meja 2	Penampung furniture di tiap ruangan
	Meja NPC	Pembatas antara NPC dan karakter utama
	Binder	Perhiasan meja yang terletak di Sebagian meja
	Poster 1	Perhiasan yang terletak di meja panjang
	Alat Tulis	Perhiasan yang terletak di meja Panjang

	Rak Buku	Rak buku sebagai interaksi player untuk belajar
	Tas Laptop	Perhiasan yang terletak di samping meja panjang
	Tong sampah	Perhiasan yang terletak di kursi tunggu

4. Halaman Luar Perpustakaan

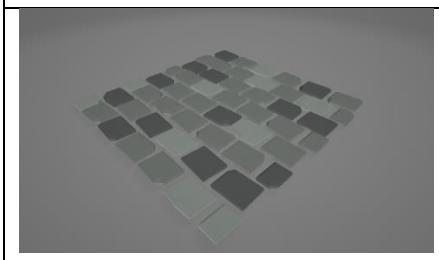
Tabel 3.4 Aset halaman luar perpustakaan

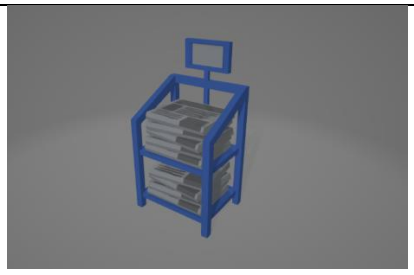
Gambar	Nama	Keterangan
	Halaman	Haula pergerakan player
	Rumput	Alas halaman luar

	Jalan rumput	Perhiasan yang terletak di sepanjang menuju perpustakaan
	Pagar	Pembatas area permainan
	Tiang Pagar	Perhiasan yang di antara 2 pagar

5. Komponen Luar Ruangan

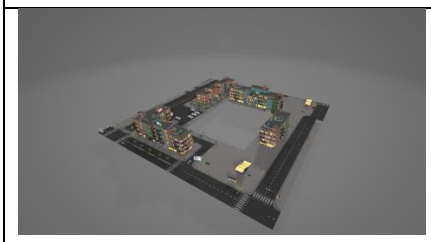
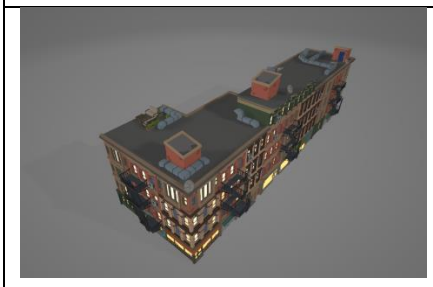
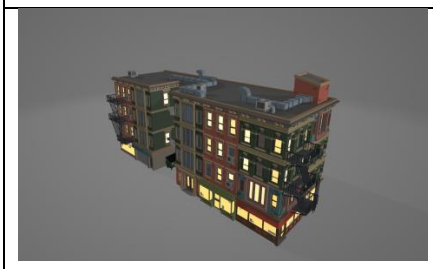
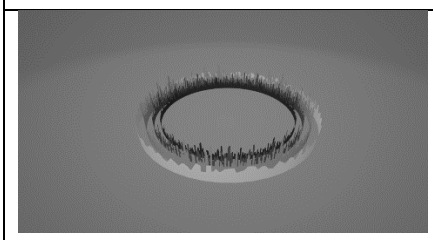
Tabel 3.5 Aset luar ruangan

Gambar	Nama	Keterangan
	Jalan Berbatu	Hiasan halaman luar
	Pohon 1	Memperindah suasanda luar ruangan

	Pohon 2	Memperindah suasanda luar ruangan
	Pohon 3	Memperindah suasanda luar ruangan
	Lampu Jalan	Memperindah suasanda luar ruangan
	Tong Sampah Besar	Memperindah suasanda luar ruangan
	Rak Koran	Memperindah suasanda luar ruangan
	Bangku Halaman	Memperindah suasanda luar ruangan

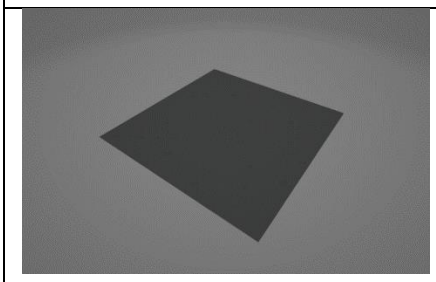
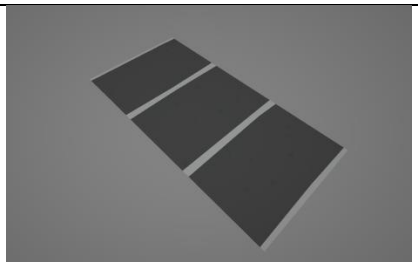
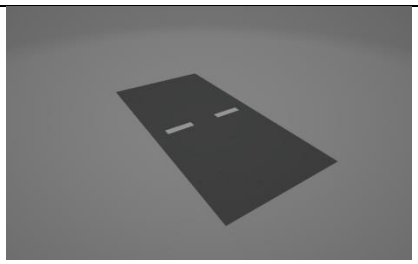
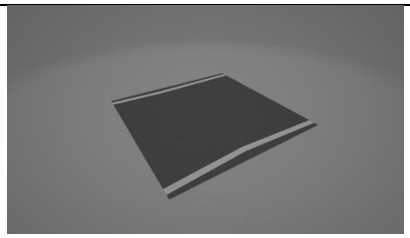
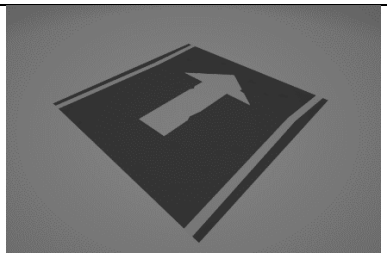
6. Kota

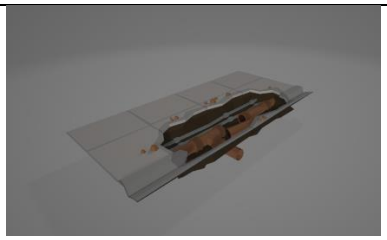
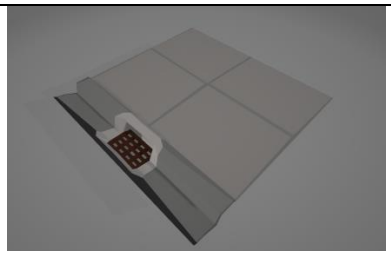
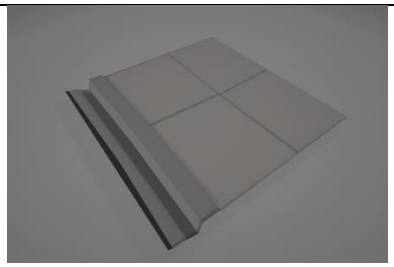
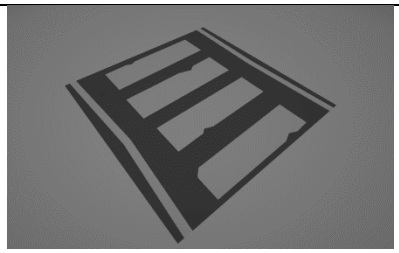
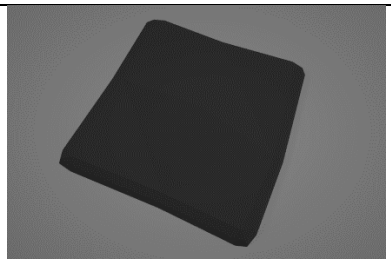
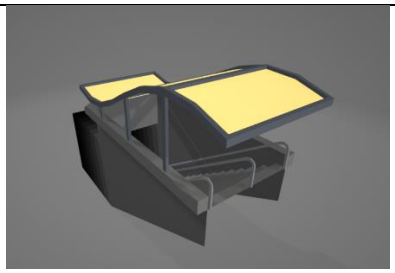
Tabel 3.6 Aset perkotaan

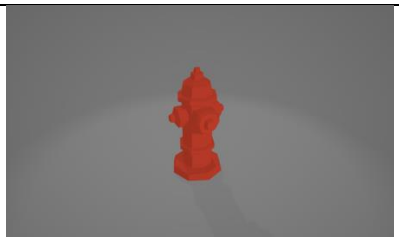
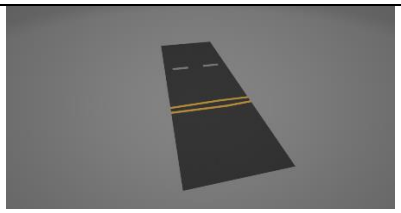
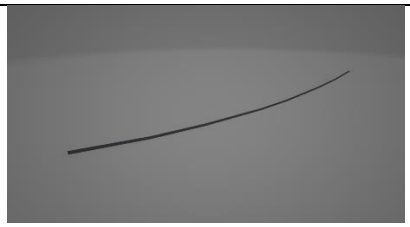
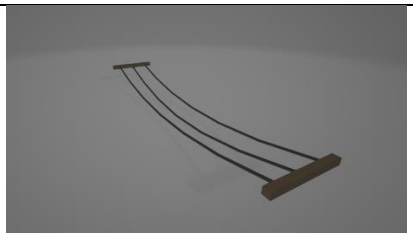
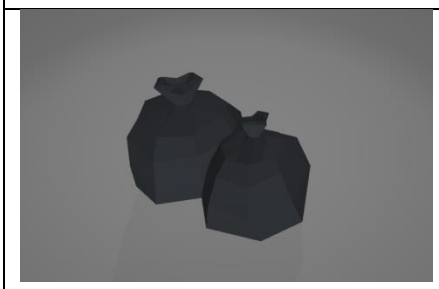
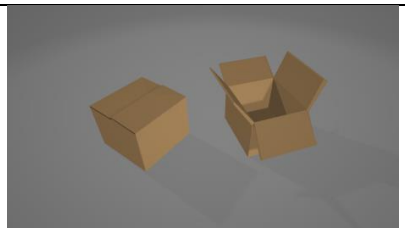
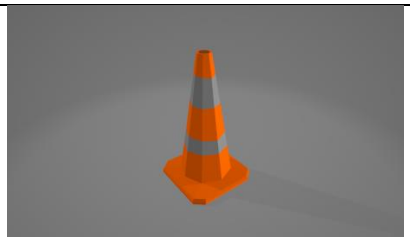
Gambar	Nama	Keterangan
	Kota	Kota yang tersusun menjadi 4 gedung
	Gedung belakang perpustakaan	Sebagai pemandangan selayak nya perkotaan
	Gedung depan perpustakaan	Sebagai pemandangan selayak nya perkotaan
	Gedung kanan Perpustakaan	Sebagai pemandangan selayak nya perkotaan
	Gedung kiri perpustakaan	Sebagai pemandangan selayak nya perkotaan
	Siluet sekeliling kota	Pemandangan di antara celah dari 4 gedung

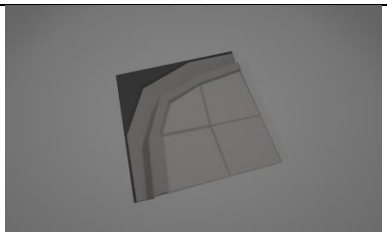
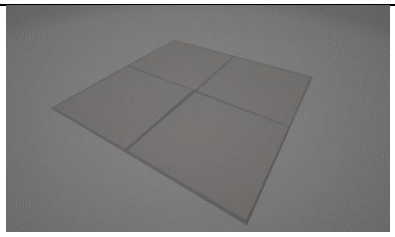
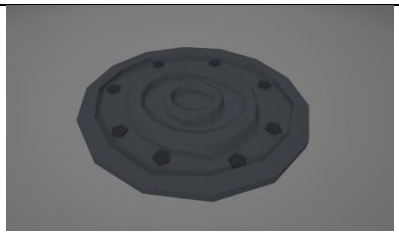
7. Komponen Kota

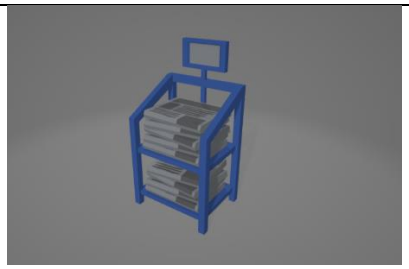
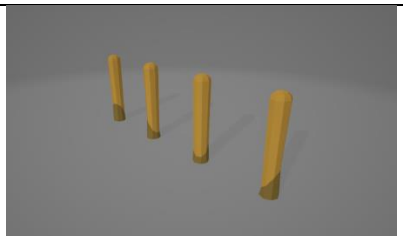
Tabel 3.7 Aset komponen kota

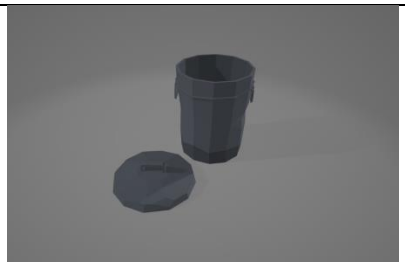
Gambar	Nama
	Aspal kosong
	Aspal parkiran
	Aspal
	Aspal 2 baris
	Aspal berpanah

	Perbaikan jalan
	Jalan berselokan
	Sisi pejalan
	Zebra cross
	Tambalan aspal
	Jalan bawah tanah

	Hidran
	Jalan Besar
	Kabel 1
	Kabel 3
	Kantong sampah
	Kardus
	Kerucut

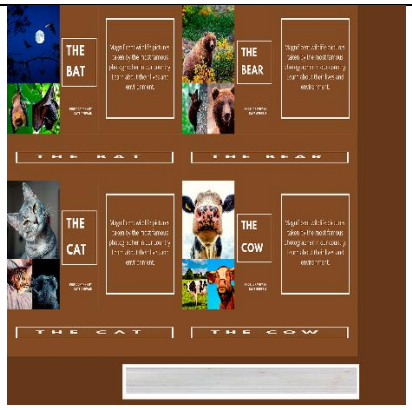
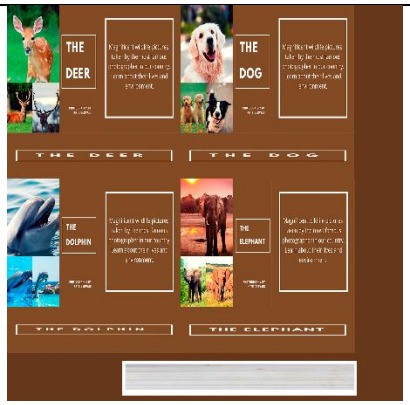
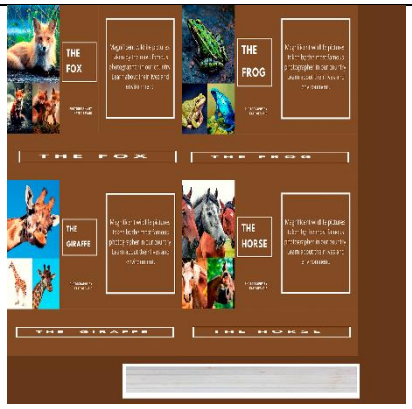
	Korner pejalan 1
	Kotak pesan
	Bangku
	Lantai jalan
	Lubang got
	Meteran parkir
	Tanda jalan

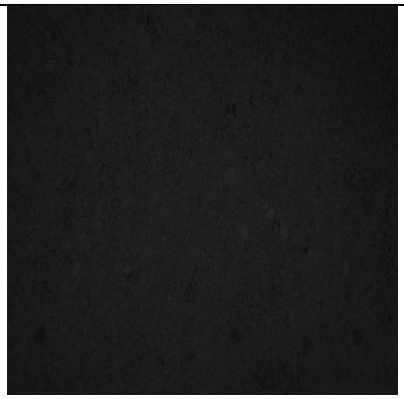
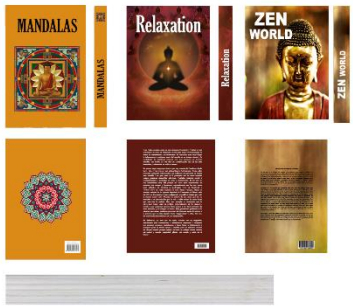
	Tanda parkiran
	Rak koran
	Tong sampah besar
	Tong sampah kecil
	Lampu jalan
	Tiang trotoar

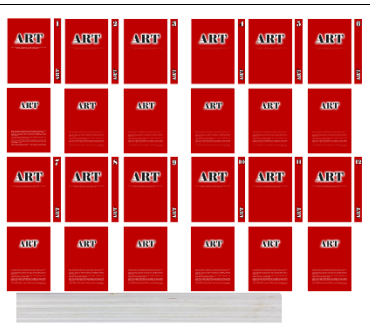
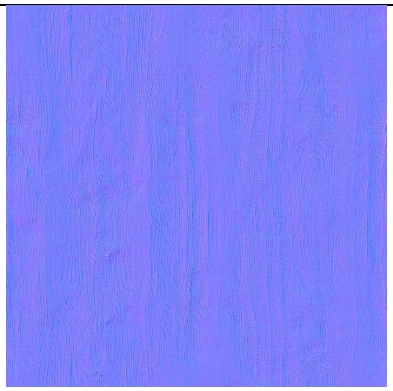
	Tong sampah terbuka
	Warung telepon
	Mobil sport
	Mobil polisi
	Mobil sedan
	Mobil van

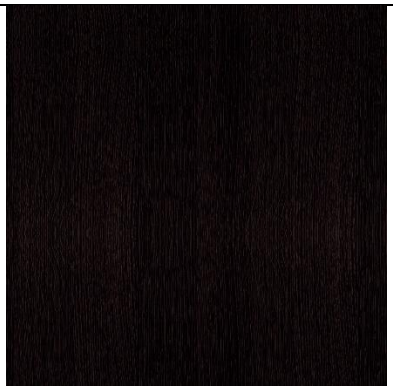
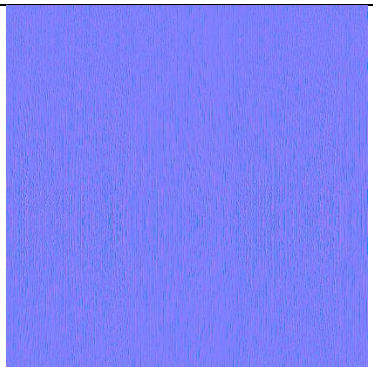
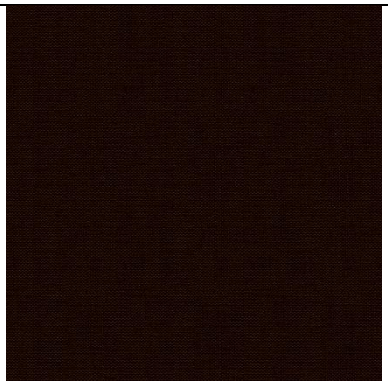
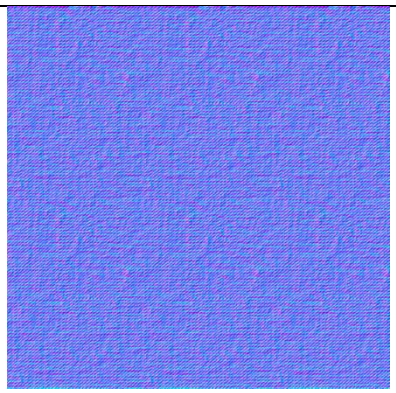
8. *Texture*

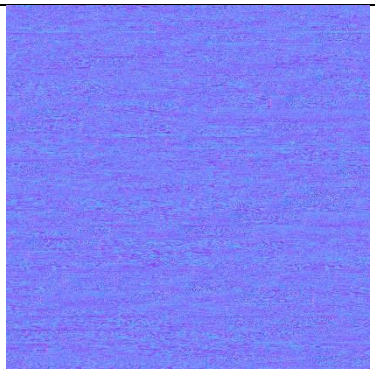
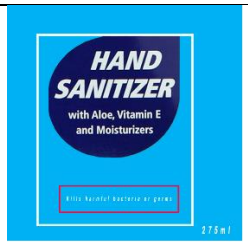
Tabel 3.8 Aset *texture*

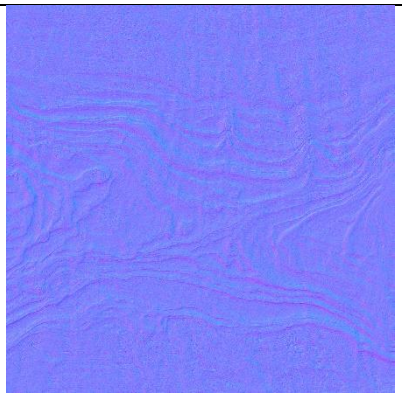
Gambar	Nama *Texture*	Keterangan
	Buku 1	Pewarnaan untuk Buku
	Buku 2	Pewarnaan untuk Buku
	Buku 3	Pewarnaan untuk buku

	Buku Antik	Pewarnaan untuk buku antik
	Tas Laptop	Pewarnaan untuk tas leptop
	Tumpukan Buku 1	Pewarnaan untuk Sebahagian tumpukan buku di rak
	Tumpukan buku 2	Pewarnaan untuk Sebahagian tumpukan buku di rak

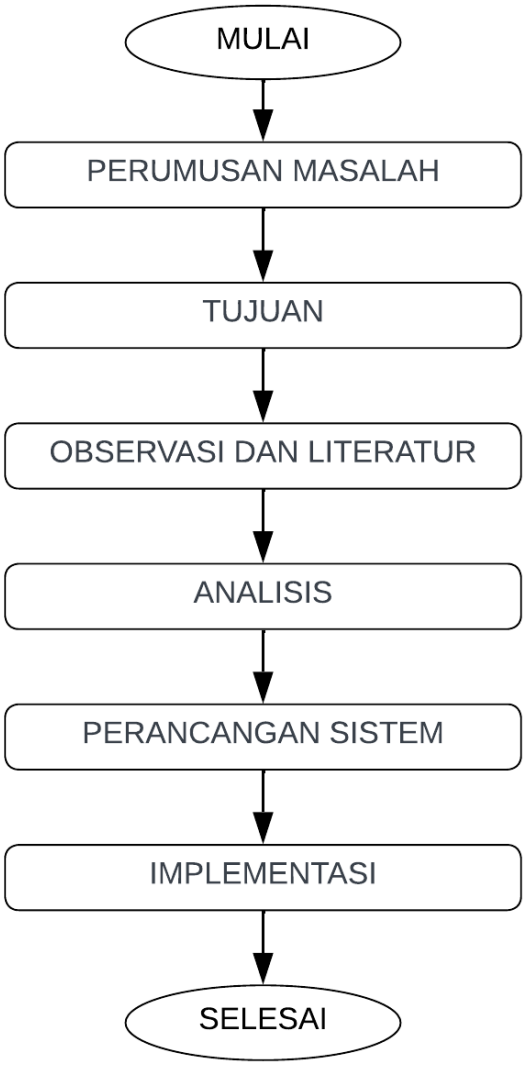
	Tumpukan buku 3	Pewarnaan untuk Sebahagian tumpukan buku di rak
	Tumpukan buku 4	Pewarnaan untuk Sebahagian tumpukan buku di rak
	Rak Buku	Pewarnaan pada rak buku
	Rak Buku	Sebagai pendetilan pada rak buku

	Kursi 1	Sebagai pewarnaan pada kursi
	Kursi 1	Sebagai pendetilan pada kursi
	Kursi sofa	Pewarnaan pada kursi sofa
	Kursi sofa	Sebagai pendetilan pada kursi sofa

	Lantai perpustakaan	Sebagai pewarnan pada lantai perpustakaan
	Lantai perpustakaan	Sebagai pendetilan pada lantai perpustakaan
 	*Hand sanitizer*	Pewarnaan pada *hand sanitizer*
	Laptop	Pewarnaan pada laptop

	Tong sampah	Pewarnaan pada tong sampah
	Meja Panjang	Pewarnaan pada meja Panjang
	Meja Panjang	Sebagai pendetilan pada meja panjang

3.5 Prosedur Alur Penelitian



Gambar 3.11 Prosedur Alur Penelitian

Adapun pejelasan dari langkah-langkah yang digambarkan pada alur penelitian adalah sebagai berikut :

1. Mulai
2. Rumusan Masalah

   Penentuan permasalahan secara jelas dan konsisten berfungsi untuk mengubah target menjadi situasi yang mudah dibentuk.
3. Tujuan

Tujuan penelitian ini adalah sebuah sasaran yang nantinya akan diwujudkan atau hasil dari penyelesaian suatu masalah yang akan diteliti.

4. Observasi dan Literatur

   Menampung data penelitian dapat dilakukan dengan observasi dan literatur. Observasi penelitian ini dilakukan guna melihat secara langsung dan mempelajari permasalahan yang ada. Literatur mencari bahan yang berkaitan dengan masalah yang sudah ditemukan, yaitu dengan cara mencari bahan dari buku-buku, internet, jurnal, skripsi dan objek yang berkaitan dengan permasalahan.

5. Analisis

   Analisis kebutuhan untuk mengetahui kebutuhan dalam membuat program dari fungsional dan nonfungsional yang diinginkan dari program.

6. Perancangan Sistem

   Perancangan terdiri dari desain arsitektur, desain data, desain antarmuka, dan desain prosedur aplikasi.

7. Implementasi

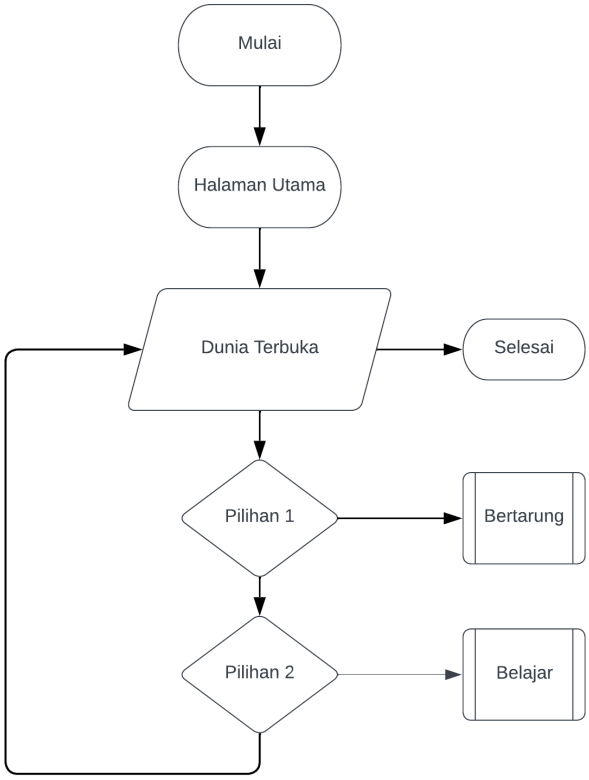
   Proses perealisasian dari tahap perancangan menjadi bentuk software atau sistem yang dapat dijalankan.

8. Selesai

3.6 Skema Sistem

Sistem yang kompleks dan terstruktur dengan baik perlu direpresentasikan dengan skema sistem yang jelas, yang mencakup struktur dan cara kerja sistem secara terperinci. Salah satu cara untuk menggambarkan proses tersebut secara visual adalah dengan menggunakan flowchart atau diagram alir.

Skema sistem meliputi struktur dan cara kerja sistem yang dipresentesikan dengan *flowchart* (diagram alir). Skema sistem meliputi struktur dan cara kerja sistem yang dipresentesikan dengan *flowchart* (diagram alir). Flowchart merupakan diagram alir yang menjelaskan proses-proses yang berlangsung di program secara keseluruhan.



Gambar 3.12 Skema Sistem

Keterangan:

1. Mulai
2. User membuka game dan melihat halaman utama
3. Kemudian melihat dan memasuki dunia terbuka.
4. Pilihan Pertama yaitu menu bertarung, dalam menu bertarung ini user dapat bertarung dengan lawan secara *real-time* menggunakan kuis pilihan ganda untuk mengasah ingatan dan mengulang materi.
5. Pilihan Kedua yaitu menu belajar, yang terdiri dari beberapa materi Bahasa Pemograman.
6. Selesai.

3.7 Teknik Pengumpulan Data

Metode pengumpulan data yang digunakan penulis dalam penelitian ini adalah *survey* kepustakaan, dan peneliti mengumpulkan data dari berbagai referensi seperti jurnal dan sumber tertulis lain yang relevan dengan penelitian ini.

3.8 Analisa Kebutuhan Sistem

Ada beberapa hal yang perlu dipersiapkan sebelum membangun aplikasi. Pada penelitian ini, penulis menggunakan perangkat pribadi yang dimiliki sebagai acuan spesifikasi untuk membangun sistem yang optimal dengan rincian sebagai berikut :

3.8.1 Analisa Kebutuhan Perangkat Keras (*Hardware*)

Adapun alat dan bahan yang peneliti gunakan dalam penelitian untuk pengembangan *game* Pengetahuan ini adalah :

1. Alat

   Laptop HP dengan spesifikasi sebagai berikut :

   1. *Processor* : *Amd Ryzen* 5 3550H
   2. *Memory* : 16 GB (*RAM*)
   3. *Storage* : 512 GB (*SSD*)
   4. *Graphics* : *NVIDIA GEFORCE GTX 1050*
   5. *OS* : *Windows 10 Pro Education*

   HP Xiaomi Redmi 9 Pro dengan spesifikasi sebagai berikut :

   1. Versi *Android* : *Android* 12
   2. Versi *UI* : MIUI *Global* 13.0.5
   3. *Processor* : *Qualcomm* SM715 *Snapdragon* 720G (8 nm)
   4. *CPU* : *Octa-core Max* 2.32Ghz
   5. *RAM* : 8 GB

2. Bahan

   Adapun bahan yang peneliti gunakan dalam penelitian ini berupa *software* aplikasi pendukung dalam rancang bangun aplikasi yaitu :

   1. *Unity 3D*
   2. *Blender*
   3. *Adobe Audition*

3.9 Tempat Dan Jadwal Penelitian

Penelitian tentang *game* pengetahuan bahasa pemograman ini yang menjadi subjek penelitiannya adalah programmer pemula dan lanjut. Dengan jadwal penelitian pada tabel 3.1 sebagai berikut:

Tabel 3.9 Tabel jadwal penelitian

Kegiatan	Tahun 2023																															
	Jan				Feb				Mar				Apr				Mei				Jun				Jul				Ags				Sep			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Penyusunan proposal																																				
Seminar																																				
Pengumpulan Data																																				
Perancangan Sistem																																				
Pembuatan Program																																				
Pengujian Program																																				
Perbaikan Program																																				
Penyusunan Laporan																																				
Sidang																																				
Revisi TA																																				

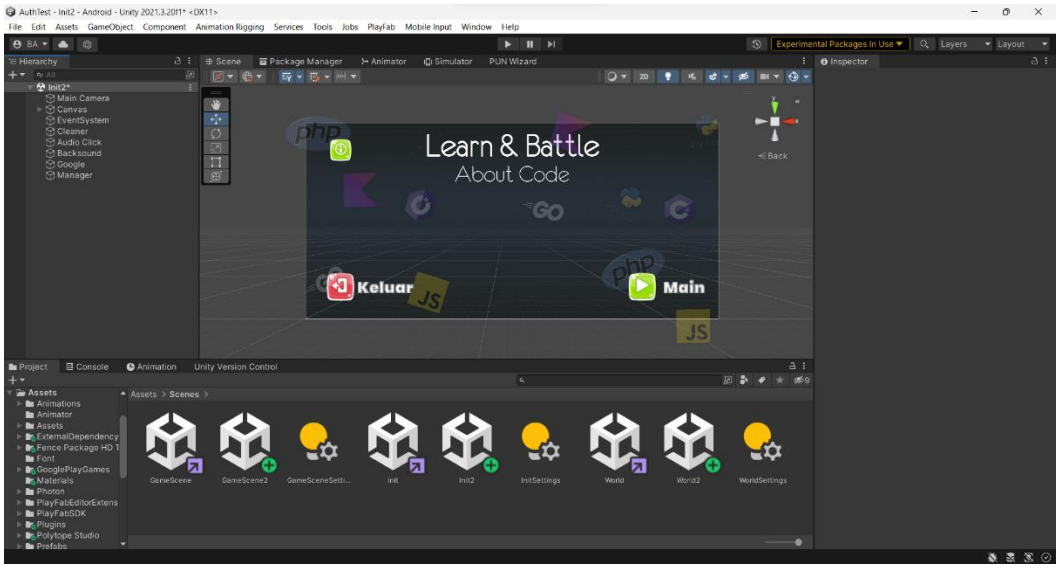
BAB IV
HASIL DAN PEMBAHASAN

4.1 Produksi

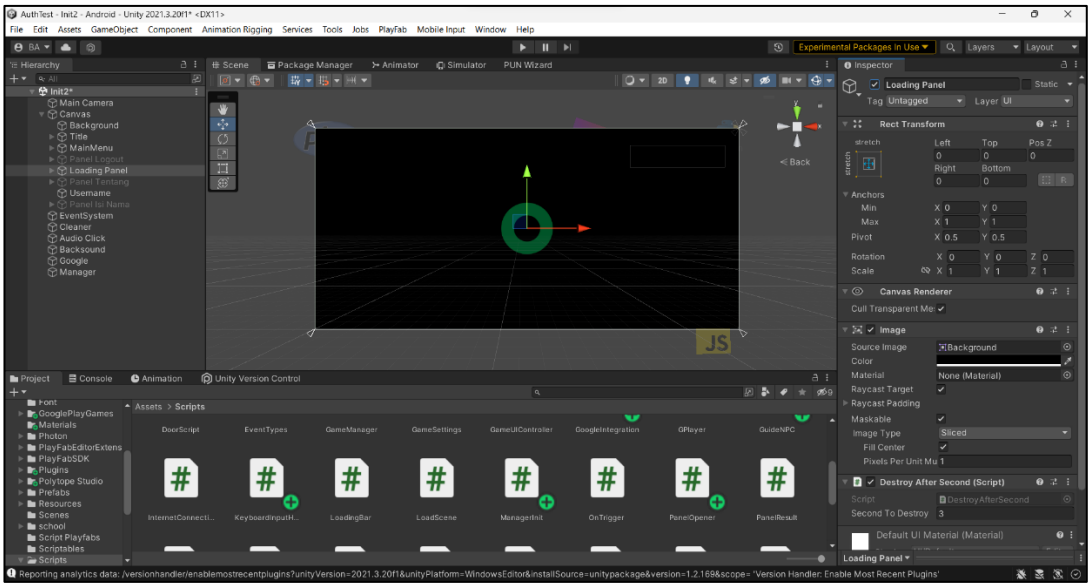
Dalam tahap produksi permainan, pengembangan permainan dimulai dengan menggunakan mesin permainan *Unity*. Seluruh aset permainan diorganisir dengan rapi dalam struktur folder di dalam lingkungan *Unity*, dan kemudian diimpor ke dalam proyek dengan metode seret-dan-lepas (*drag-and-drop*). Proses pembuatan permainan menggunakan mesin *Unity* melibatkan beberapa tahap, termasuk *Autentikasi*, Pembuatan Peta, Desain Karakter, Pembuatan Karakter Non-Pemain (*NPC*), Pembelajaran Pemrograman, dan Pengembangan Kompetisi kuis. Hasil akhir dari proses ini adalah permainan yang dapat dijalankan pada sistem operasi *Android*.

4.1.1 *Main Menu*

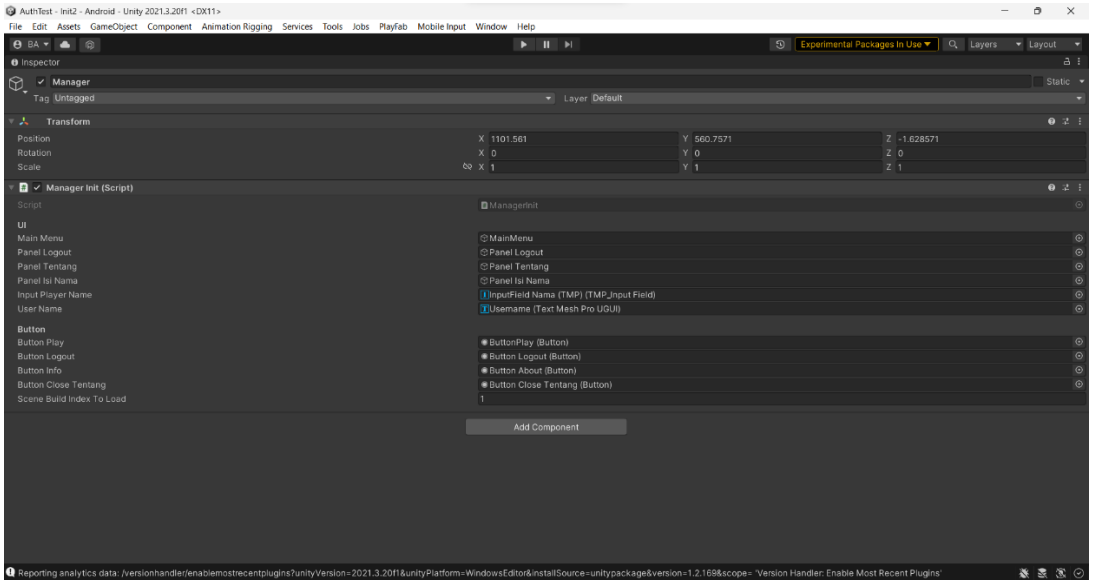
Main menu merupakan hal yang penting bagi setiap *game*, untuk proses pembuatan *main menu* di *game "Learn and Battle About Code"* di mulai dari penempatan UI dari asset- asset yang telah ada, terdapat pembuatan UI *main menu* 5 gambar yaitu dari Gambar 4.1 sampai Gambar 4.5.



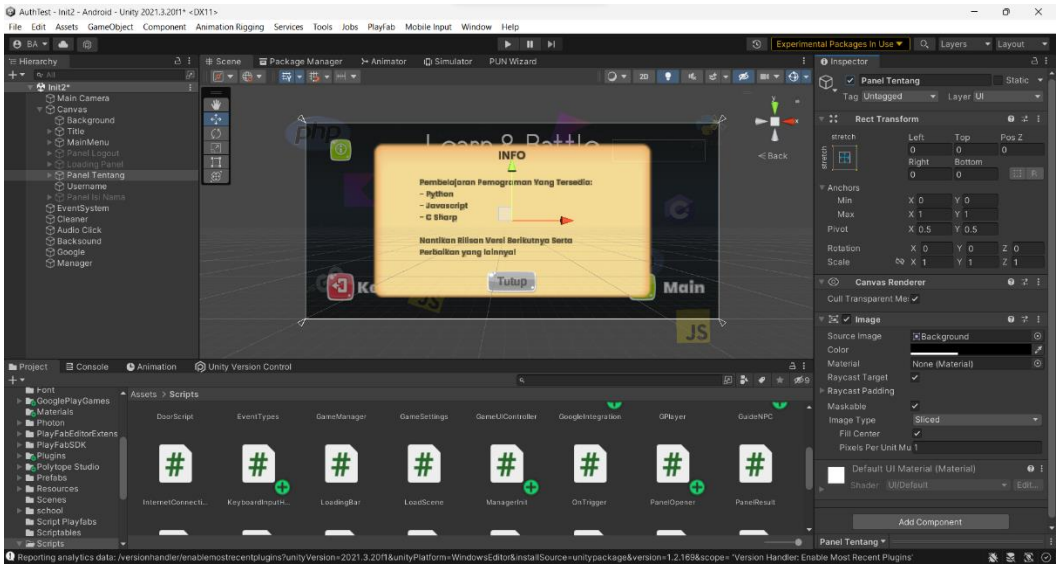
Gambar 4.1 Pembuatan UI *Main Menu*



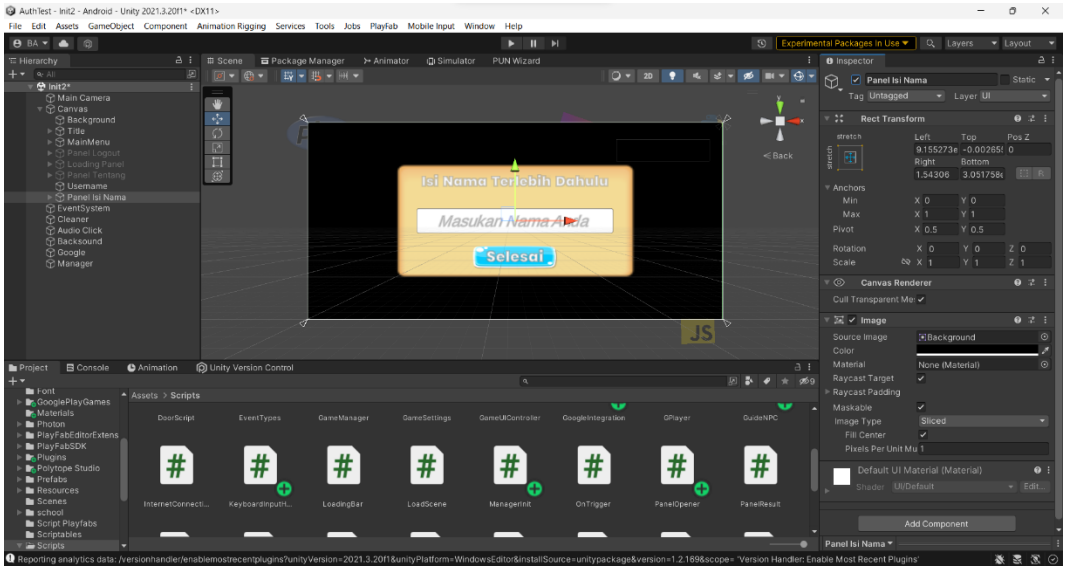
Gambar 4.2 Pembuatan *Loading Screen*



Gambar 4.3 Hasil *Output Script Manager Init*



Gambar 4.4 Pembuatan Tampilan Info



Gambar 4.5 Pembuatan Tampilan Isi Nama

Pertama, saat membangun permainan di *Unity*, kita perlu mendesain dan mengimplementasikan antarmuka pengguna (UI) yang memungkinkan pemain berinteraksi dengan permainan. Ini melibatkan menciptakan elemen-elemen seperti *panel*, *teks*, tombol, dan *input fields* yang akan menjadi bagian dari antarmuka.

Setelah elemen-elemen UI tersebut dibuat, kita perlu mendefinisikan variabel di dalam kode untuk menyimpan referensi ke elemen-elemen tersebut.

Variabel ini akan diinisialisasi menggunakan atribut `*SerializeField*` di *Unity*, yang memungkinkan kita untuk mengaitkan objek UI dengan variabel di dalam skrip.

Kemudian, kita menanggapi interaksi pengguna dengan elemen-elemen UI melalui metode-metode yang terkait. Misalnya, ketika tombol "*Play*" ditekan, kita memanggil metode `*OnPressedPlay()*` yang kemudian akan memutuskan tindakan selanjutnya berdasarkan input pengguna.

Untuk memastikan konsistensi dan tampilan yang sesuai, penting untuk memastikan tata letak yang baik. Panel-panel dan elemen-elemen UI lainnya perlu diatur secara estetis dan intuitif agar pengalaman pengguna menjadi lebih baik.

Dalam hal ini, kita juga menggunakan *coroutine* untuk melakukan tindakan tertentu secara bertahap, seperti memuat adegan. *Coroutine* memungkinkan tindakan tersebut dilakukan tanpa memblokir eksekusi utama dan menghindari membeku atau 'hang' dalam aplikasi.

Penting untuk selalu memeriksa dan mengelola keadaan UI, termasuk menampilkan dan menyembunyikan panel sesuai dengan aksi yang diambil oleh pengguna. Hal ini memastikan bahwa antarmuka responsif dan memberikan pengalaman pengguna yang memadai.

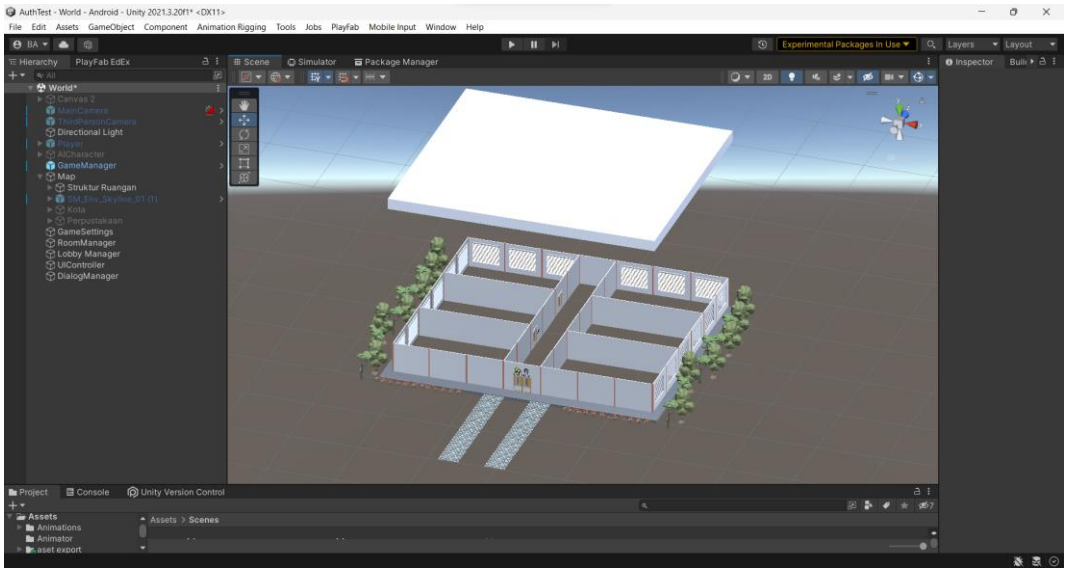
Terakhir, perlu diingat bahwa keluar dari permainan juga merupakan aspek penting yang perlu dipertimbangkan. Kita perlu memastikan bahwa pengguna dapat keluar dari permainan dengan benar dan sesuai dengan kebutuhan *platform* yang digunakan, baik itu saat di *editor Unity* maupun saat permainan sudah dibangun.

Dengan demikian, proses pembuatan UI melibatkan desain, penghubungan UI dengan kode, menanggapi interaksi pengguna, mengelola tampilan, mempertimbangkan konsistensi tata letak, menggunakan *coroutine*, dan memastikan keluaran yang memadai bagi pengguna.

4.1.2 Pemodelan dan Pemerenderan Lingkungan *3D*

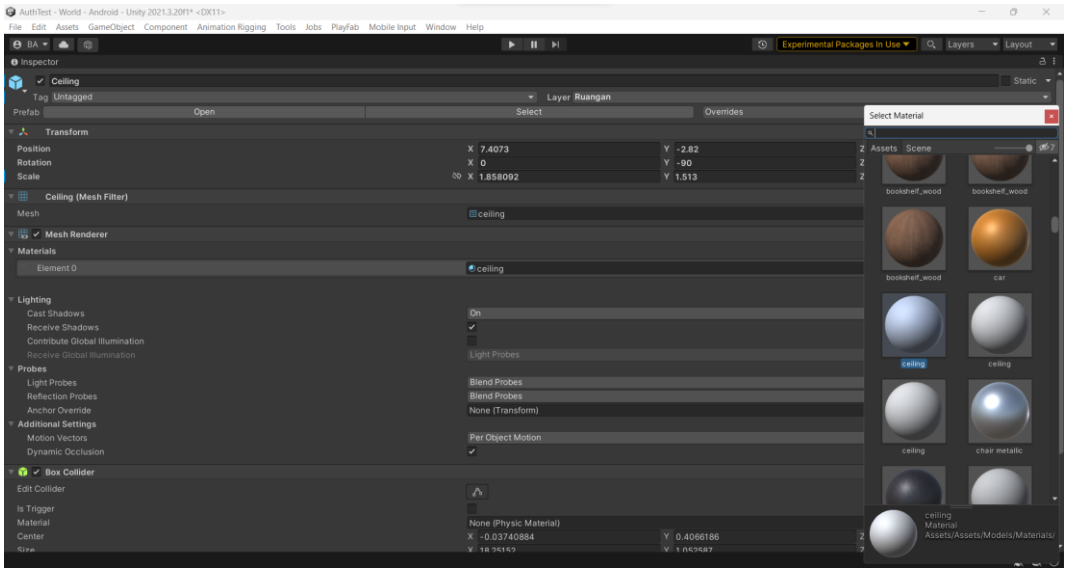
Setelah selesai membangun semua aset, mulai dari jendela, lantai, gedung, dan asset lainnya, pada tahap ini dilakukan proses penyatuan dari seluruh aset

tersebut. Hasil dari penyatuan ini akan membentuk sebuah perpustakaan yang lengkap dengan halaman serta kota secara keseluruhan.



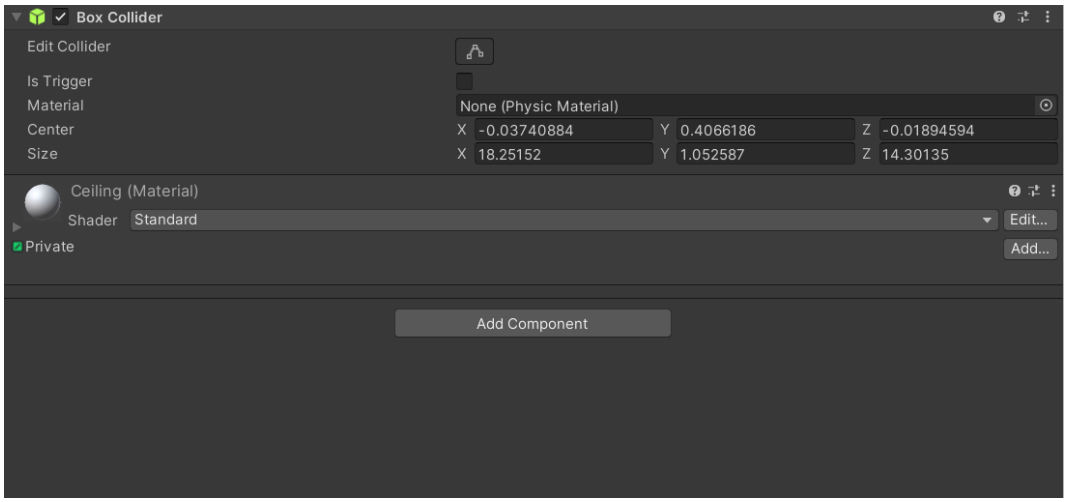
Gambar 4.6 Proses Pembuatan Bangunan Perpustakaan

Proses dimulai dengan pengidentifikasian semua asset yang diperlukan, termasuk 12 jendela yang ditempatkan pada sisi kanan dan kiri bangunan, serta 7 pintu yang terdiri dari 1 pintu depan dan 6 pintu yang memfasilitasi akses ke dalam ruangan-ruangan. Selanjutnya, perencanaan tata letak bangunan dengan mengatur 6 ruangan sesuai dengan desain perpustakaan seperti gambar di atas. Penempatan pintu dan jendela dilakukan sesuai dengan tata letak yang telah direncanakan sebelumnya. Ruangan-ruangan disusun dengan mempertimbangkan fungsionalitas dan efisiensi penggunaan ruang. Seluruh asset seperti lantai, dinding, dan elemen bangunan lainnya diintegrasikan sesuai dengan desain yang telah ditetapkan. Selanjutnya, furnitur dan elemen *interior* ditempatkan di setiap ruangan sesuai dengan desain *interior* yang telah dipersiapkan sebelumnya. Pengujian fungsionalitas dilakukan untuk memastikan bahwa pintu dan jendela dapat berfungsi dengan baik, serta memastikan alur akses yang lancar di seluruh bangunan. Akhirnya, setelah *verifikasi* desain terakhir, perpustakaan dengan 6 ruangan, 7 pintu, dan 12 jendela telah selesai dibangun dan siap digunakan. Bangunan ini telah dirancang dengan 1 lantai untuk memenuhi kebutuhan perpustakaan yang diinginkan.



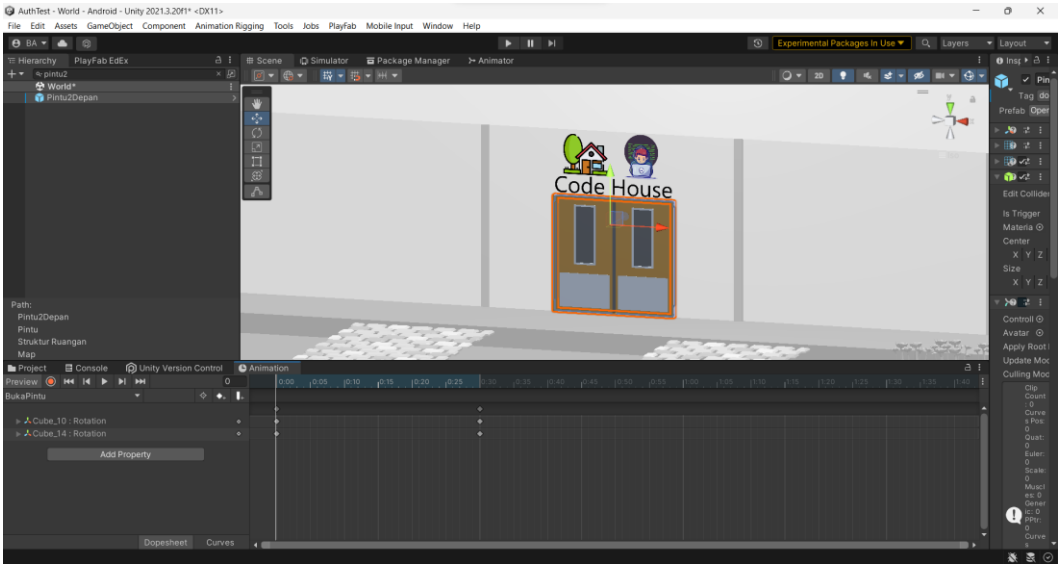
Gambar 4.7 Proses Penerapan *Material*

Proses pemasukan *material* ke dalam asset di *Unity* diawali dengan pemilihan objek yang akan diberi material, baik itu *model 3D* seperti karakter atau bangunan. *Material* baru dapat diciptakan atau yang telah ada dapat digunakan. Apabila *material* baru dibuat, berbagai properti *visual* seperti warna, tekstur, dan tingkat kecerahan diatur sesuai kebutuhan. Setelah material siap, material tersebut ditarik dan dilepaskan ke objek yang telah terpilih di Unity. Langkah terakhir, penyesuaian dan pengaturan properti *material* dilakukan melalui *Inspector*, guna mendapatkan tampilan *visual* yang diinginkan.

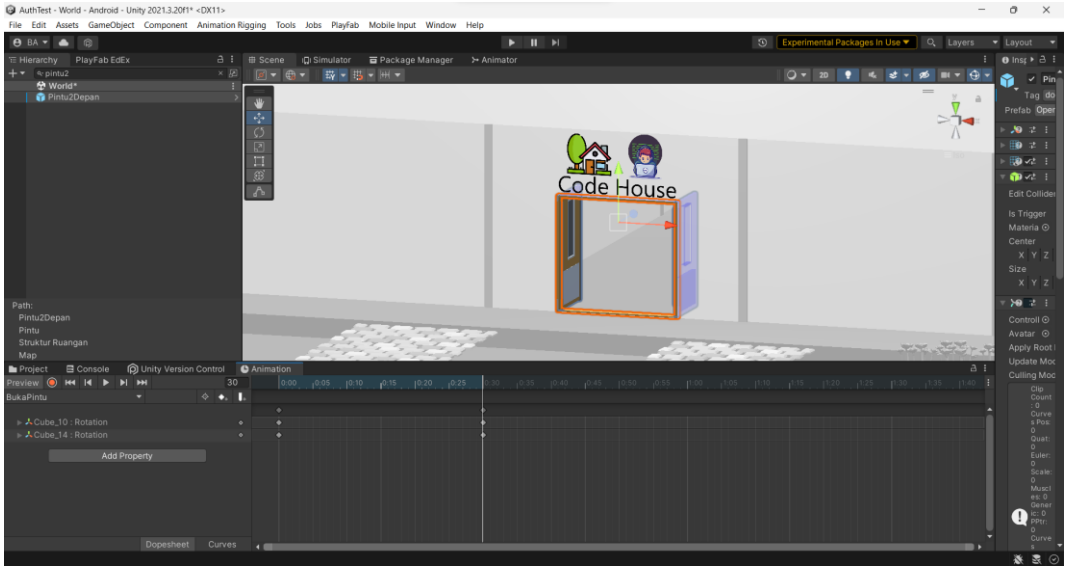


Gambar 4.8 Penerapan *Collider*

Collider di *Unity* dianggap kunci dalam mengelola interaksi fisik di lingkungan simulasi *3D* dan *2D*. Proses penerapannya dimulai dengan memilih objek yang akan diberi *Collider*, baik itu karakter, dinding, lantai, atau elemen lain dalam permainan. Setelah memilih objek, *Collider* dapat ditambahkan dengan memilih tipe yang sesuai, seperti "*Box Collider*" atau "*Sphere Collider*" untuk objek *3D*, atau "*Box Collider 2D*" atau "*Circle Collider 2D*" untuk objek *2D*. Properti seperti ukuran dan bentuk *Collider* dapat disesuaikan melalui *Inspector* sesuai karakteristik objek. Pengujian interaksi fisik yang dihasilkan dapat dilakukan melalui pengujian permainan. Dengan adanya *Collider*, pencipta permainan dapat membangun interaksi yang akurat dan menarik antarobjek.

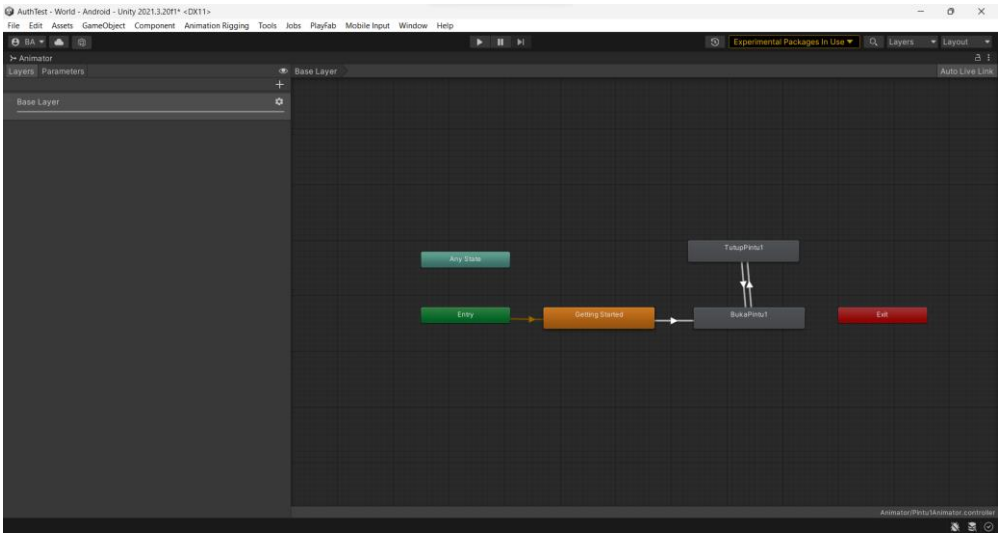


Gambar 4.9 Proses Pembuatan *Frame 1* Animasi Pintu



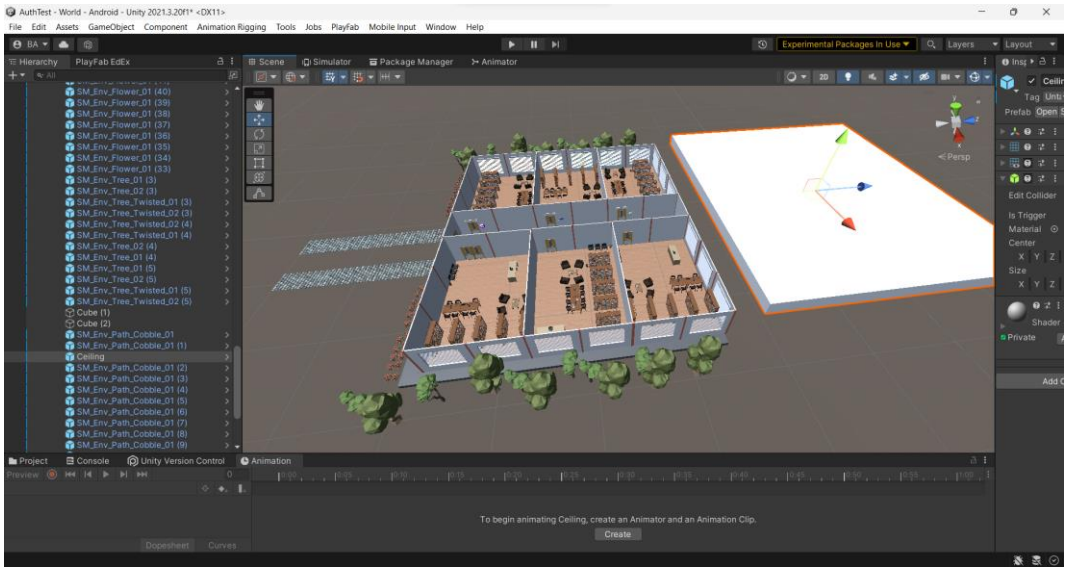
Gambar 4.10 *Frame 2* Animasi Pintu

Proses animasi pintu dua *frame* rotasi dimulai dengan pemilihan objek pintu, memperhatikan kemampuan rotasi sumbu y dan keberadaan *Collider* untuk interaksi yang sesuai. Animasi baru dibuat dengan pengaturan nilai rotasi awal (*x*, *y*, *z*) pada *frame* pertama menjadi 0. *Keyframe* pertama merekam nilai rotasi ini. Pada *frame* kedua, rotasi sumbu y diatur menjadi 90 derajat sambil mempertahankan rotasi x dan z pada 0. *Keyframe* kedua merepresentasikan nilai rotasi yang telah diatur. Hasilnya, pintu berputar 90 derajat sekitar sumbu y saat beralih dari *frame* pertama ke *frame* kedua. Animasi ini meningkatkan interaksi dan pengalaman pengguna dengan objek pintu.



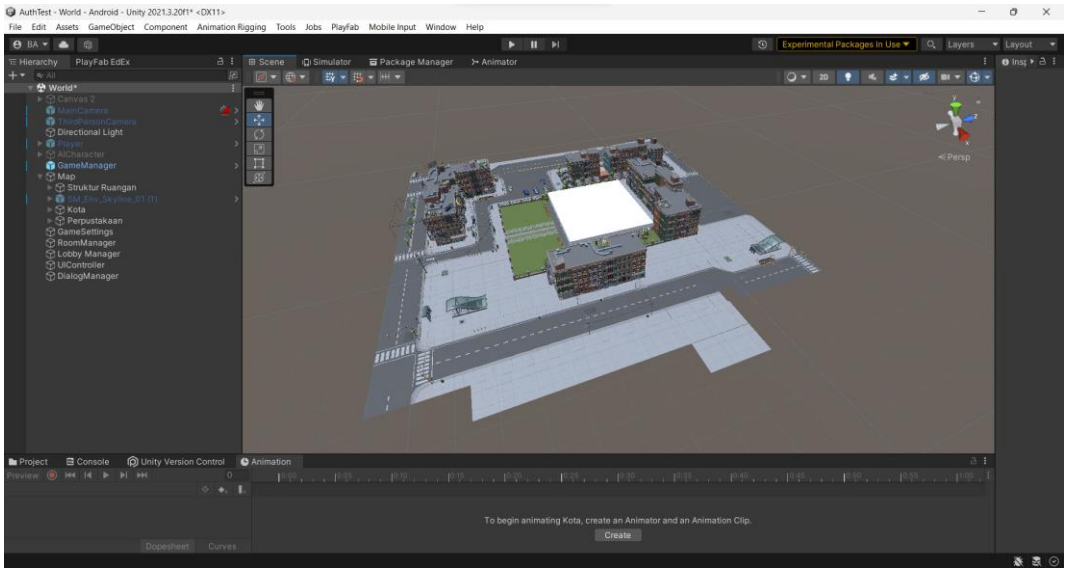
Gambar 4.11 *Animator* Pintu

Dalam proses penciptaan *animator* pintu dengan kode yang diberikan, dimulai dengan inisialisasi variabel seperti `*doorState*` yang digunakan untuk merekam status pintu, baik dalam keadaan terbuka maupun tertutup, serta `*inRange*` untuk memonitor apakah pemain berada dalam jarak yang memadai untuk berinteraksi dengan pintu. Selanjutnya, komponen Animator dari objek tersebut diinisiasi melalui metode `*Start()*`. Setiap kali *frame* diperbarui, melalui metode `*Update()*`, input dari pemain dipantau, terutama aksi untuk membuka pintu, yang dapat dipicu dengan menekan tombol "*OpenDoor*". Jika pemain berada dalam jarak yang sesuai (sesuai dengan tag "*Player*"), metode `*ToggleDoorState()*` dipanggil. Dalam metode `*ToggleDoorState()*`, status pintu diubah, dan parameter *animator* "*DoorState*" diperbarui sesuai dengan status terbaru. Pengaturan status pintu ini juga didokumentasikan melalui pesan *Debug Log* yang mencatat perubahan status pintu. Selanjutnya, terdapat pengecekan saat terjadi kontak, yang diimplementasikan melalui metode `*OnTriggerEnter()*` dan `*OnTriggerExit()*`, untuk memastikan bahwa objek yang berinteraksi adalah pemain (berdasarkan tag "*Player*"), sehingga memungkinkan konfirmasi apakah pemain berada dalam jarak yang tepat untuk berinteraksi dengan pintu. Dengan pendekatan ini, *Animator* dapat diaplikasikan dengan efektif untuk mengendalikan animasi pintu yang responsif dan sesuai dengan tindakan serta posisi pemain dalam lingkungan permainan.



Gambar 4.12 Proses Peletakan Dekorasi, Perabotan, Furnitur

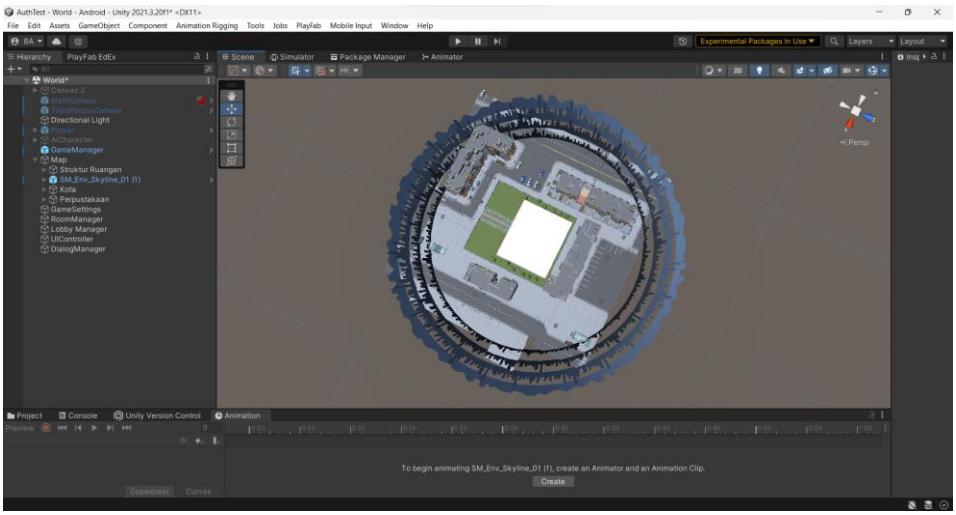
Proses peletakan dekorasi, perabotan, dan furnitur di *Unity* dimulai dengan identifikasi ruang atau area di mana elemen-elemen ini akan ditempatkan. Setelah ruang ditentukan, pemilihan dekorasi dan perabotan yang sesuai dengan konsep desain dilakukan. Selanjutnya, elemen-elemen ini disesuaikan dengan tata letak ruangan, memperhatikan aspek estetika, fungsionalitas, dan kenyamanan. Penggunaan alat-alat Unity seperti *Transform* dan *Collider* digunakan untuk menempatkan dengan tepat dan mencegah tumpang tindih antar elemen. Interaksi dan navigasi dalam ruangan juga harus dipertimbangkan agar elemen-elemen dapat diakses dengan baik oleh pengguna. Terakhir, dilakukan pengujian dan iterasi untuk memastikan tampilan dan tata letak yang dihasilkan memenuhi tujuan desain dan memberikan pengalaman pengguna yang optimal.



Gambar 4.13 Proses Penataan Bangunan dan Jalanan

Proses penataan bangunan dan jalanan sebagai hiasan sekeliling dimulai dengan konseptualisasi desain yang mengintegrasikan elemen-elemen bangunan dan jalanan secara estetis. Identifikasi elemen-elemen bangunan seperti tembok, jendela, pintu, serta jalanan, trotoar, dan lampu penerangan. Setelah identifikasi, penempatan elemen-elemen ini disesuaikan dengan tata letak yang memperhatikan komposisi visual dan estetika yang diinginkan. Perhatian khusus diberikan pada penggunaan *material*, tekstur, dan pencahayaan untuk meningkatkan nuansa estetik yang diinginkan. Penggunaan teknologi digital dan perangkat lunak seperti *Unity* memudahkan penataan dan penyusunan elemen-elemen ini secara visual dan

interaktif. Proses ini memastikan bangunan dan jalanan menjadi elemen penambah daya tarik visual yang melengkapi atmosfer dan nuansa keseluruhan suatu lingkungan.

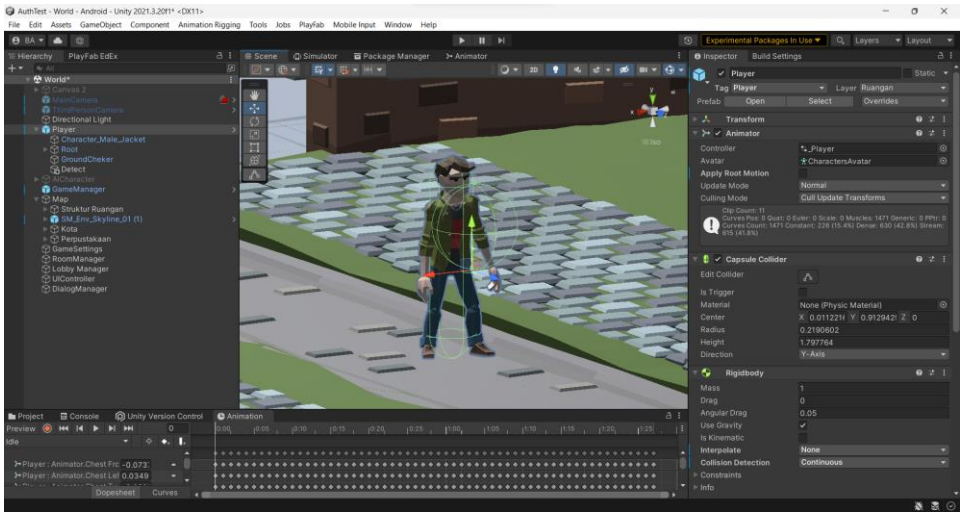


Gambar 4.14 Proses Peletakan Siluet Perkotaan

Proses peletakan siluet perkotaan diawali dengan analisis karakteristik yang ingin dicerminkan oleh perkotaan, dengan mempertimbangkan sudut pandang yang mencakup seluruh kota dan perspektif yang sesuai untuk menempatkan siluet-siluet ini dalam suatu komposisi visual yang menarik.

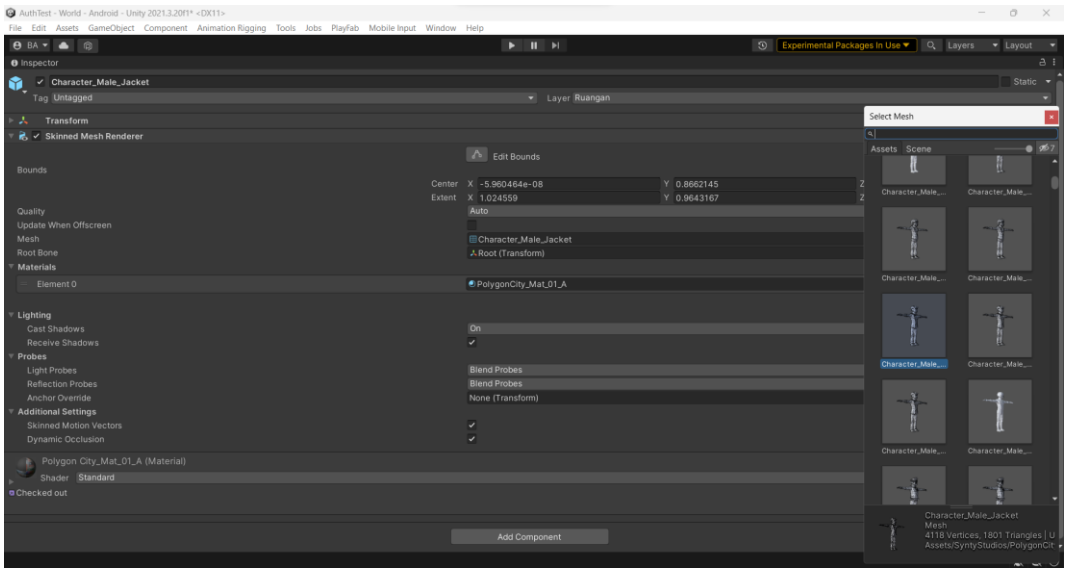
4.1.3 Karakter Utama

1. *Model 3D* Karakter, *Mesh Renderer*, *Material*



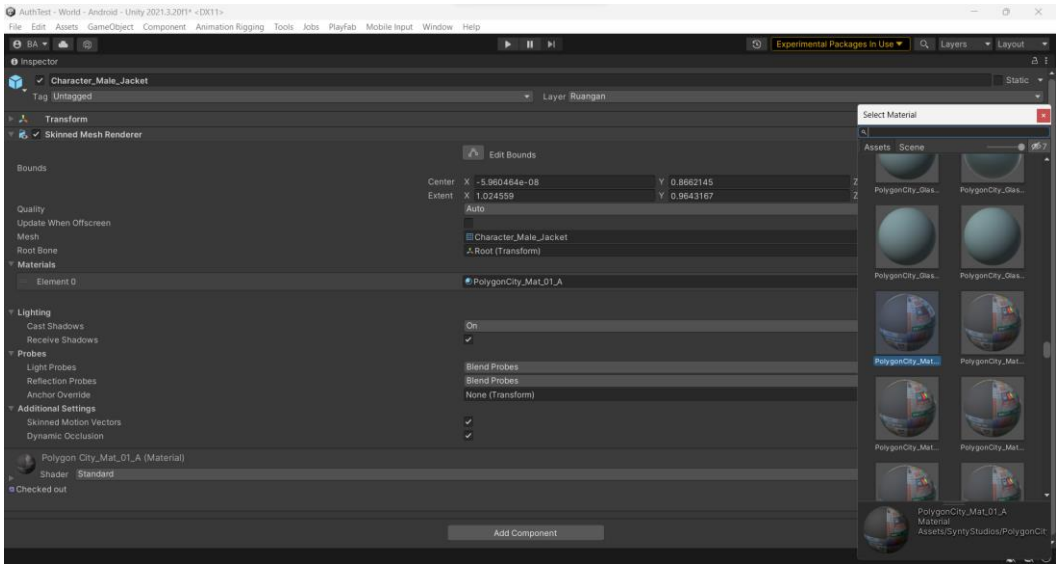
Gambar 4.15 Peletakan Karakter Utama

Peletakan karakter utama diilustrasikan dengan presisi, memastikan karakter tersebut ditempatkan dalam posisi yang optimal di dalam lingkungan permainan, mempertimbangkan tata letak yang memungkinkan interaksi yang tepat dan alur permainan yang diinginkan.



Gambar 4.16 Proses Konfigurasi *Mesh Renderer* Karakter

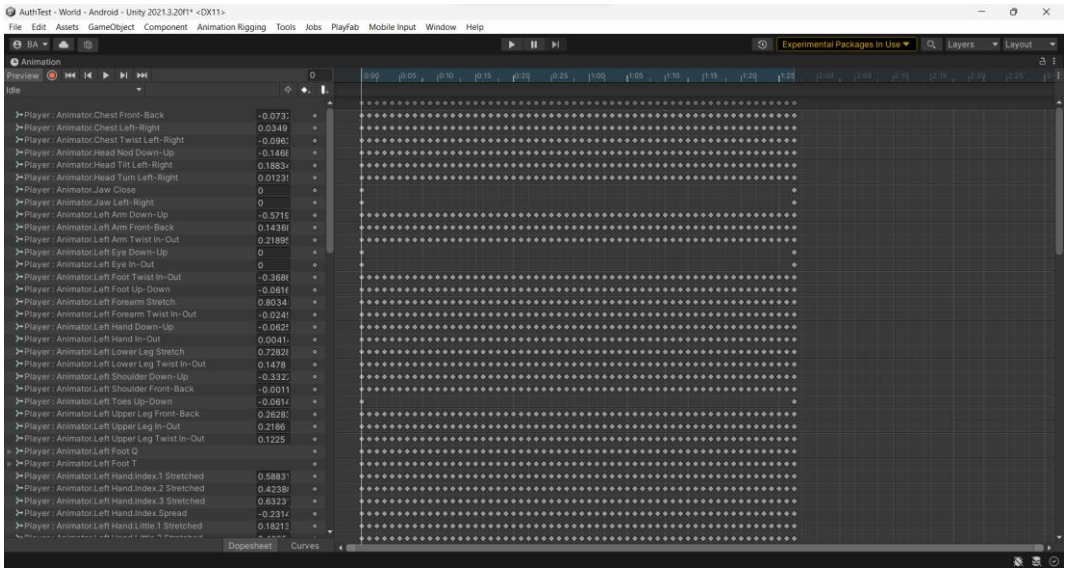
Proses konfigurasi *Mesh Renderer* karakter dijelaskan dengan seksama, di mana parameter dan properti dari *Mesh Renderer* disesuaikan secara hati-hati untuk memastikan tampilan visual karakter memenuhi standar yang diinginkan, menciptakan imersi yang memadai bagi pengalaman bermain.



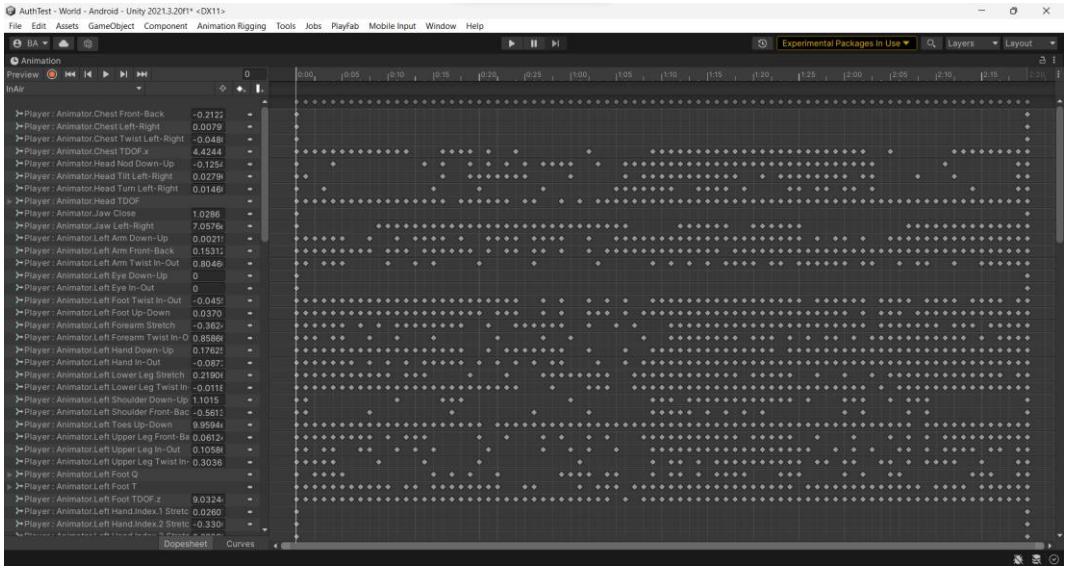
Gambar 4.17 Proses Konfigurasi *Material* Karakter

Proses konfigurasi *Material* karakter seperti Gambar 4.17, mencakup pengaturan sifat-sifat material seperti tekstur, warna, dan properti visual lainnya, sehingga menghasilkan tampilan karakter yang estetis dan sesuai dengan desain permainan.

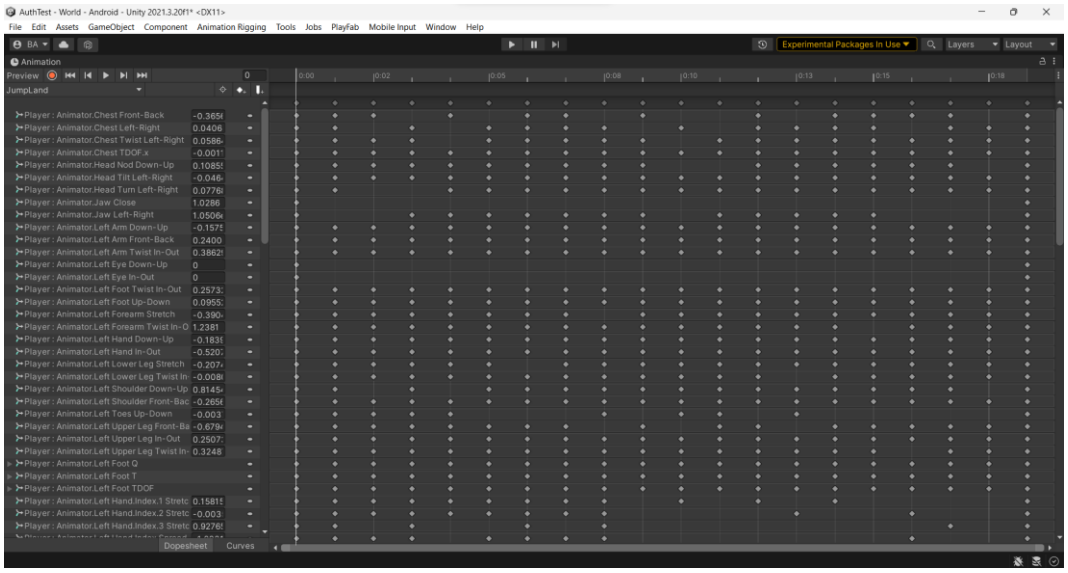
2. Animasi dan *Animator* Karakter



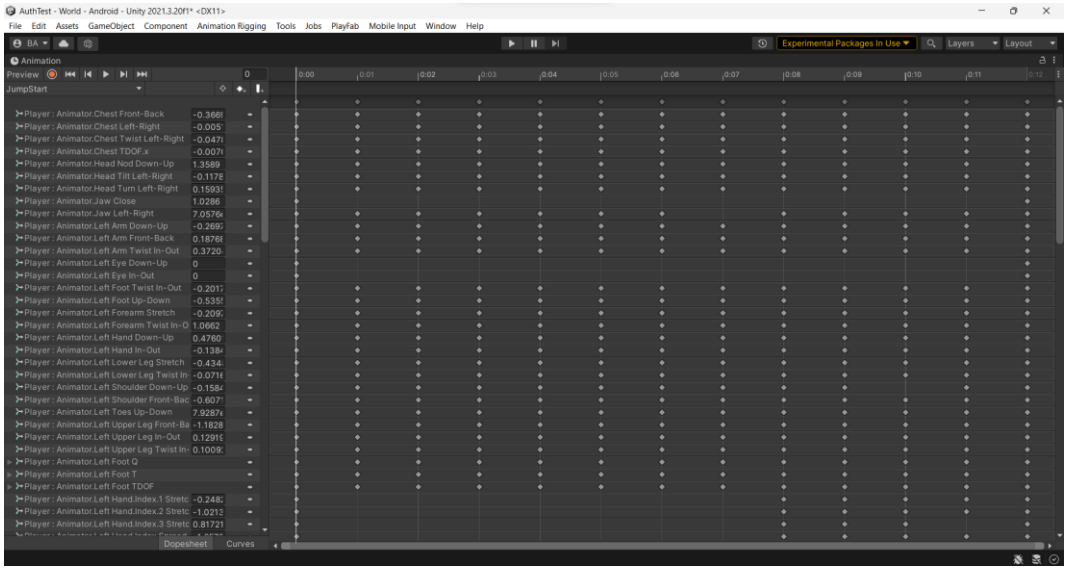
Gambar 4.18 Proses Pembuatan Animasi *Idle*



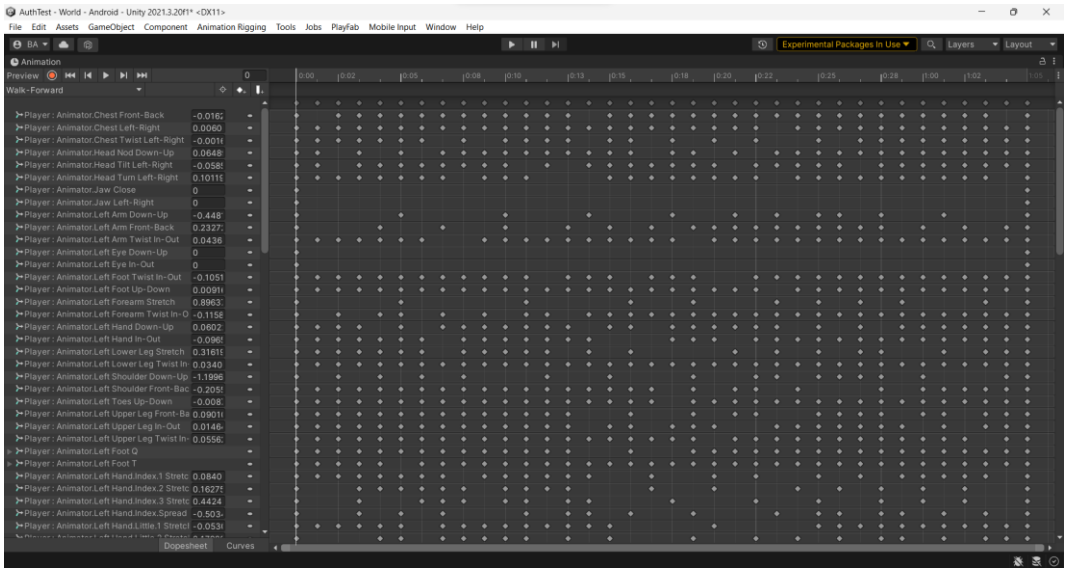
Gambar 4.19 Proses Pembuatan Animasi di Udara



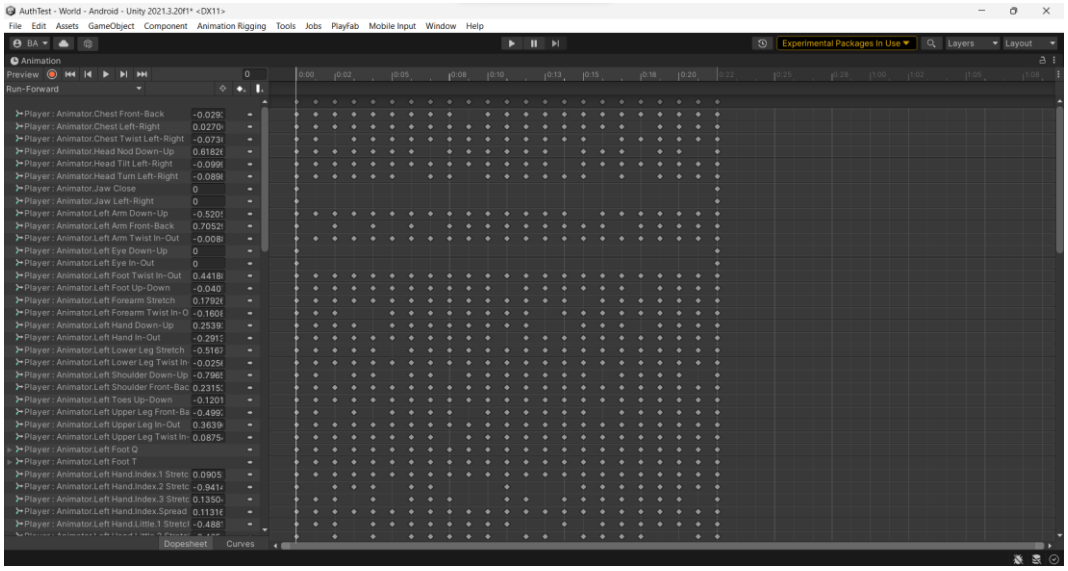
Gambar 4.20 Proses Pembuatan Animasi Mendarat



Gambar 4.21 Proses Pembuatan Animasi Saat Melompat



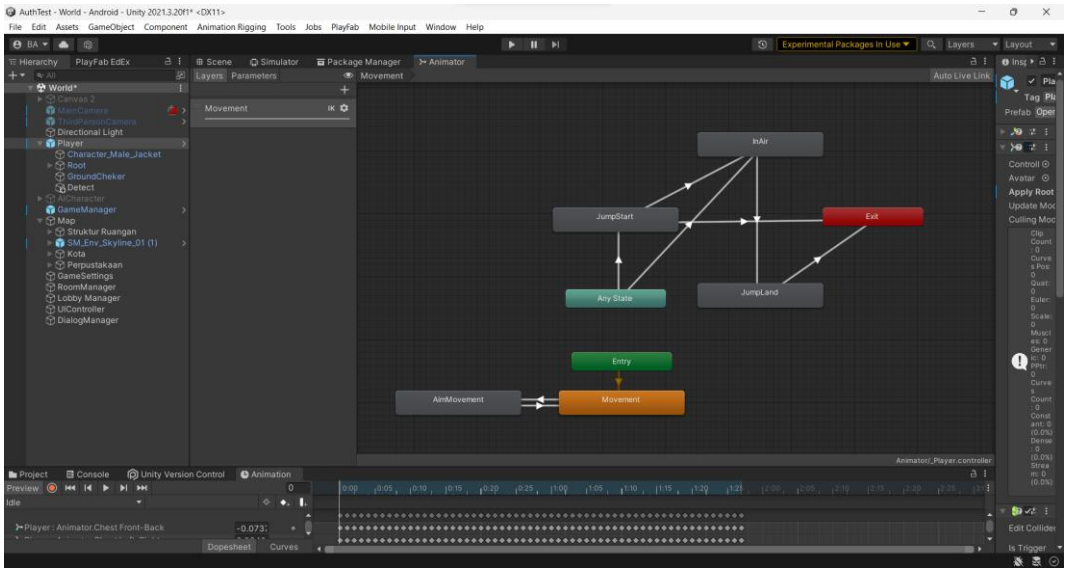
Gambar 4.22 Proses Pembuatan Animasi Jalan



Gambar 4.23 Proses Pembuatan Animasi Lari

Gambar 4.18 mencerminkan proses pembuatan animasi *idle* yang dilakukan dengan teliti, menghasilkan adegan di mana karakter tidak bergerak tetapi memancarkan ekspresi yang sesuai dengan situasi permainan. Gambar 4.19 memberikan gambaran tentang proses pembuatan animasi di udara, menggambarkan langkah-langkah dalam menciptakan animasi yang terasa lepas di lingkungan udara dalam permainan. Sementara itu, Gambar 4.20 menunjukkan proses pembuatan animasi mendarat, di mana langkah-langkah untuk menghasilkan animasi yang realistis saat karakter mendarat setelah lompatan dipaparkan dengan

jelas. Gambar 4.21 mengilustrasikan proses pembuatan animasi saat karakter melakukan lompatan, memperlihatkan perincian langkah demi langkah untuk menciptakan efek visual yang memadai. Di sisi lain, Gambar 4.22 menampilkan proses pembuatan animasi jalan dengan seksama, menggambarkan bagaimana karakter dapat dihidupkan dengan gerakan yang alami saat berjalan. Terakhir, Gambar 4.23 menunjukkan proses pembuatan animasi lari yang memerlukan perencanaan yang cermat, memastikan karakter terlihat dinamis dan realistis ketika bergerak dengan kecepatan.



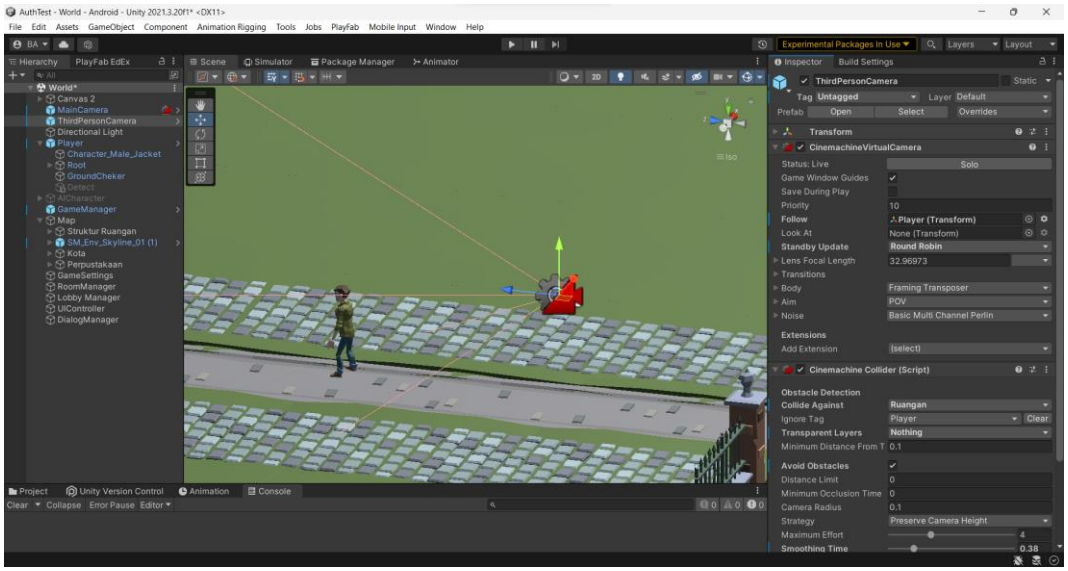
Gambar 4.24 Proses Integrasi Animasi Kedalam *Animator*

Gambar 4.24 memperlihatkan susunan dan alur animasi. Dimulai dari *Any State*, potensi transisi ke *JumpStart* atau *InAir* tergambar. Pada kondisi *JumpStart*, ada kemungkinan transisi ke *InAir* atau keluar dari konteks animasi ini. Begitu juga dengan kondisi *InAir*, potensi transisi ke *JumpLand* sebelum mencapai terminasi. Selanjutnya, perencanaan transisi dari *entry* menuju *Blend Tree* yang memfasilitasi pergerakan dengan kecepatan hingga 0.5 terlihat jelas. *Blend Tree* ini terhubung dengan animasi *Idle*, *Walk Forward*, dan *Run-Forward* yang dapat diakses sesuai dengan pergerakan karakter.

Proses ini juga melibatkan empat parameter, yaitu *groundCheckDistance* dengan tipe data *int*, *grounded* dengan tipe data *Trigger*, *jump* dengan tipe data *bool*, dan *speed* dengan tipe data *int*. Parameter-parameter ini memegang peran

krusial dalam penentuan alur dan respons karakter dalam animasi, memastikan tingkat respons4itas dan akurasi animasi yang tercipta.

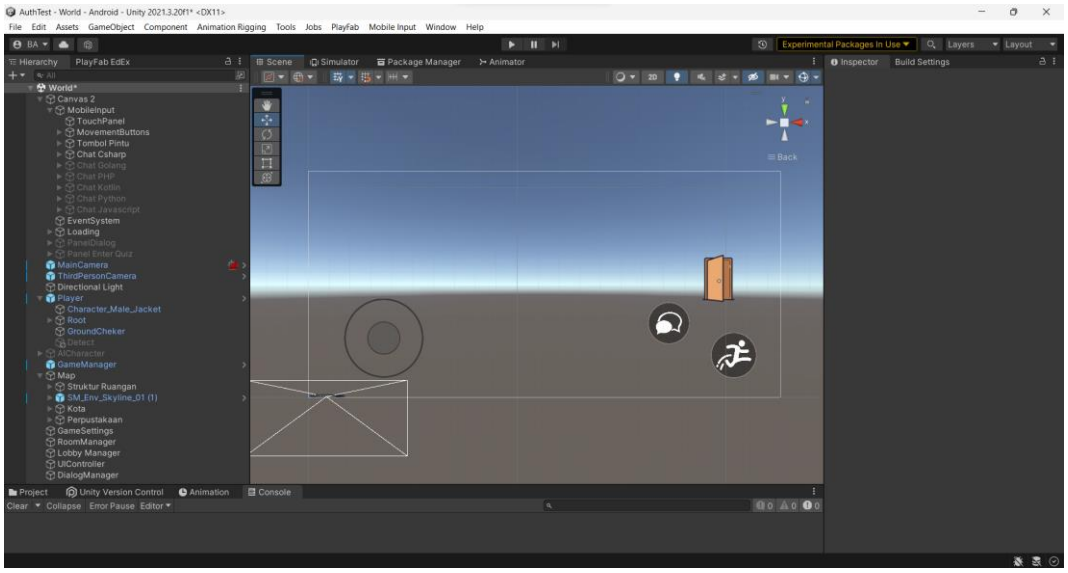
3. *Third Person Camera*



Gambar 4.25 Proses Pembuatan *Third Person Camera*

Pada Gambar 4.25 merupakan proses pembuatan kamera yakni tahap pertama, memilih objek karakter pemain telah disiapkan. Setelahnya, pada tahap kedua, objek kosong dipersiapkan di Hierarki untuk digunakan sebagai wadah kamera *virtual*, yang diwujudkan dengan memilih *Create* > *Create Empty*. Kemudian, pada tahap tersebut, objek kosong dipilih, dan komponen *Cinemachine* > *Cinemachine Virtual Camera* ditambahkan. Proses selanjutnya melibatkan penanganan hubungan antara kamera dengan pemain. Pada tahap ketiga, dalam komponen *Cinemachine Virtual Camera*, terdapat opsi "*Follow*" yang diidentifikasi, dan objek pemain diseret ke sana untuk mencapai kamera yang mengikuti pergerakan pemain. Pada tahap keempat, pengaturan kamera diperinci, termasuk pengaturan tingkat *zoom*, sudut pandang, dan rotasi sesuai dengan kebutuhan permainan. Perlu diingat bahwa pada tahap ini, opsi "*Look At*" diaktifkan, dengan pemilihan objek pemain sebagai target, untuk memastikan fokus kamera pada pemain. Tahap selanjutnya, melibatkan penambahan *Cinemachine Collider* pada objek pemain. Pada tahap ini, objek pemain yang terpilih menjadi tujuan, dan komponen *Cinemachine* > *Cinemachine Collider* ditambahkan.

Penambahan ini membuka kemungkinan untuk kamera menghindari tumpang tindih dengan objek-objek dalam lingkungan. Tahap keenam, yang merupakan tahap konfigurasi *Cinemachine Collider*, mempertimbangkan parameter seperti "*Min Distance From Target*", "*Damping*" untuk pergerakan kamera, dan "*Camera Radius*". Penyesuaian parameter ini menjadi penting untuk mencegah terjadinya tumpang tindih kamera dengan objek-objek di sekitarnya. Pada tahap berikutnya, permainan dijalankan dalam *mode Editor* untuk menguji kamera *third-person*. Tahap ini mencakup pengecekan untuk memastikan bahwa kamera mengikuti pemain dan berperilaku sesuai dengan yang diinginkan. Pengujian juga mencakup pengecekan apakah kamera mampu menghindari tumpang tindih dengan objek-objek di sekitarnya.



Gambar 4.26 Proses Pembuatan Konsol Pemain

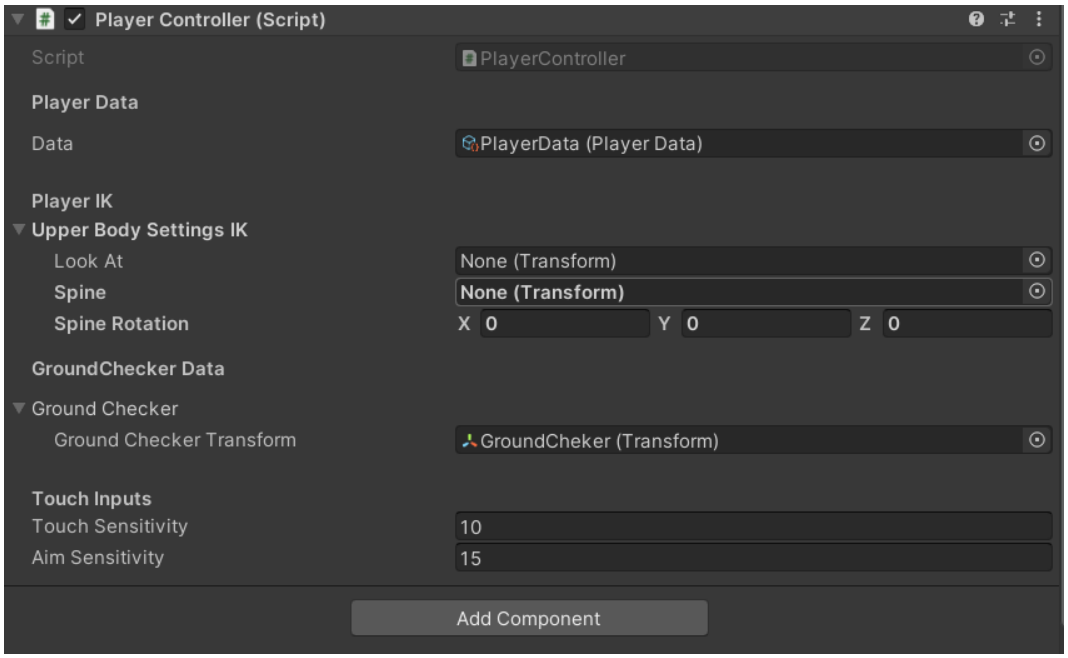
Gambar di atas menunjukkan antarmuka pengguna (*UI*) untuk menerapkan konsol pengaturan yang dibuat dengan *Unity*. UI terdiri dari beberapa elemen, yaitu:

a. Tombol membuka pintu berbentuk persegi panjang dengan warna hijau dan ikon "Open Door" di tengahnya. Tombol ini berfungsi untuk membuka pintu yang terkunci di dalam permainan. Tombol ini terletak di sudut kanan bawah layar.
b. Tombol *chat* yang berbentuk lingkaran dengan gambar ikon percakapan di dalamnya. Tombol ini berfungsi untuk berinteraksi dengan karakter non-

pemain (*NPC*) yang ada di dalam permainan. Tombol ini terletak di sebelah kiri tombol membuka pintu.

c. Tombol lompat yang berbentuk lingkaran dengan gambar ikon panah ke atas di dalamnya. Tombol ini berfungsi untuk membuat karakter pemain melompat di dalam permainan. Tombol ini terletak di sudut kiri bawah layar.

d. *Joystick* yang berbentuk lingkaran dengan garis-garis putih yang menunjukkan arah gerakan. *Joystick* ini berfungsi untuk mengontrol gerakan karakter pemain di dalam permainan. *Joystick* ini terletak di sudut kiri atas layar.



Gambar 4.27 Hasil *Output Code Player Controller*

Implementasi kontrol pemain (*PlayerController*) dalam permainan *Unity* dengan menggunakan bahasa pemrograman *C#*. Pada awal kode, *namespace* dan *class* utama, yaitu *Bagas.PlayerController*, diinisialisasi. Selanjutnya, berbagai variabel dan properti yang akan digunakan dalam kelas ini telah didefinisikan, termasuk variabel yang menyimpan data pemain, objek animator, kamera pemain, serta *rigidbody*.

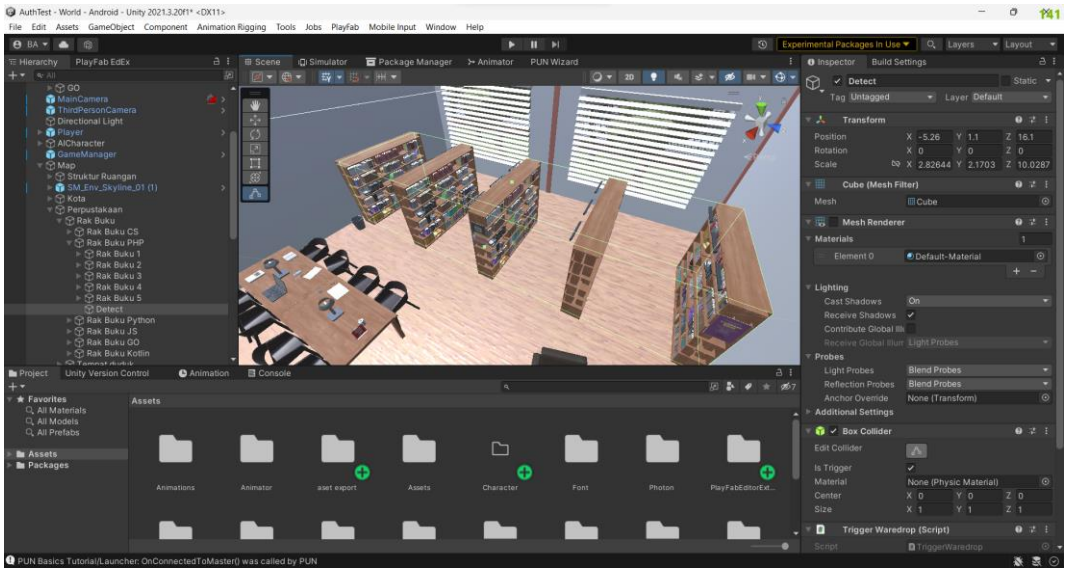
Metode *AssignComponents()* menjadi bagian yang bertanggung jawab dalam inisialisasi komponen-komponen penting seperti *animator*, *joystick*, kamera

pemain, dan *rigidbody*. Selanjutnya, terdapat metode *AssignDefaultData()* yang digunakan untuk mengatur data *default* yang terkait dengan input dari *Cinemachine*.

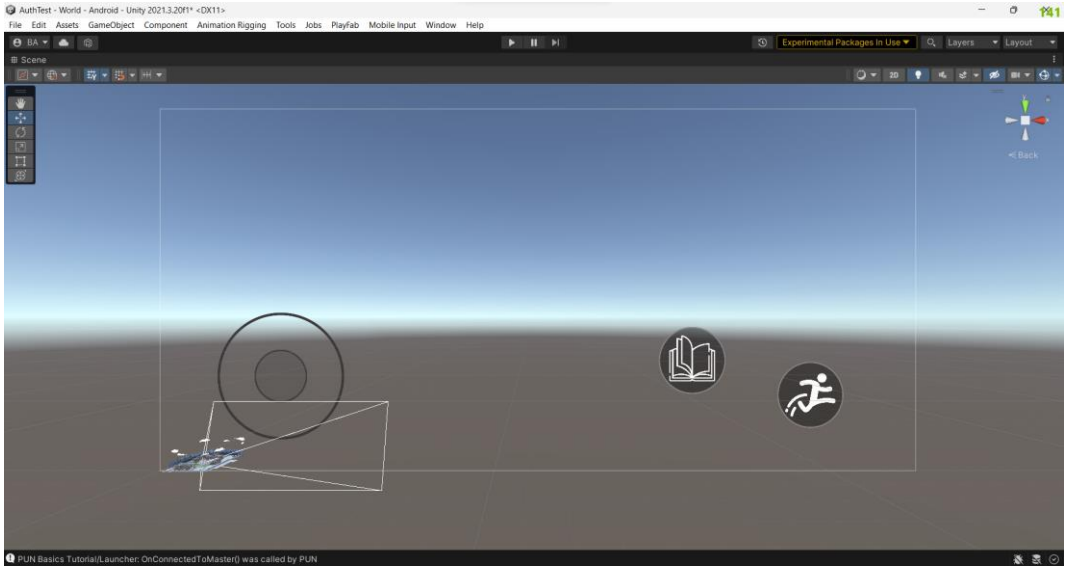
Kode ini juga mencakup metode *AssignAnimationIDs()* yang melakukan inisialisasi terhadap *ID* animasi. *ID* ini dihasilkan dari nama parameter animasi dan dimanfaatkan untuk meningkatkan performa. Beberapa metode lain seperti *GroundChecker(), JumpInput(), dan MovementInput()* dipanggil pada setiap *frame update* untuk memeriksa status pemain dan mengontrol pergerakannya.

Terakhir, metode *PlayAudioClip()* dapat diidentifikasi, yang bertanggung jawab untuk memainkan suara menggunakan komponen *AudioSource*. Secara keseluruhan, kode ini membentuk bagian dari logika kontrol pemain dan integrasi animasi dalam permainan *Unity*.

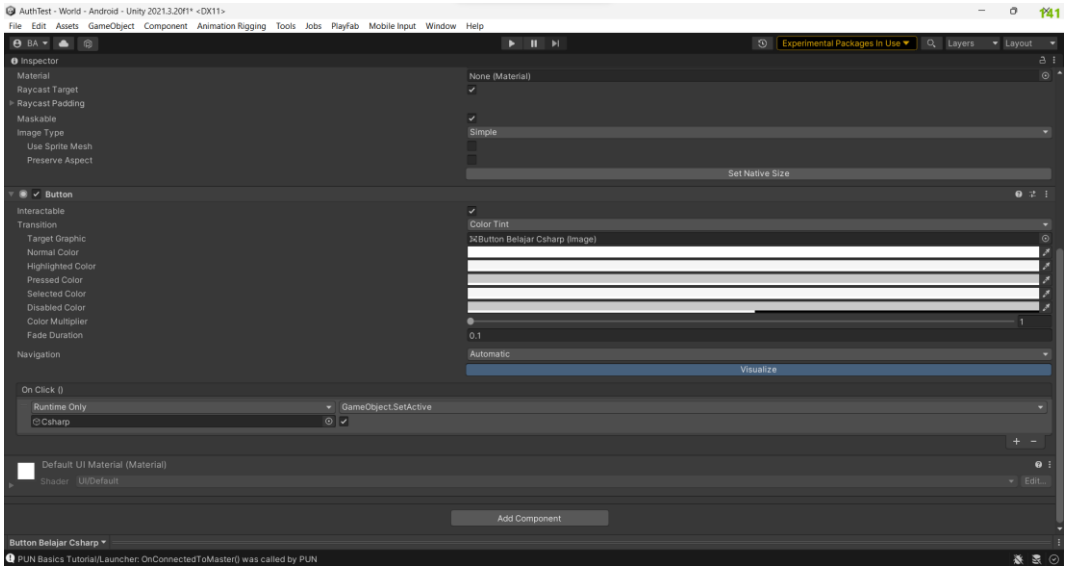
4.1.4 Pembelajaran Pemograman



Gambar 4.28 Proses Deteksi Player



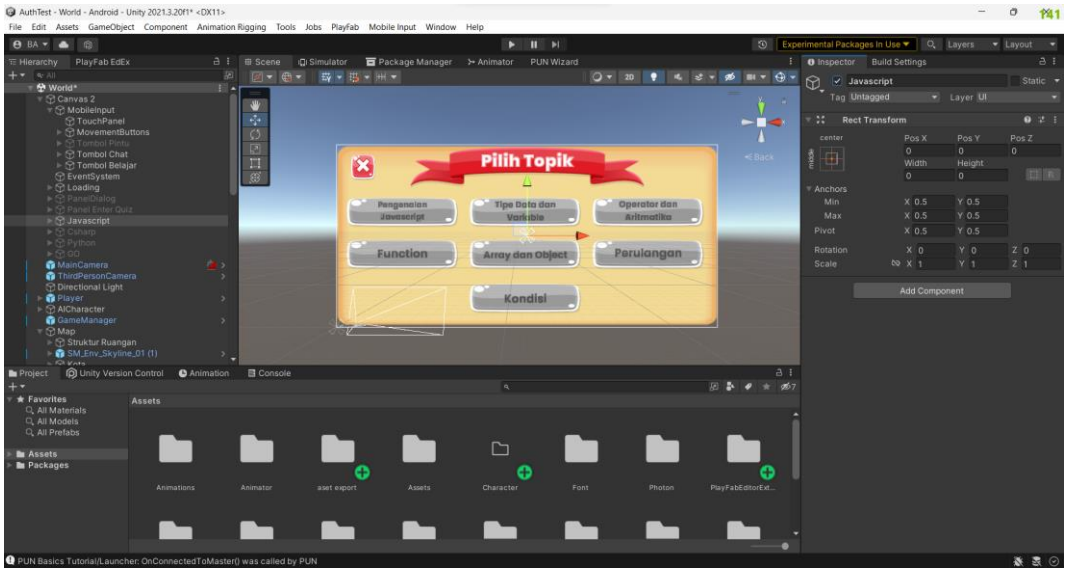
Gambar 4.29 Pembuatan Tombol Belajar



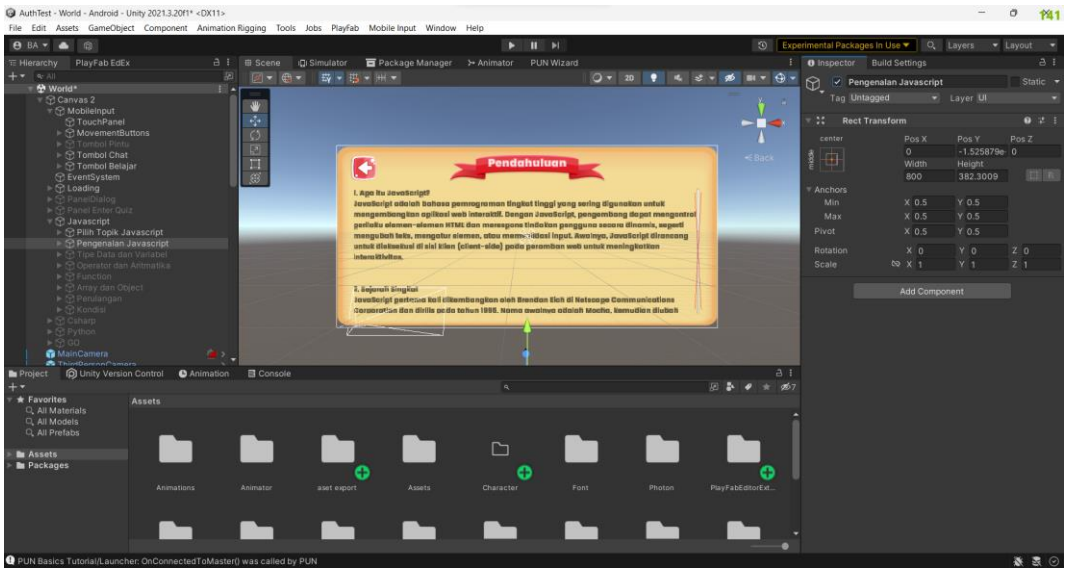
Gambar 4.30 Proses Klik Tombol Belajar

Untuk membangun pengalaman pembelajaran bahasa pemrograman yang menarik, langkah pertama adalah menciptakan kubus transparan yang akan berfungsi sebagai area interaktif. Hal ini dapat dicapai dengan membuat objek kubus melalui menu `*GameObject* > *3D Object* > *Cube*`, lalu mengatur transparansi melalui pembuatan material transparan di menu `*Assets* > *Create* > *Material*`. Setelah memiliki kubus transparan, langkah selanjutnya adalah menambahkan komponen *Collider*, seperti *Box Collider*, untuk mendeteksi interaksi dengan objek lain. Pastikan untuk mengatur "*Is Trigger*" menjadi true pada

komponen *Collider* tersebut untuk mengaktifkan mode *trigger*, yang akan memungkinkan deteksi saat objek memasuki dan meninggalkan kubus.



Gambar 4.31 Pembuatan Pemilihan Topik



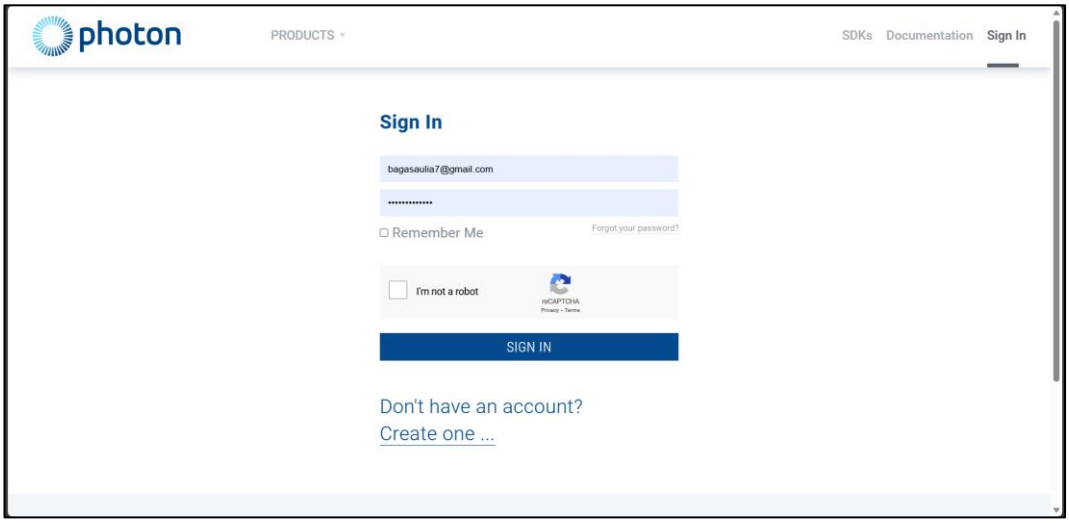
Gambar 4.32 Pembuatan Isi Topik

Setelah kubus transparan siap, langkah berikutnya adalah membuat dan mengimplementasikan skrip dalam bahasa *C#* yang akan mengatur aksi ketika objek memasuki (*Enter*) dan meninggalkan (*Exit*) kubus. Skrip ini akan mengontrol tampilan dan penyembunan tombol belajar. Anda juga dapat menyusun *UI* dengan tombol belajar dan panel pilih topik, yang akan muncul sesuai interaksi dengan kubus. Selanjutnya, hubungkan tombol belajar dengan panel pilih topik

menggunakan *event* "*On Click*" pada tombol. Dengan langkah-langkah ini, Anda akan memiliki kubus transparan sebagai area interaktif yang memunculkan dan menyembunyikan elemen-elemen *UI* yang sesuai, memberikan pengalaman pembelajaran yang interaktif dan terstruktur dalam konteks bahasa pemrograman. Sesuaikan dengan kebutuhan dan konsep pembelajaran yang ingin di implementasikan.

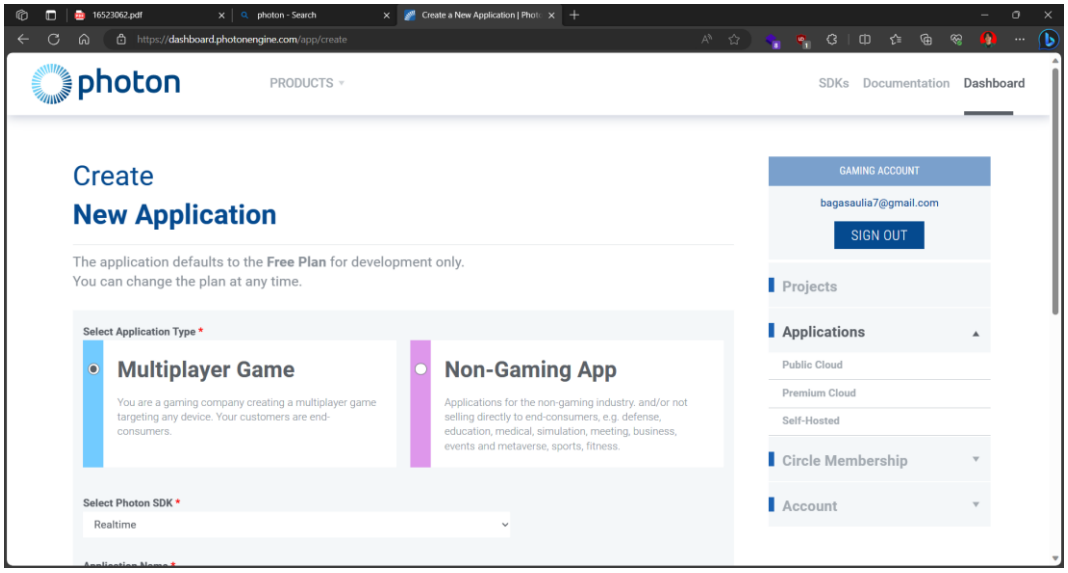
4.1.5 Kompetisi kuis

1. Konfigurasi *Photon Cloud*

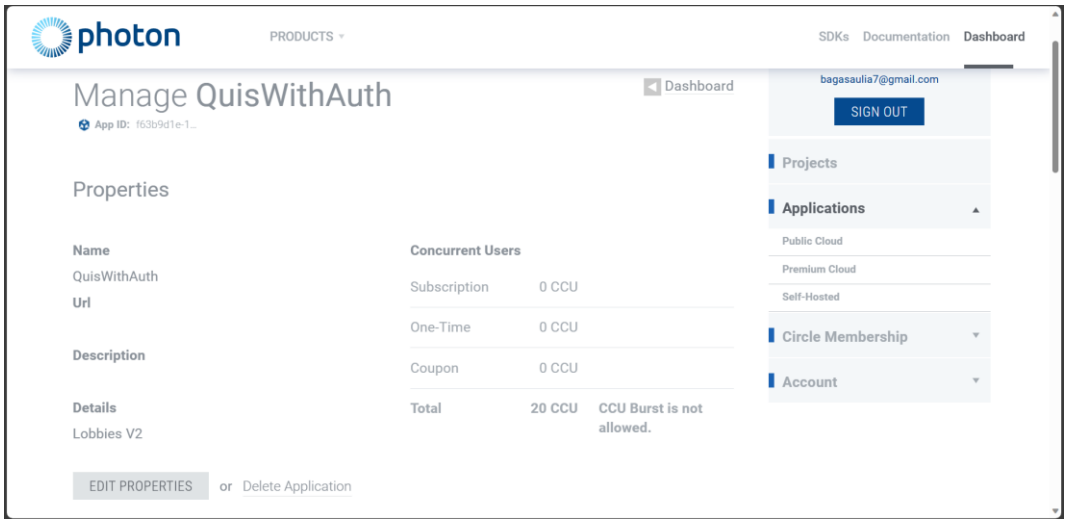


Gambar 4.33 Proses Masuk atau Daftar Akun *Photon PUN*

Pada tahap awal, dilakukan pendaftaran akun di *platform Photon* dengan memasukkan informasi penting seperti alamat *email* dan kata sandi. Proses pendaftaran ini dimaksudkan untuk membentuk identitas pengguna di *Photon*. Setelah pendaftaran berhasil, pengguna dapat melanjutkan dengan *login* ke *Photon* menggunakan kredensial yang telah didaftarkan sebelumnya. Proses *login* ini memungkinkan pengguna untuk mengakses berbagai layanan dan alat yang disediakan oleh *Photon*.



Gambar 4.34 Pembuatan Aplikasi *Cloud* Untuk *Multiplayer*

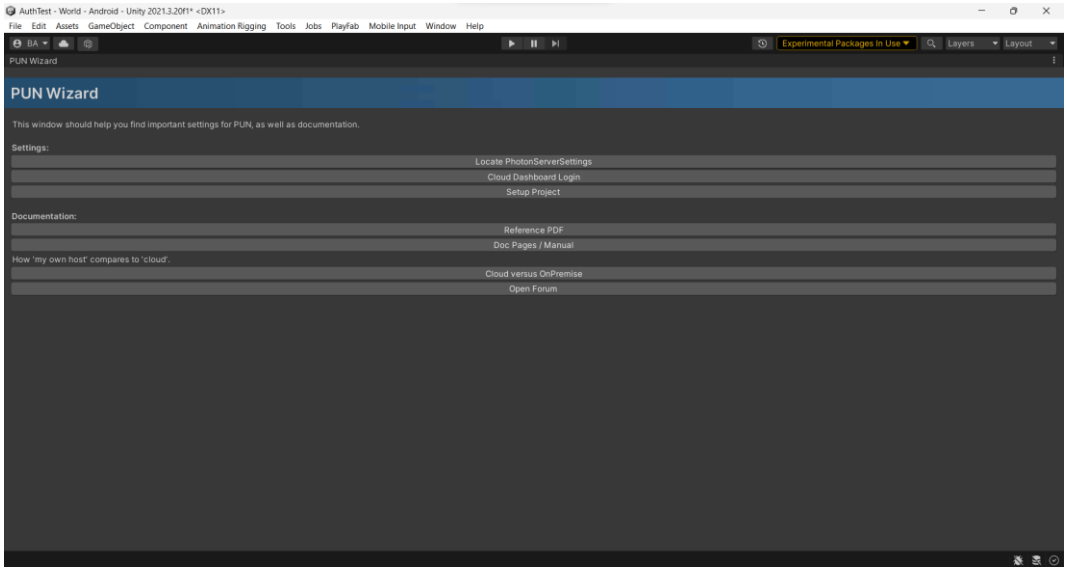


Gambar 4.35 Hasil Akhir Pembuatan Aplikasi *Cloud* Untuk *Multiplayer*

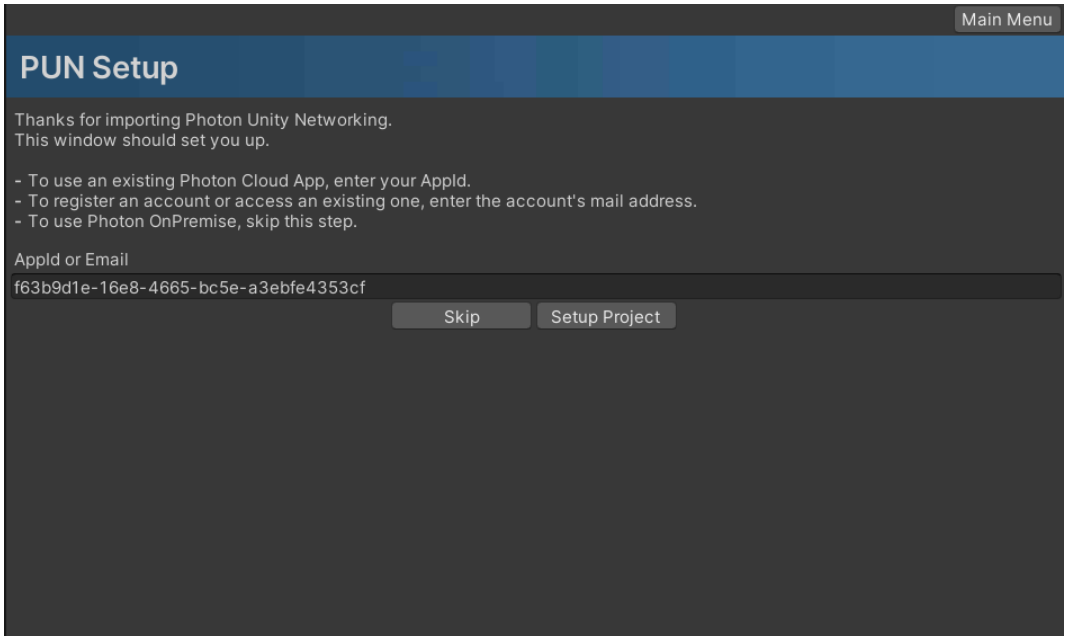
Pada akhir tahapan ini, sebuah aplikasi *cloud* untuk *multiplayer* telah berhasil dibuat di *Photon*. Nama dan deskripsi aplikasi telah ditetapkan, dan *platform* yang sesuai telah dipilih, seperti *Unity* atau *Unreal Engine*. Konfigurasi produk dan lisensi telah disesuaikan sesuai kebutuhan proyek. Selanjutnya, *SDK Photon* yang relevan untuk *platform* yang digunakan telah diunduh dan diinstal, dengan mematuhi panduan instalasi yang telah tersedia. Dengan demikian, proyek kini terhubung secara efektif dengan Photon, memungkinkan pengembangan selanjutnya pada aspek-aspek *multiplayer*.

2. *Setup Photon PUN* ke dalam *Project*

Gambar 4.36 *Photon Pun Assets*



Gambar 4.37 Tampilan *PUN Wizard*



Gambar 4.38 *Setup Project Photon PUN* Ke *Photon Cloud*

Proses pembuatan proyek *Unity* yang melibatkan penggunaan *Photon Networking* melalui *Unity Asset Store* dimulai dengan langkah pertama, yaitu *Photon* diperoleh melalui *Unity Asset Store* seperti Gambar 4.38. Pencarian dan pengunduhan *Photon PUN 2* atau versi yang diinginkan dapat dilakukan di *Unity Asset Store*. Setelah proses pengunduhan selesai, akan muncul dialog untuk memasang paket. Selanjutnya, manajemen akun *Photon* dilakukan. Jika belum memiliki akun, akun dapat dibuat di situs *web Photon*. Namun, jika sudah memiliki akun, dapat dilakukan proses *login*.

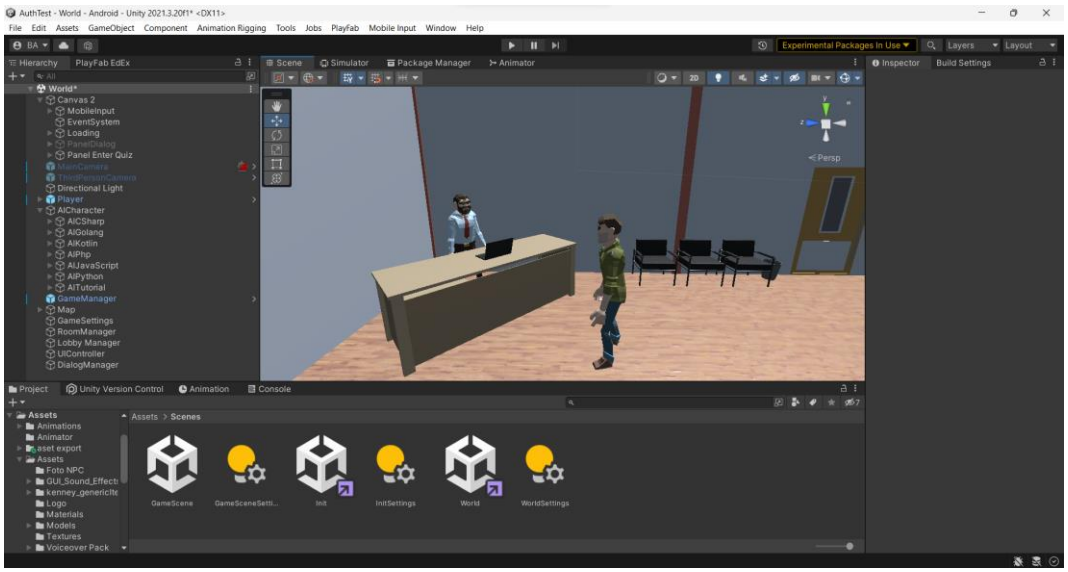
Langkah berikutnya adalah mendaftarkan aplikasi baru di *Photon* dengan membuat proyek baru. Saat mendaftar aplikasi, akan dihasilkan *AppID* yang akan digunakan sebagai kunci untuk menghubungkan proyek *Unity* dengan *server Photon*.

Setelah akun *Photon* terkait dan proyek didaftarkan, dapat kembali ke *Unity* dan membuka *Photon Control Panel* melalui *menu Window > Photon Unity Networking*. Di *AppID* yang sebelumnya diperoleh dimasukkan. Langkah ini penting untuk menghubungkan proyek *Unity* ke aplikasi *Photon* yang sesuai. Selanjutnya, dapat mulai menggunakan *Photon Networking* dalam proyek *Unity*, termasuk fitur-fitur seperti manajemen koneksi dan ruangan. Sebagai contoh, kelas

untuk mengelola koneksi dapat diciptakan dan metode Photon seperti `*PhotonNetwork.ConnectUsingSettings()*` digunakan untuk menghubungkan ke *server Photon*.

Terakhir, dapat mengintegrasikan dan memanfaatkan berbagai fitur *Photon Networking* sesuai dengan kebutuhan proyek. Ini termasuk penanganan pemain, sinkronisasi objek, dan banyak lagi. Merujuk pada dokumentasi dan panduan pengembangan yang disediakan oleh *Photon Engine* sangat disarankan untuk mendapatkan wawasan lebih mendalam tentang cara menggunakan *Photon Networking* dengan *Unity*. Proses ini memberikan fondasi yang kuat untuk memulai pengembangan permainan multipemain dengan memanfaatkan *Photon* di *Unity*.

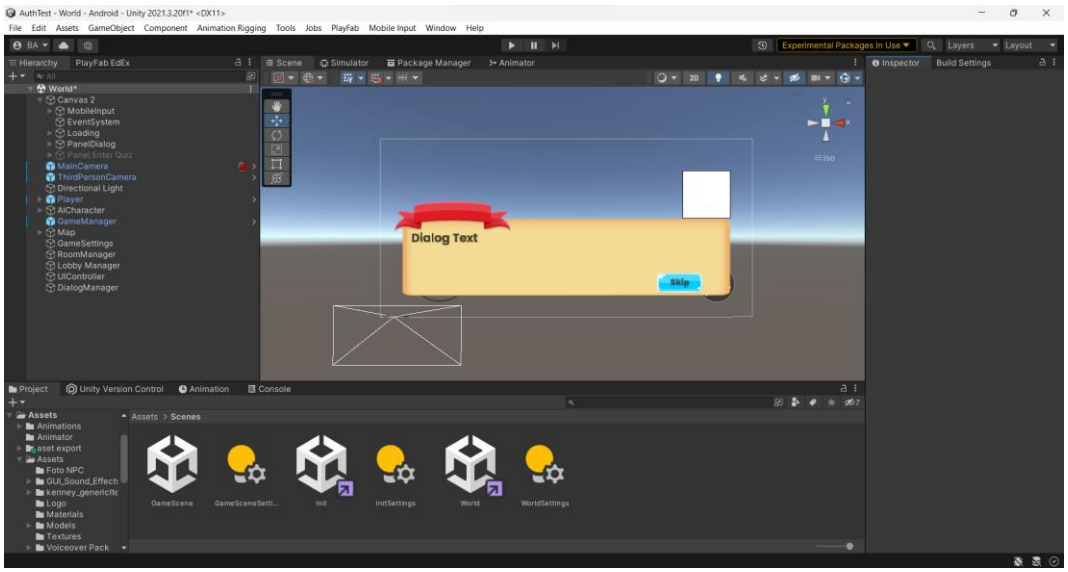
3. Pembuatan Interaksi *NPC*



Gambar 4.39 Proses Interaksi ke *NPC*

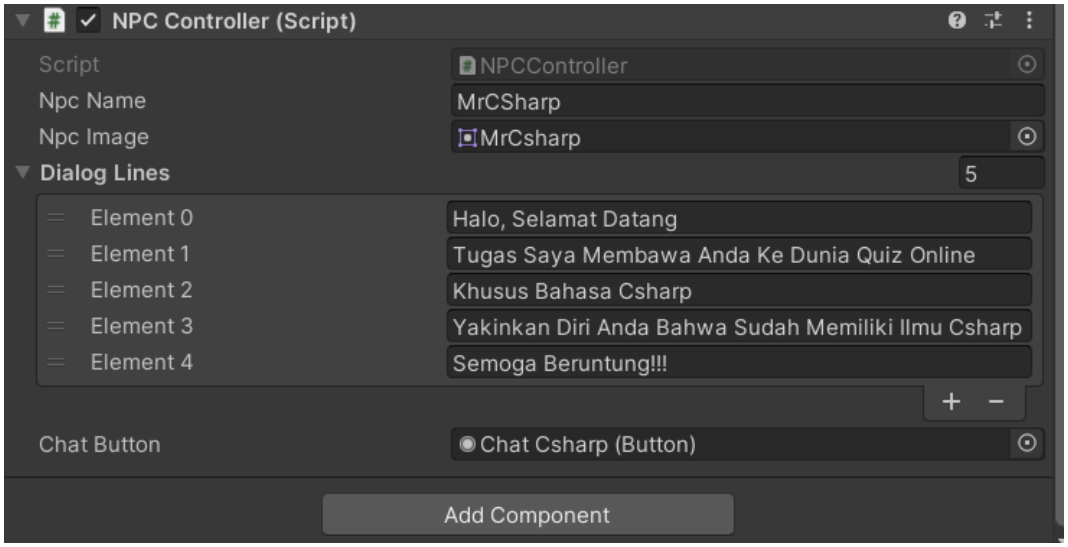
Dalam implementasi interaksi dengan *NPC* menggunakan `*OnTriggerEnter*` dan *tag*, pertama-tama Anda perlu menambahkan *tag* "*NPC*" pada objek yang mewakili karakter *NPC* dalam *Unity*. *Tag* ini digunakan untuk mengidentifikasi *NPC*. Selanjutnya, pada skrip `*NPCController*`, Anda menggunakan metode `*OnTriggerEnter*` yang diaktifkan saat pemain memasuki area *collider* yang telah ditentukan. Apabila pemain memiliki *tag* "*Player*", tombol obrolan `*chatButton*` akan ditampilkan, dan *NPC* saat ini akan diatur sebagai *NPC* yang dapat diinteraksi. Ketika pemain keluar dari area *collider*, `*OnTriggerExit*`

diaktifkan, menyebabkan tombol obrolan disembunyikan. Dengan menggunakan sistem *collider* dan *trigger* ini, interaksi antara pemain dan *NPC* dapat diatur secara efektif, memungkinkan pengalaman bermain yang lebih dinamis dan terhubung dalam permainan.

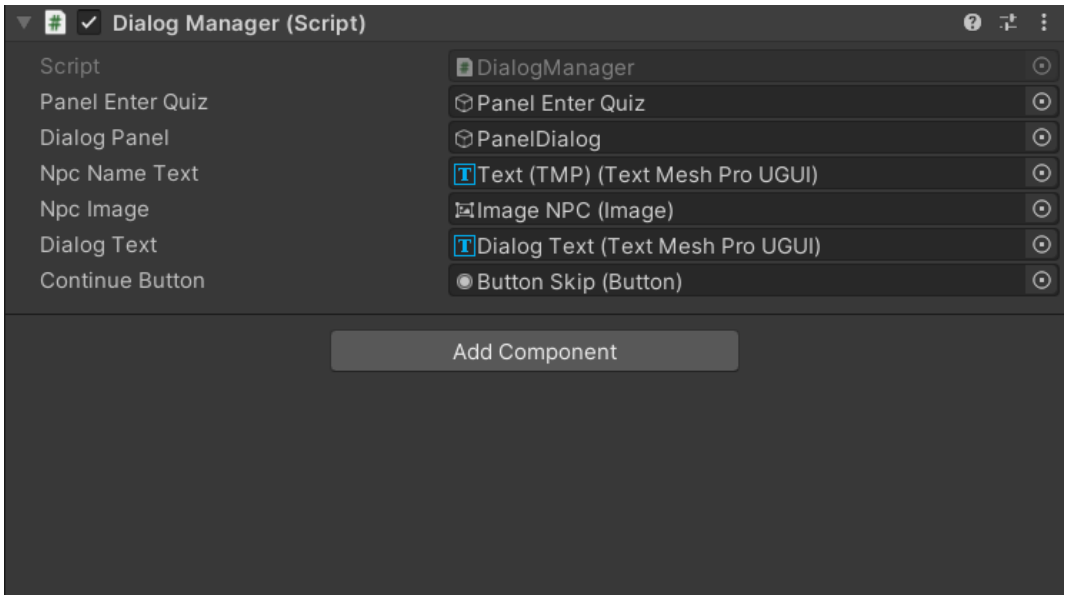


Gambar 4.40 Proses Pembuatan Dialog

Proses pengembangan *UI* dialog dimulai dengan tahap desain yang sudah ada dan mencakup penentuan tata letak, ukuran, dan elemen visual seperti ikon karakter yang terkait dengan dialog yang akan disajikan. Pada *platform* pengembangan *Unity*, digunakanlah *Canvas* sebagai dasar *UI*, di mana elemen-elemen penting seperti panel untuk menampilkan kotak dialog, *teks* untuk merepresentasikan percakapan, dan gambar karakter *NPC* ditempatkan. Manajemen konten dan tampilan dialog dilakukan melalui implementasi skrip yang mengendalikan karakter *NPC* atau peristiwa-peristiwa yang terjadi dalam permainan, memungkinan adaptasi teks dialog dan mengaktifkan serta menonaktifkan elemen-elemen *UI* dengan memperhatikan alur permainan. Ketika terdapat pilihan yang harus diambil oleh pemain, tombol atau elemen interaktif lainnya diperkenalkan untuk merespons keputusan yang dibuat oleh pemain.



Gambar 4.41 Hasil *Output Code NPC Controller*

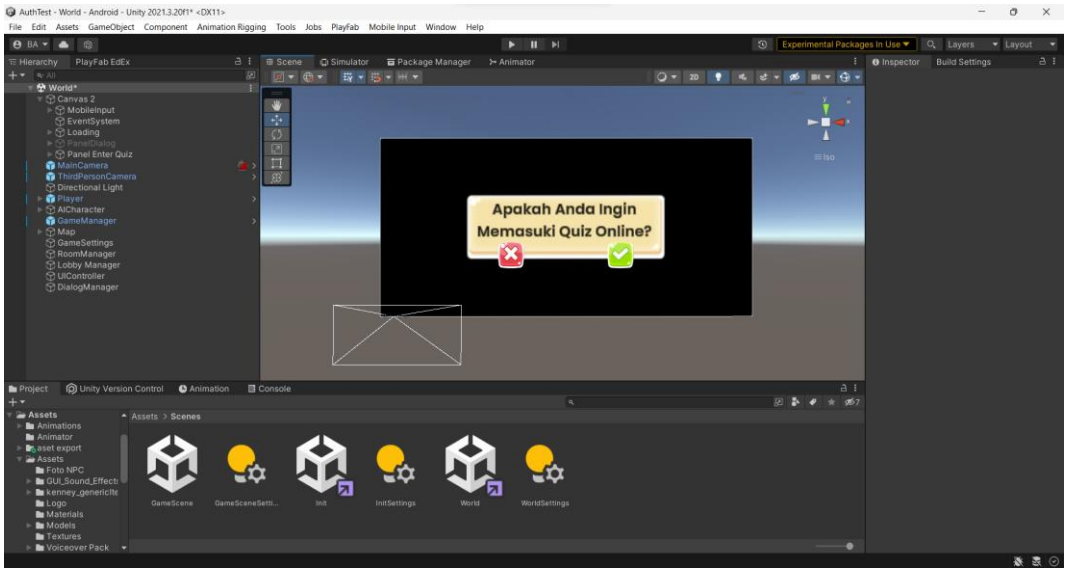


Gambar 4.42 Hasil *Output Code Dialog Manager*

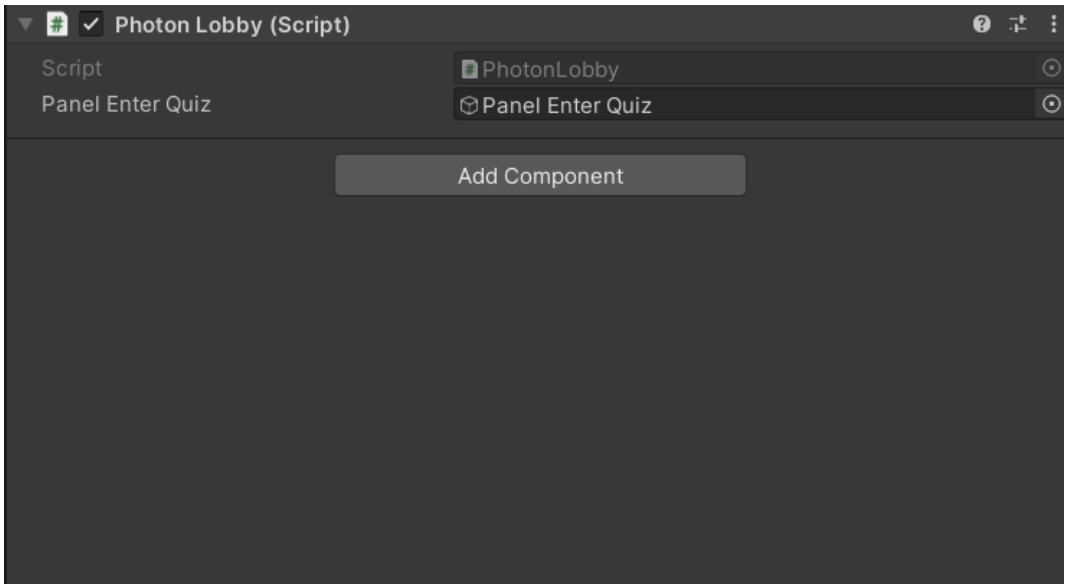
Penerapan sistem dialog dengan *NPC* dalam permainan didukung oleh peran utama dari dua kode, yaitu `*NPCController*` dan `*DialogManager*`. `*NPCController*` memiliki tanggung jawab dalam mengatur perilaku *NPC* dengan mendefinisikan nama, gambar, dan dialog yang akan dihadirkan. Prinsip desain *Singleton* digunakan untuk memastikan hanya ada satu instansi dari `*NPCController*`. Di sisi lain, `*DialogManager*` bertanggung jawab atas pengelolaan tampilan dan manajemen dialog, mengontrol antarmuka pengguna dan *UI* untuk menampilkan percakapan serta memfasilitasi pemain dalam memulai serta

melanjutkan interaksi dengan *NPC*. Kedua kode ini berinteraksi: `*NPCController*` memicu panggilan ke `*DialogManager*` untuk memulai dialog ketika pemain berada dalam jarak yang tepat, membentuk pengalaman interaktif yang memperkenalkan elemen dialog dalam permainan.

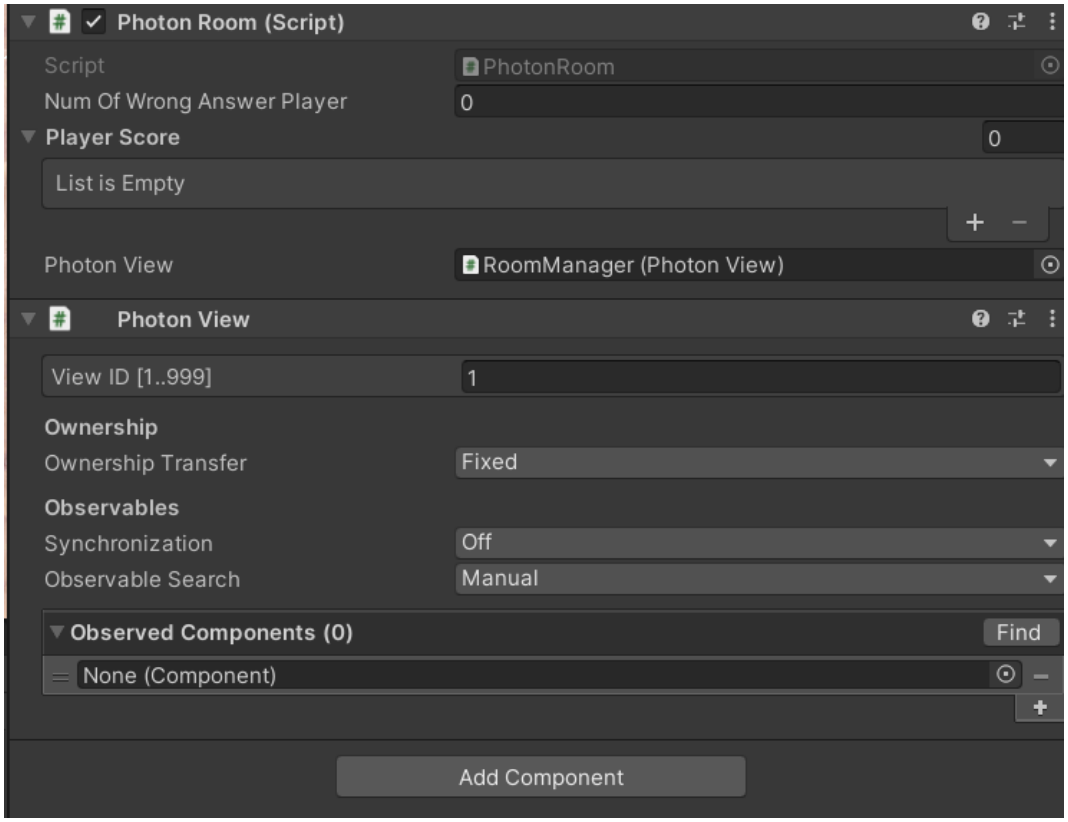
4. *Matchmaking*



Gambar 4.43 Validasi Sebelum Memasuki *Matchmaking*



Gambar 4.44 Hasil *Output Script Photon Lobby*



Gambar 4.45 Hasil *Output Code Photon Room*

Kode `*PhotonLobby*` dan `*PhotonRoom*` menjadi fokus utama dalam implementasi sistem permainan multipemain menggunakan *Photon Unity Networking (PUN)*. `*PhotonLobby*` memiliki peran krusial dalam mengelola awal koneksi ke *server Photon* dan menginisiasi proses permainan. Pada tahap awal, panggilan `*Start()*` digunakan untuk memulai koneksi dengan *server Photon* melalui `*PhotonNetwork.ConnectUsingSettings()*`.

Selanjutnya, pengaturan `*PhotonNetwork.AutomaticallySyncScene*` memungkinkan sinkronisasi otomatis antar *scene* saat pemain berpindah ruangan. Seiring itu, fungsi `*OnConnected()*` mempersiapkan antarmuka pengguna dengan menghentikan tampilan *loading* dan mengambil nama pemain dari penyimpanan lokal.

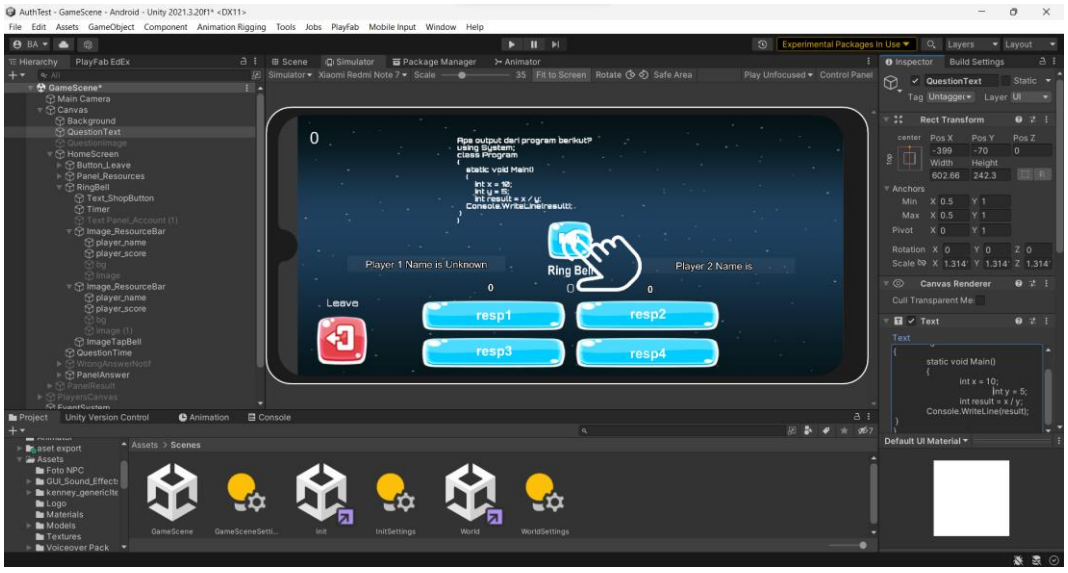
Sementara itu, `*PhotonRoom*` mengambil peran penting dalam manajemen ruangan selama permainan berlangsung. Kode ini mencakup implementasi beberapa fungsi *callback Photon* yang vital, seperti `*OnJoinedRoom()*` yang dipanggil saat pemain berhasil memasuki ruangan, dan `*OnPlayerEnteredRoom()*`

yang terpanggil ketika pemain baru bergabung. `*CheckPlayer()*` digunakan untuk memeriksa apakah jumlah pemain memadai untuk memulai permainan, dan jika cukup, `*StartGame()*` dimulai untuk menginisiasi permainan.

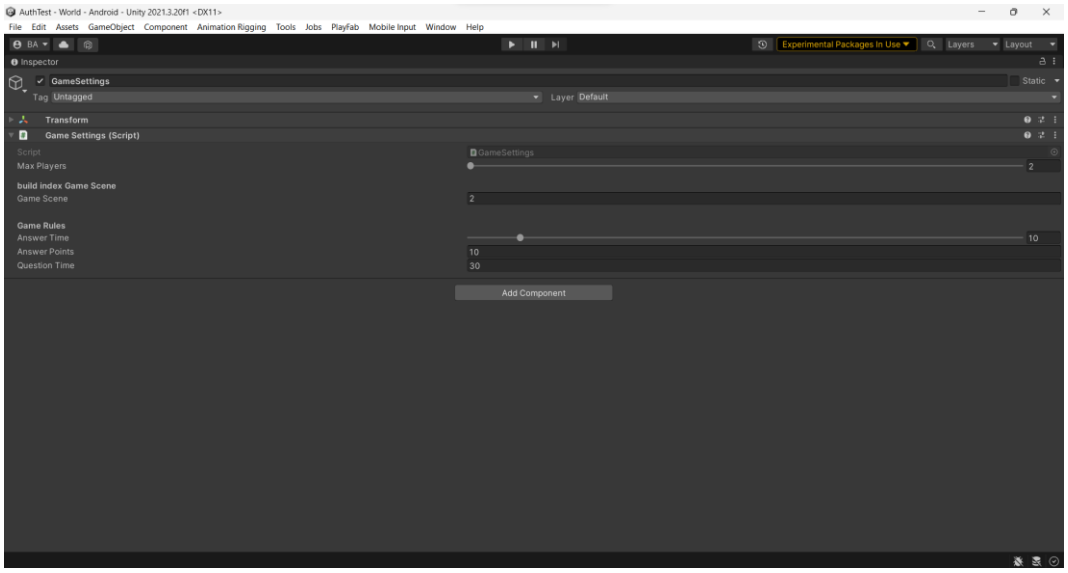
Penggabungan kedua komponen ini memungkinkan terwujudnya interaksi multipemain yang terkoordinasi. Ketika jumlah pemain mencapai batas maksimum yang diinginkan, permainan dapat dimulai dan semua pemain memasuki *scene* permainan. Di dalam *scene* tersebut, `*PhotonRoom*` menggunakan `*RPC_CreatePlayer()*` untuk menciptakan objek pemain dan memulai permainan. Selain itu, sistem ini memungkinkan pembaruan skor melalui `*RPC_UpdateScore()*` dan menanggapi situasi khusus, seperti pemain yang meninggalkan permainan atau ruangan, melalui *callback* seperti `*OnPlayerLeftRoom()*`.

Secara keseluruhan, kolaborasi yang baik antara `*PhotonLobby*` dan `*PhotonRoom*` menghasilkan sistem permainan multipemain yang responsif dan terkoordinasi. *Photon Unity Networking* membantu mewujudkan pengalaman bermain yang kompetitif dan mengasyikkan bagi pemain, di mana mereka dapat berinteraksi dan bermain bersama dalam permainan multipemain yang menarik.

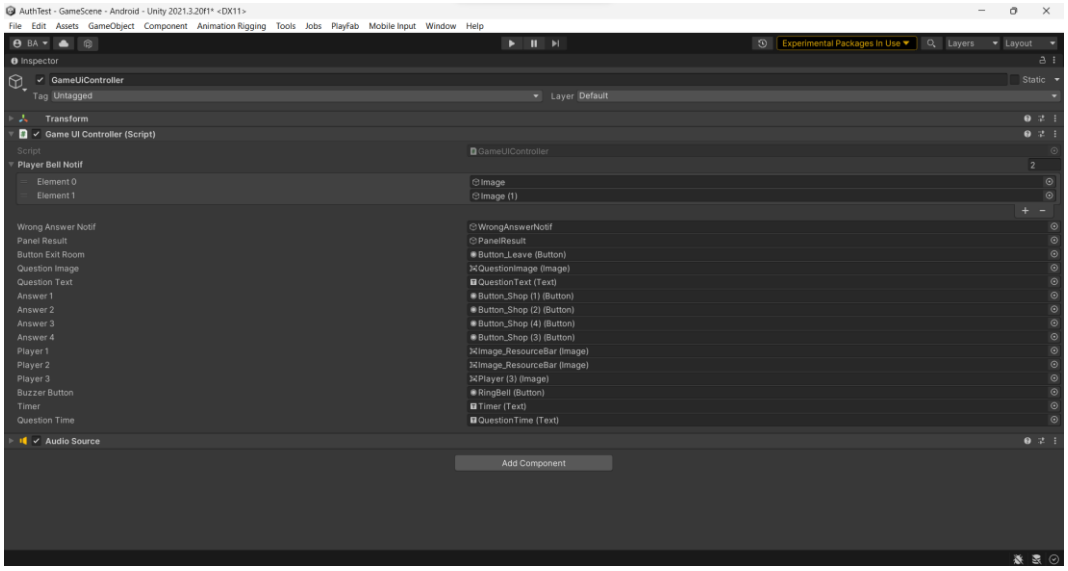
5. Mekanisme Kuis



Gambar 4.46 Proses Pembuatan *UI* dan Sistem Kuis



Gambar 4.47 Hasil *Output Script Game Settings*



Gambar 4.48 Hasil *Output Script Game UI Controller*

Mekanisme permainan diawali dengan fase persiapan. Pemain bergabung ke dalam permainan melalui layanan *Photon Network*, dan jumlah pemain yang dibutuhkan disesuaikan dengan konfigurasi *GameSettings*. Setiap pemain diberi nama pengguna yang disimpan dalam *PlayerPrefs* untuk keperluan identifikasi. Apabila jumlah pemain mencukupi, permainan dimulai dan statusnya diubah menjadi "*Start*."

Selama permainan, pengelolaan berbagai aspek permainan menjadi tanggung jawab dari *Photon Room*. Ini mencakup manajemen skor pemain,

pemantauan jumlah pemain, serta penyesuaian status permainan sesuai dengan kondisi saat itu. Sebagai contoh, setelah semua pemain bergabung, permainan dianggap telah "*start*," dan tidak ada lagi pemain baru yang diperbolehkan untuk masuk.

Peran kunci dalam menyajikan antarmuka pengguna permainan kepada pemain dimiliki oleh *GameUIController*. Antarmuka pengguna ini meliputi tampilan pertanyaan, opsi jawaban, tombol *buzzer*, sisa waktu untuk menjawab pertanyaan, dan skor pemain. Pemain dapat memilih jawaban dengan mengklik tombol yang sesuai, yang akan memicu perhitungan skor berdasarkan kebenaran jawaban tersebut. Selain itu, pemain dapat menggunakan tombol buzzer untuk memberikan jawaban dengan segera. *GameUIController* juga memantau sisa waktu untuk setiap pertanyaan dan mengelola perhitungan mundur.

Koordinasi pertanyaan dan opsi jawaban selama permainan diatur oleh *QuestionController*. Data pertanyaan telah disiapkan sebelumnya dalam bentuk kumpulan data pertanyaan yang beragam. Selama permainan, *QuestionController* secara berkala mengambil pertanyaan dari kumpulan data ini dan menampilkannya kepada pemain. Pertanyaan dapat berupa teks atau berbasis gambar, dan pemain harus memilih jawaban yang menurut mereka benar. Pilihan jawaban ini akan mempengaruhi akumulasi skor pemain.

Situasi khusus seperti pemain keluar dari permainan atau kehilangan koneksi ditangani oleh *Photon Room* selama permainan. Jika pemain keluar dari permainan, hal ini ditangani dengan baik oleh *Photon Room*, dan permainan dapat berlanjut tanpa kehadiran pemain tersebut. Kehilangan koneksi juga ditangani dengan bijak, dan kehilangan koneksi tidak akan mempengaruhi jalannya permainan secara negatif.

Pada akhir permainan, setelah semua pertanyaan telah dijawab, permainan mencapai tahap "*ended*." Hasil permainan, termasuk pemenang dan pemain dengan skor tertinggi, dipresentasikan kepada semua pemain. Apabila terjadi hasil imbang, hal ini juga diberitahukan kepada pemain. *GameUIController* memegang peranan penting dalam menampilkan hasil ini kepada pemain, memberikan pemahaman yang jelas kepada pemain tentang pencapaian mereka dalam permainan tersebut.

Secara keseluruhan, mekanisme permainan ini dirancang untuk menciptakan pengalaman bermain kuis yang mengasyikkan, terkoordinasi, dan menyenangkan. Kolaborasi erat antara *QuestionController*, *GameUIController*, dan *Photon Room* memastikan bahwa alur permainan berjalan dengan mulus dan responsif terhadap tindakan pemain. Dengan demikian, permainan ini memberikan pengalaman bermain kuis yang menarik dan seru bagi semua pemain yang terlibat.

4.2 *Testing*

4.2.1 *Alpha Testing*

Alpha testing merupakan tahap pengujian internal yang dilakukan oleh tim pengembang atau tim khusus yang bertanggung jawab untuk mengembangkan permainan. Pada tahap ini, permainan belum sepenuhnya selesai dan mungkin memiliki *bug*, kecacatan, atau fitur yang belum lengkap. Tujuan *alpha testing* adalah untuk mengidentifikasi masalah dan memperbaiki mereka sebelum permainan mencapai tahap *beta testing*.

1. Hasil Pengujian *Black Box*

Hasil dari pengujian *black-box* menunjukkan bahwa output yang dihasilkan pada setiap adegan sudah sesuai dengan harapan yang telah ditetapkan. Proses pengujian mencakup evaluasi tiga adegan kunci, yaitu menu utama, Dunia Terbuka, dan adegan Kuis, yang dianalisis secara rinci mulai dari Tabel 4.1 hingga Tabel 4.3. Evaluasi ini bertujuan untuk memastikan bahwa respons dan tampilan pada setiap adegan memenuhi standar kualitas yang diinginkan sebelumnya.

Tabel 4.1 Hasil Pengujian *Scene* Menu Utama

No	Skenario	Hasil yang di harapkan	Hasil
1	Masuk ke dalam *game*	Tampilan halaman Utama, *backsound*.	Sesuai dengan hasil yang diharapkan
2	Tekan Tombol "*Sign Up*"	Tampilan *Login*, kolom *email*, kolom kata sandi, tombol *login*, tombol daftar.	Sesuai dengan hasil yang diharapkan

3	Tekan tombol “Sini”	Tampilan daftar akun, kolom *email*, kolom nama, kolom kata sandi, dan tombol daftar	Sesuai dengan hasil yang diharapkan
4	Tekan tombol “Daftar”	Tampilan Berhasil atau gagal, dan tampilan Beranda	Sesuai dengan hasil yang diharapkan
5	Tekan Tombol “Keluar”	Tampilan *Menu* Keluar berisi Keluar akun, keluar *game*, dan batalkan keluar	Sesuai dengan hasil yang diharapkan
6	Tekan Tombol “Main”	Tampilan Scene Dunia Terbuka	Sesuai dengan hasil yang diharapkan

Tabel 4.2 Hasil Pengujian *Scene* Dunia Terbuka

No	Skenario	Hasil yang di harapkan	Hasil
1	Geser *analog* pergerakan	Karakter utama Bergerak sesuai arah *analog*	Sesuai dengan hasil yang diharapkan
2	Geser bagian kanan perangkat	Arah kamera berubah	Sesuai dengan hasil yang diharapkan
3	Tekan Tombol ikon “Lompat”	Karakter utama akan melompat	Sesuai dengan hasil yang diharapkan
4	Tekan tombol ikon “Tombol Pintu”	Pintu akan terbuka	Sesuai dengan hasil yang diharapkan
5	Tekan tombol ikon “Percakapan”	Tampilan dialog percakapan muncul	Sesuai dengan hasil yang diharapkan

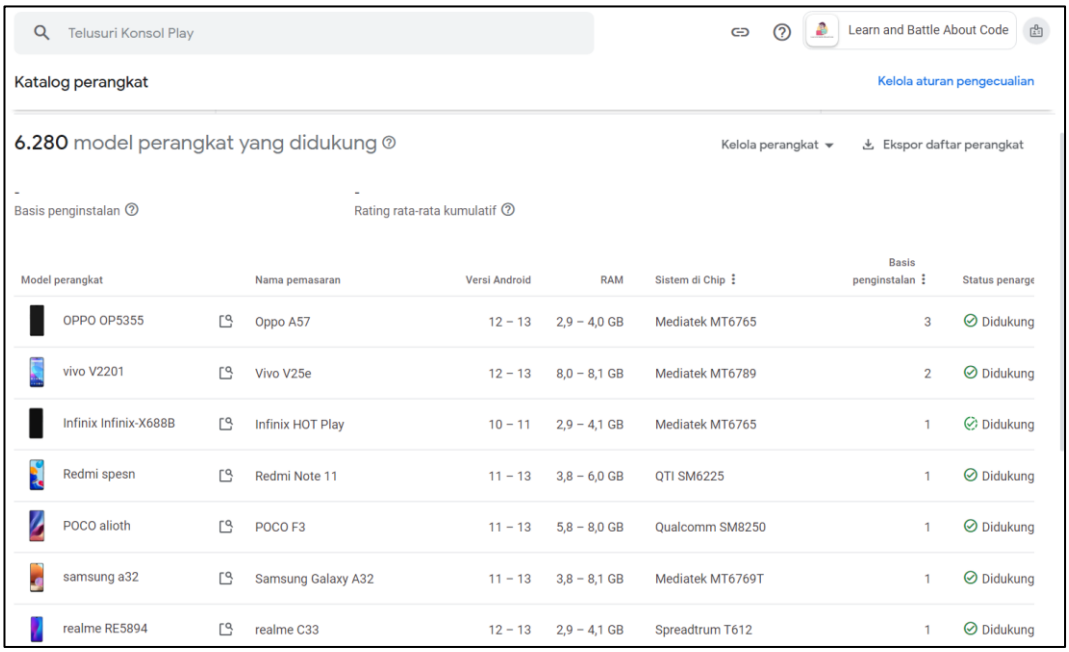
6	Tekan tombol "*skip*"	Dialog akan di lompati	Sesuai dengan hasil yang diharapkan
7	Tekan tombol ikon "Belajar"	Tampilan pemilihan topik pembelajaran	Sesuai dengan hasil yang diharapkan
8	Tekan tombol ikon "tutup topik"	Tampilan dunia terbuka	Sesuai dengan hasil yang diharapkan
9	Tekan tombol "Pilihan topik"	Tampilan Pembelajaran	Sesuai dengan hasil yang diharapkan
10	Tekan tombol ikon "tutup pembelajaran"	Tampilan pilihan topik	Sesuai dengan hasil yang diharapkan
11	Seret isi tulisan pembelajar	Tampilan pembelajaran yang di bawah	Sesuai dengan hasil yang diharapkan
12	Tekan tombol ikon "masuk kuis"	Tampilan Menunggu *player* lain	Sesuai dengan hasil yang diharapkan
13	Tekan tombol ikon "batal kuis"	Tampilan dunia terbuka	Sesuai dengan hasil yang diharapkan
14	Tekan tombol "Lanjut"	Tampilan lanjutan panduan *game*	Sesuai dengan hasil yang diharapkan
15	Tekan tombol "batalkan"	Tampilan dunia terbuka	Sesuai dengan hasil yang diharapkan

Tabel 4.3 Hasil Pengujian Kuis *Online*

No	Skenario	Hasil yang di harapkan	Hasil
1	Tekan Tombol ikon "*bell*"	Akses tombol menjawab akan di buka	Sesuai dengan hasil yang diharapkan
2	Tekan Tombol "Respon Jawaban"	Skor Akan Bertambah Ketika respon benar dan tampilan salah muncul ketika respon salah	Sesuai dengan hasil yang diharapkan
3	Tekan tombol ikon "keluar"	Tampilan pertanyaan	Sesuai dengan hasil yang diharapkan
4	Tekan tombol ikon "iya"	Tampilan dunia terbuka	Sesuai dengan hasil yang diharapkan
5	Tekan tombol ikon "tidak"	Tampilan kuis online	Sesuai dengan hasil yang diharapkan
6	Tekan tombol ikon "Rumah"	Tampilan pemilihan topik pembelajaran	Sesuai dengan hasil yang diharapkan
7	Menampilkan Nama lawan	Nama lawan beserta skor lawan	Sesuai dengan hasil yang diharapkan

Setelah melakukan pengujian pada 3 adegan dengan total 21 uji kasus, hasilnya menunjukkan bahwa kinerja *game* memenuhi harapan yang telah ditetapkan sebelumnya. Dalam konteks ini, dapat dipastikan bahwa *game* telah mencapai tingkat kesiapan yang memungkinkannya untuk dijalankan sesuai dengan fungsionalitas yang diinginkan. Oleh karena itu, *game* memiliki kualitas yang memadai untuk melanjutkan ke tahap pengujian *beta* dengan keyakinan bahwa potensi pengembangan selanjutnya dapat dioptimalkan.

2. Hasil Pengujian Perangkat



Gambar 4.49 Hasil Pengujian Perangkat

Berdasarkan hasil pengujian, dapat disimpulkan bahwa game dapat berjalan optimal pada 6.280 model perangkat yang diuji langsung oleh Google Play. Penyesuaian ini bertujuan untuk memastikan tampilan yang konsisten dan tidak mengganggu pengalaman bermain terhadap pengguna yang memiliki perangkat yang berbeda. Dengan demikian, pengembang game telah melakukan upaya maksimal dalam memastikan bahwa game ini memberikan kualitas yang seragam dan tanpa gangguan, bahkan ketika dijalankan pada berbagai jenis perangkat yang beragam. Hal ini penting untuk memastikan bahwa pengguna dengan berbagai perangkat dapat menikmati game ini dengan optimal, sehingga pengalaman bermain mereka tetap lancar dan menyenangkan.

4.2.2 *Beta Testing*

Pengujian melibatkan partisipasi dari total 20 mahasiswa, di antaranya terdiri dari pemula dan mahasiswa dengan tingkat keahlian lanjut dalam pemrograman. Metode yang digunakan dalam pengujian adalah *System Usability Scale* (SUS) dengan 10 pernyataan SUS yang telah diadaptasi khusus untuk menilai pengalaman pengguna saat memainkan *game*. Pertanyaan-pertanyaan ini diberikan

kepada para peserta setelah mereka bermain *game* selama kurang lebih 30 menit. Setelah pengumpulan data dari responden, dilakukan perhitungan untuk menganalisis hasilnya. Berikut adalah daftar pertanyaan dari *System Usability Scale* (SUS).

1. Saya berencana untuk memainkan *game* ini Kembali
2. Saya kesulitan dalam memainkan *game* ini
3. Saya Merasa kesulitan belajar saat memainkan *game* ini
4. saya berpengalaman terkait dengan fitur-fitur yang ada di dalam *game* ini dan semuanya berjalan sesuai dengan harapan.
5. saya mengira adanya inkonsistensi atau hal-hal yang tidak selaras dalam elemen-elemen
6. saya dengan cepat memahami cara memainkan *game* ini
7. saya merasa kebingungan saat memainkan *game* ini
8. saya merasa ada hambatan yang dirasakan saat memainkan *game* ini
9. saya merasa perlu berlatih terlebih dahulu sebelum memainkan *game* ini
10. saya merasa *game* ini tidak pantas.

Untuk sistem penilaian dari pernyataan positif di atas adalah sebagai berikut:

a. Sangat Setuju (Diberi nilai 5)
b. Setuju (Diberi nilai 4)
c. Netral (Diberi nilai 3)
d. Tidak Setuju (Diberi nilai 2)
e. Sangat Tidak Setuju (Diberi nilai 1)

Untuk sistem penilaian dari pernyataan negatif di atas adalah sebagai berikut:

a. Sangat Tidak Setuju (Diberi nilai 5)
b. Tidak Setuju (Diberi nilai 4)
c. Netral (Diberi nilai 3)
d. Setuju (Diberi nilai 2)
e. Sangat Setuju (Diberi nilai 1)

Pengujian dilakukan dengan metode tatap muka, dan kuesioner diisi melalui Google Form dan *Game* diunggah ke Google Drive. Pengujian berlangsung pada tanggal 22 September 2023. Tabel 4.5 akan memuat data responden berupa nama dan NIM mahasiswa yang terlibat dalam pengujian.

Tabel 4.4 Data Responden

No	Nama Responden	NIM	Universitas
1	Prabowo Nofieldi	1301184477	Universitas Telkom
2	Putri Nurhidayanti	190170051	Universitas Malikussaleh
3	Firauza Alif Firdaus	18.82.0455	Universitas Amikom Yogyakarta
4	Aulia Abdurrafi	7708190058	Universitas Telkom
5	Nurhamni	190170050	Universitas Malikussaleh
6	Abitdavy Athallah Muhammad	1810511079	Universitas Pembangunan Nasional Veteran Jakarta
7	Dimas Pratama Faesta	19.82.0647	Universitas Amikom Yogyakarta
8	Nurul Huda	201904017	Politeknik Enjinering Indorama

9	Farid Nabil Firdaus	201904006	Politeknik Enjinering Indorama
10	Dendy Ramdhani	19552011134	Sekolah Tinggi Teknologi Bandung
11	Sirajuddin Hawari	1810130011	Sekolah Tinggi Ilmu Manajemen dan Ilmu Komputer ESQ
12	Koir Herianto	H191600499	Politeknik Pertanian Negeri Samarinda
13	Aurandri Tazkia Bagaskara	5190411482	Universitas Teknologi Yogyakarta
14	Octa Travelian Purnomo	1901010035	Universitas Bumigora
15	Christian Bernard Kuswandi	2301919706	Universitas Bina Nusantara
16	Miguel Yehezkiel Noya	2301938850	Universitas Bina Nusantara
17	Rizky Chandra Purnama Santosa	19.82.0774	Universitas Amikom Yogyakarta
18	Irawan Tanudirdjo	16691315	Universitas Terbuka
19	Yulian Bagus Sadewo	57418517	Universitas Gunadarma

20	Husein Alfikri	20419094	Sekolah Tinggi Manajemen Informatika dan Komputer Jakarta STI&K

Hasil pengujian usabilitas mencerminkan pandangan para mahasiswa yang memiliki kemampuan pemograman terkait dengan game "Learn and Battle About Code" setelah mereka selesai memainkannya. Tabel 4.6 di bawah ini memperlihatkan nilai yang diperoleh pada setiap pertanyaan yang diberikan kepada responden.

Tabel 4.5 Hasil Pengujian *Usability* Mahasiswa Unersitas Malikussaleh

No	Nama Responden	Pernyataan										Total	Skor SUS (Total x 2,5)
		P1	P2	P3	P4	P5	P6	P7	P8	P9	P10		
1	Prabowo Nofieldi	3	4	3	3	2	3	2	3	2	4	29	73
2	Putri Nurhidayanti	2	2	3	2	2	3	2	2	2	3	23	58
3	Firauza Alif Firdaus	4	3	3	3	2	4	4	3	2	4	32	80
4	Aulia Abdurrafi	2	2	2	2	2	2	2	2	2	2	20	50
5	Nurhamni	4	4	4	4	4	3	4	4	3	4	38	95
6	Abitdavy Athallah Muhammad	4	4	4	4	2	4	3	3	2	4	34	85
7	Dimas Pratama Faesta	4	3	3	4	3	4	4	4	1	4	34	85
8	Nurul Huda	3	4	4	4	2	3	4	2	4	3	33	83
9	Farid Nabil Firdaus	2	3	3	3	4	3	4	4	3	4	33	83
10	Dendy Ramdhani	3	2	3	3	3	2	3	3	1	3	26	65
11	Sirajuddin Hawari	3	3	3	3	3	3	3	3	1	3	28	70
12	Koir Herianto	4	3	4	4	4	4	4	4	1	4	36	90
13	Aurandri Tazkia Bagaskara	4	2	2	3	3	4	3	3	1	4	29	73

14	Octa Travelian Purnomo	4	3	2	4	2	3	2	2	1	3	26	65
15	Christian Bernard Kuswandi	3	2	3	2	3	3	3	2	3	4	28	70
16	Miguel Yehezkiel Noya	3	3	3	4	4	4	3	3	3	4	34	85
17	Rizky Chandra Purnama Santosa	4	3	3	2	3	3	3	3	2	4	30	75
18	Irawan Tanudirdjo	4	3	3	4	3	4	3	3	3	4	34	85
19	Yulian Bagus Sadewo	3	3	2	3	3	2	3	3	3	3	28	70
20	Husein Alfikri	3	4	3	3	2	3	2	3	2	4	29	73
Skor Rata-rata (Hasil Akhir)													**72**

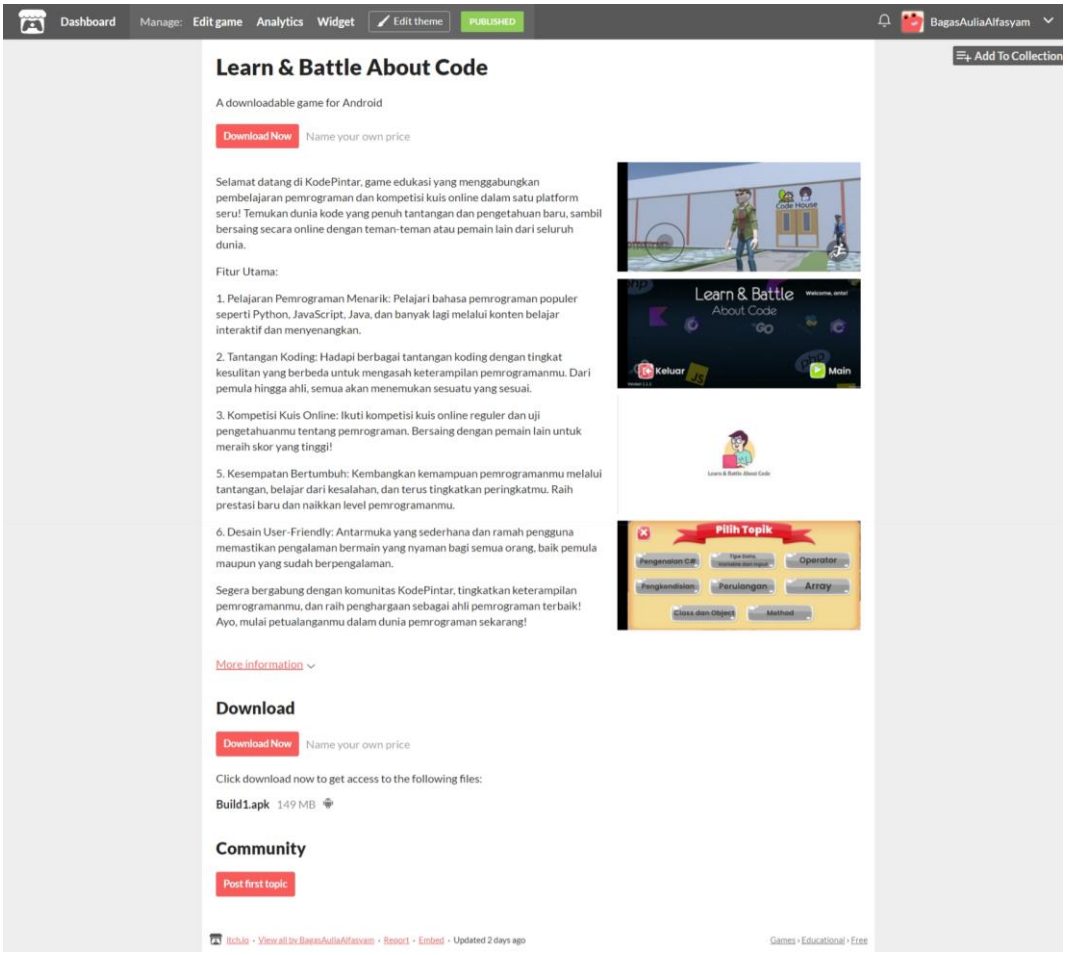
Skor rata-rata dihitung dari total skor pada SUS, kemudian hasilnya dibagi dengan jumlah total mahasiswa, menghasilkan skor rata-rata sekitar 72. Hasil ini diinterpretasikan menggunakan skala penilaian SUS, yang rentangnya antara 1 hingga 5. Dengan skor 72 ini, penilaian masuk dalam grade scale yang mencakup adjective rating "*good*" dan juga memenuhi kriteria "*acceptable*". Oleh karena itu, dapat disimpulkan bahwa game "Learn and Battle About Code" telah diterima dengan baik oleh mahasiswa yang memiliki ilmu pemograman.

4.3 Rilis

4.3.1 Perilisan *Itch.io*

Permainan "Learn and Battle About Code" telah diresmikan untuk diunduh melalui *website* yaitu *Itch.Io*. Pengguna dapat mengakses game ini melalui smartphone dengan membuka alamat *https://bagasauliaalfasyam.itch.io/learn-battle-about-code*. *Itch.io* yang dikenal sebagai salah satu platform distribusi digital terkemuka untuk game multi platform, menjadi wadah penyebaran utama. Tujuan dari ketersediaan permainan ini adalah memberikan manfaat yang signifikan bagi

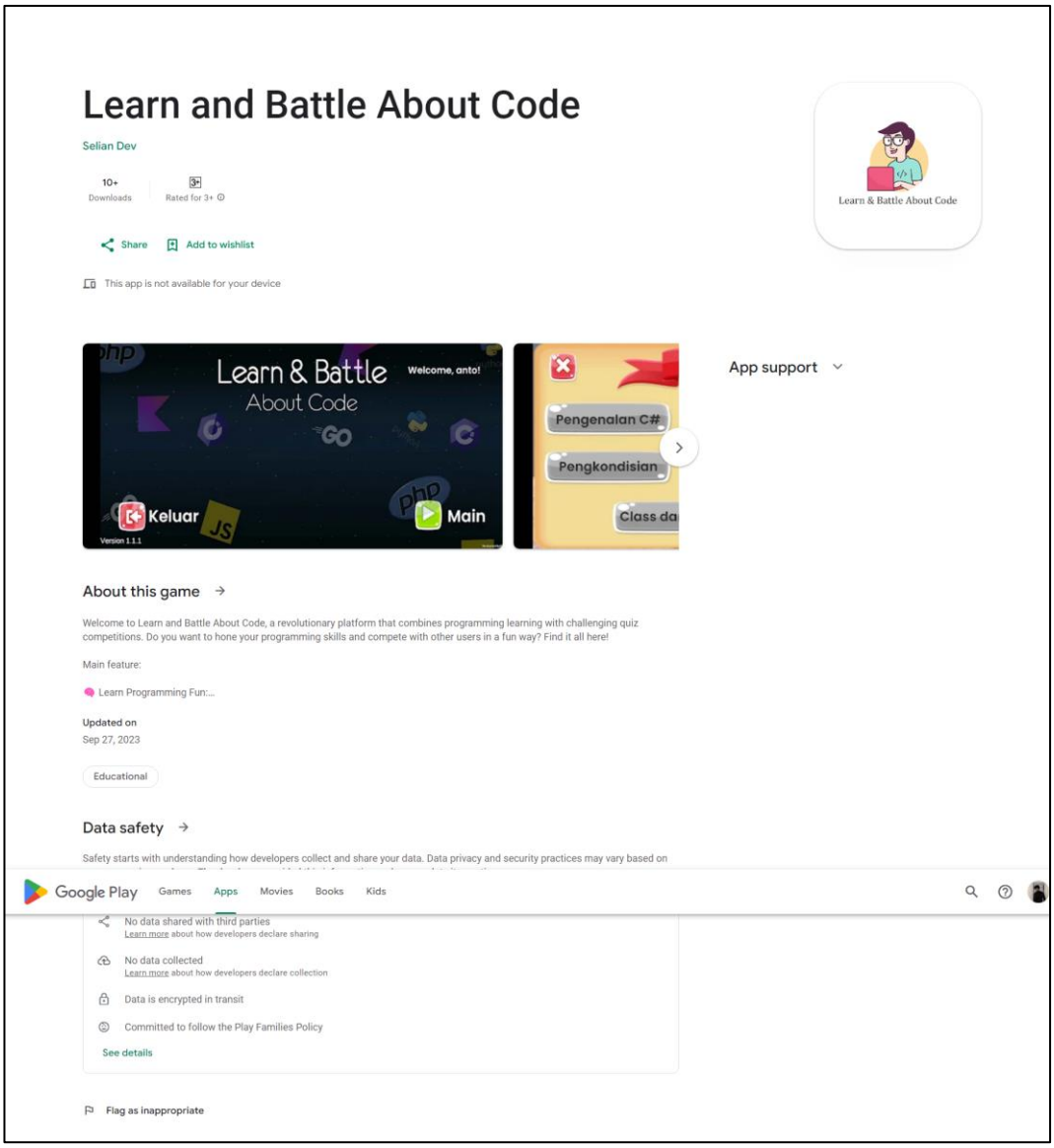
para programmer dan seluruh pengguna. Untuk melihat tampilan permainan secara lebih detail, tersedia Gambar 4.52 pada halaman resmi *Itch.Io*.



Gambar 4.49 Unduh *Learn and Battle About Code* di *itch io*

4.3.2 Perilisan *Google Play*

Permainan "*Learn and Battle About Code*" telah diresmikan untuk diunduh melalui *Google play*. Pengguna dapat mengakses game ini melalui smartphone dengan membuka alamat Learn and Battle About Code - Apps on Google Play. *Google Play* yang dikenal sebagai salah satu platform distribusi digital terkemuka untuk game multi platform, menjadi wadah penyebaran utama. Tujuan dari ketersediaan permainan ini adalah memberikan manfaat yang signifikan bagi para programmer dan seluruh pengguna. Untuk melihat tampilan permainan secara lebih detail, tersedia Gambar 4.53 pada halaman resmi *Google Play*.



Gambar 4.50 Unduh *Learn and Battle About Code* di *Google Play Store*

BAB V
KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil dari perancangan dan pengujian sistem, beberapa poin kesimpulan dapat diuraikan:

1. Game "*Learn and Battle About Code*" berhasil diciptakan melalui penerapan metode *Game Development Life Cycle* (GDLC), memastikan efisiensi program dan pengembangan.
2. Pengembangan game pembelajaran "*Learn and Battle About Code*" telah dilakukan dengan melalui 5 tahapan GDLC, yakni inisiasi, pra-produksi, produksi, pengujian (*alpha* dan *beta*), dan rilis. Proses pengembangan ini dilaksanakan menggunakan *Game Engine Unity 3D*.
3. Hasil pengujian black box dan pengujian perangkat menunjukkan bahwa game berjalan sesuai dengan ekspektasi yang diinginkan.
4. Dengan skor rata-rata 72 pada *System Usability Scale* (SUS), *game* "*Learn and Battle About Code*" dapat disimpulkan telah diterima dengan baik oleh mahasiswa yang menguasai pemrograman. Skor ini masuk dalam kategori "baik" dan memenuhi kriteria "*acceptable*" pada skala penilaian SUS.

5.2 Saran

Masih terdapat beberapa aspek yang perlu diperbaiki dalam penelitian ini, seperti yang teridentifikasi selama proses pengujian dan pengembangan. Rekomendasi perbaikan untuk penelitian mendatang meliputi:

1. Penyajian pengetahuan bahasa pemrograman perlu diperkaya dengan tampilan visual yang lebih baik.
2. Responsivitas resolusi *game* perlu ditingkatkan untuk memastikan tampilan optimal pada berbagai rasio resolusi perangkat selama pengujian.
3. Diperlukan perluasan *server game* agar dapat menangani jumlah pengguna yang lebih besar selama kompetisi kuis.

DAFTAR PUSTAKA

Agustina, C., & Wahyudi Program Studi Manajemen Informatika AMIK BSI Yogyakarta, T. (2015). Aplikasi Game Pendidikan Berbasis Android Untuk Memperkenalkan Pakaian Adat Indonesia. *IJSE – Indonesian Journal*, *1*(1), 2–3.

Andriyat Krisdiawan, R. (2019). PENERAPAN MODEL PENGEMBANGAN GAMEGDLC (GAME DEVELOPMENT LIFE CYCLE)DALAM MEMBANGUN GAME PLATFORM BERBASIS MOBILE. *TEKNOKOM*, *2*(1).

Aprianti, W., Maliha, U., Teknik Informatika, J., Negeri, P., Laut, T., Km, J. A. Y., Laut, P. T., & Selatan, K. (2016). SISTEM INFORMASI KEPADATAN PENDUDUK KELURAHAN ATAU DESA STUDI KASUS PADA KECAMATAN BATI-BATI KABUPATEN TANAH LAUT. In Jurnal Sains dan Informatika (Vol. 2, Issue 1).

Frialdo, D., Helmina, A., Oktaviani Melianti, E., Jalinus, N., Abdullah, R., Sarjana, P., Negeri Padang, U., Hamka, J., Tawar Bar, A., & Barat, S. (2023). Rancang Bangun Media Pembelajaran Sistem Operasi Jaringan Materi Instalasi Debian Berbasis Android. *Journal on Education*, *05*(02), 2003–2010.

Mulyono, K. M., & Al Fatta, H. (2012). PEMBUATAN GAME LABIRIN DENGAN MENGGUNAKAN BLENDER 3D. *JURNAL DASI*, *13*(2).

Najuah, N., Sidiq, R., Simamora, & R. S. (2022). Game Edukasi: Strategi dan Evaluasi Belajar Sesuai Abad 21. In Yayasan Kita Menulis.

Nugroho, A., & Pramono, B. A. (2017). *APLIKASI MOBILE AUGMENTED REALITY BERBASIS VUFORIA DAN UNITY PADA PENGENALAN OBJEK 3D DENGAN STUDI KASUS GEDUNG M UNIVERSITAS SEMARANG* (Vol. 14, Issue 2). Retrieved from www.unity3d.com.

Ramdhan, S., Tullah, R., Janah, & S. N. (2019). Iklan Animasi Stop Bullying Pada SD Negeri Cibadak II Berbasis Multimedia. *Jurnal Sisfotek Global*, *9*(2).

Rochmawati, & F.D. (2023). PERKEMBANGAN BAHASA PEMROGRAMAN KOMPUTER DI AMERIKA SERIKAT TAHUN 1955-1995. *Doctoral Dissertation, UIN Kiai Haji Achmad Siddiq Jember*.

Simatupang, J., Sianturi, & S. (2019). PERANCANGAN SISTEM INFORMASI PEMESANAN TIKET BUS PADA PO. HANDOYO BERBASIS ONLINE. *Jurnal Intra Tech*, *3*(2), 11–25.

LAMPIRAN

1. Game Settings.cs

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

public class GameSettings : MonoBehaviour {

    public static GameSettings instance;
    [Range(2,2)]
    public int MaxPlayers = 2;

    [Header("build index Game Scene")]
    [Tooltip("Jangan lupa ganti dengan build index Game Scene")]
    public int GameScene = 2;

    [Space(10)]
    [Header("Game Rules")]
    [Range(1,100)]
    public int AnswerTime = 10;
    public int AnswerPoints = 10;
    public int QuestionTime = 30;


    public void Awake()
    {
        if(instance == null)
        {
            instance = this;
        }
        else if(instance != this)
        {
            Destroy(gameObject);
        }
        DontDestroyOnLoad(gameObject);
    }
}
```

2. PhotonRoom.cs

```
A using Photon.Pun;
using Photon.Realtime;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using System;

public enum GameState
{
    waiting,
    started,
    ended
}
```

```
public class PhotonRoom : MonoBehaviourPunCallbacks
{
    public static PhotonRoom instance;

    public int NumOfWrongAnswerPlayer = 0;

    [SerializeField]
    private Hashtable players = new Hashtable();

    public int[] playerScore;// = new
int[GameSettings.instance.MaxPlayers];


    private GameState gameState;

    private int numberOfPlayers;

    [SerializeField]
    private PhotonView _photonView;


    public Hashtable Players
    {
        get
        {
            return players;
        }

        set
        {
            players = value;
        }
    }


    private void Awake()
    {

        if (instance == null)
        {
            instance = this;
        }
        else
        {
            if (instance != this)
                Destroy(gameObject);
        }




        gameState = GameState.waiting;
        DontDestroyOnLoad(this);

        playerScore = new int[2];

    }

    public override void OnEnable()
```

```
    {
        base.OnEnable();
        PhotonNetwork.AddCallbackTarget(this);
        SceneManager.sceneLoaded += OnSceneLoaded;

    }

    public override void OnDisable()
    {
        base.OnDisable();
        PhotonNetwork.RemoveCallbackTarget(this);
        SceneManager.sceneLoaded -= OnSceneLoaded;

    }



    public override void OnJoinedRoom()
    {
        base.OnJoinedRoom();

        numberOfPlayers = PhotonNetwork.PlayerList.Length;
        string name = PlayerPrefs.GetString("PlayerName");
        PhotonNetwork.NickName = name;

        CheckPlayer();

        _photonView.RPC("RPC_UpdateNumberPlayers", RpcTarget.All);


    }

    public override void OnPlayerEnteredRoom(Player newPlayer)
    {
        base.OnPlayerEnteredRoom(newPlayer);

        numberOfPlayers = PhotonNetwork.PlayerList.Length;




        if(numberOfPlayers == GameSettings.instance.MaxPlayers)
        {

        }

        CheckPlayer();
    }

    public override void OnConnectedToMaster()
    {
        Debug.Log("PUN Basics Tutorial/Launcher:
OnConnectedToMaster() was called by PUN");
    }


    public override void OnDisconnected(DisconnectCause cause)
    {
        Debug.LogWarningFormat("PUN Basics Tutorial/Launcher:
OnDisconnected() was called by PUN with reason {0}", cause);
```

```
        if(GameUIController.instance != null)

GameUIController.instance.ShowResult((int)GameUIController.ResultType
.Disconnected);
    }



    private void CheckPlayer()
    {


        if (!PhotonNetwork.IsMasterClient)
            return;

        if (numberOfPlayers == GameSettings.instance.MaxPlayers)
        {
            StartGame();
        }
    }


    private void StartGame()
    {
        gameState = GameState.started;
        PhotonNetwork.CurrentRoom.IsOpen = false;
        SceneManager.LoadScene(GameSettings.instance.GameScene);


        _photonView.RPC("LoadPlayers", RpcTarget.All);


    }


    private bool PlayerCreated = false;


    private void OnSceneLoaded(Scene scene, LoadSceneMode mode)
    {
        if (scene.buildIndex == GameSettings.instance.GameScene)
        {
            if( ! PlayerCreated)
            {
                RPC_CreatePlayer();
                PlayerCreated = true;
            }

        }
    }

    [PunRPC]
    public void RPC_UpdateScore(string PlayerName, int Points)
    {
        int _score = 0;
        foreach (DictionaryEntry entry in players)
        {
            if (entry.Key as string == PlayerName)
```

```
            {
                Debug.Log(entry.Key);
                _score = (int)entry.Value + Points;
                players[entry.Key] = _score;


                break;
            }
        }

        foreach (DictionaryEntry entry in players)
        {


GameUIController.instance.UpdateScore(entry.Key.ToString(),
(int)entry.Value);


        }
    }

    [PunRPC]
    public void StopTimerAnswerAllPlayer()
    {
        Player[] _players = PhotonNetwork.PlayerList;
        for (int i = 0; i < _players.Length; i++)
        {


            GameUIController.instance.StopAnswerTimer();

GameUIController.instance.PlayerBellNotif[i].SetActive(false);

        }



    }

    [PunRPC]
    public void RingBellNotifOnPlayer(string PlayerName)
    {
        Player[] _players = PhotonNetwork.PlayerList;
        for (int i = 0; i < _players.Length; i++)
        {
            if (_players[i].NickName == PlayerName)
            {


                GameUIController.instance.BellNotifOn(PlayerName);


            }
        }


    }

    [PunRPC]
    public void setHurryUpToAll(bool newVal)
```

```
    {
        GameUIController.instance.HurryUp = newVal;
    }


    public void AddScore(string PlayerName, int Points)
    {

        Player[] _players = PhotonNetwork.PlayerList;

        for (int i = 0; i < _players.Length; i++)
        {
           if (_players[i].NickName  == PlayerName)
            {

                playerScore[i] += Points;
                GameUIController.instance.UpdateScore(PlayerName,
playerScore[i]);

            }
        }

    [PunRPC]
    public void RPC_CreatePlayer()
    {
        GameObject obj =
PhotonNetwork.Instantiate("prefabs/PlayerObject", Vector3.zero,
Quaternion.identity);
    }

    [PunRPC]
    public void RPC_UpdateNumberPlayers()
    {
        if (SceneManager.GetActiveScene().buildIndex !=
GameSettings.instance.GameScene)
            UIController.instance.ShowMessage("Menunggu pemain\n" +
numberOfPlayers + "/" + GameSettings.instance.MaxPlayers);
    }

    [PunRPC]
    public void LoadPlayers()
    {
        Player[] _players = PhotonNetwork.PlayerList;

        for (int i = 0; i < _players.Length; i++)
        {
            Debug.Log("added player: " + _players[i].NickName);
            players.Add(_players[i].NickName, 0);
        }

    }

    public void LeaveRoom()
    {
```

```
        PhotonNetwork.LeaveRoom();

GameUIController.instance.ShowResult((int)GameUIController.ResultType
.PlayerLeave, "", PlayerPrefs.GetString("PlayerName"));

    }

    override
    public void OnPlayerLeftRoom(Player otherPlayer)
    {
        Debug.Log("PhotonPlayer left the room : " +
otherPlayer.NickName);

        if( QuestionController.instance.QuestionIndex <
QuestionController.instance._questions.Count )
            GameUIController.instance.ShowResult( (int)
GameUIController.ResultType.PlayerLeave, "", otherPlayer.NickName);
        else if (QuestionController.instance.QuestionIndex >=
QuestionController.instance._questions.Count)

GameUIController.instance.ShowResult((int)GameUIController.ResultType
.AllAnswered);

    }



    public  void OnPhotonPlayerConnected(Player newPlayer)
    {
        Debug.Log("PhotonPlayer entered the room : " +
newPlayer.UserId);
    }

    public  void OnDisconnectedFromPhoton()
    {
        Debug.Log("Disconnected from Photon");
        PhotonNetwork.ReconnectAndRejoin();
    }

    public void OnPhotonPlayerDisconnected(Player otherPlayer)
    {
        Debug.Log("OnPhotonPlayerDisconnected   : " +
otherPlayer.NickName);

    }
}
```

3. Question.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Perangonline.PhotonQuis
{
    public class Question
    {
```

```
        private int _idQ;
        private string _description;
        private bool _isImage;


        public Question(int idQ, string description, bool isImage)
        {
            IdQ = idQ;
            Description = description;
            IsImage = isImage;
        }


        public string Description
        {
            get
            {
                return _description;
            }

            set
            {
                _description = value;
            }
        }

        public int IdQ
        {
            get
            {
                return _idQ;
            }

            set
            {
                _idQ = value;
            }
        }

        public bool IsImage
        {
            get
            {
                return _isImage;
            }

            set
            {
                _isImage = value;
            }
        }
    }
}
```

4. PhotonLobby.cs

```
using System.Collections;
using Photon.Pun;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using Photon.Realtime;

public class PhotonLobby: MonoBehaviourPunCallbacks {

    public GameObject BeginButton;
    public GameObject CancelButton;
    public GameObject LoadingPanel;
    public InputField PlayerName;

       void Start () {

        PhotonNetwork.ConnectUsingSettings();

        PhotonNetwork.AutomaticallySyncScene = true;
        UIController.instance.SetLoading(true);
        UIController.instance.ShowMessage("Memuat..");

    }


    public override void OnConnected()
    {
        base.OnConnected();

        BeginButton.SetActive(true);
        PlayerName.gameObject.SetActive(true);
        UIController.instance.HideMessage();
        UIController.instance.SetLoading(false);
        string _playerName = PlayerPrefs.GetString("PlayerName");

        if (_playerName != string.Empty)
        {
            PlayerName.text = _playerName;
        }
    }


    public void OnBeginClick()
    {
        BeginButton.SetActive(false);
        CancelButton.SetActive(true);
        PlayerName.gameObject.SetActive(false);
        PlayerPrefs.SetString("PlayerName", PlayerName.text);

        UIController.instance.SetLoading(true);

        PhotonNetwork.JoinRandomRoom();

    }
    public void OnCancelClick()
    {
        BeginButton.SetActive(true);
```

```
        CancelButton.SetActive(false);
        PhotonNetwork.LeaveRoom();
        UIController.instance.Loading.SetActive(false);
        UIController.instance.HideMessage();
    }

    public void SetLoading(bool active, String msg)
    {
        LoadingPanel.SetActive(active);
    }

    public override void OnJoinRandomFailed(short returnCode, string
message)
    {
        CreateRoom();
    }

    public override void OnCreateRoomFailed(short returnCode, string
message)
    {
        CreateRoom();
    }

    private void CreateRoom()
    {
        RoomOptions options = new RoomOptions() { IsOpen = true,
IsVisible = true, MaxPlayers = (byte)GameSettings.instance.MaxPlayers
};
        PhotonNetwork.CreateRoom("Room " +
UnityEngine.Random.Range(1, 1000) ,options);

    }



}
```

5. PlayerRPCHandler.cs

```
using Photon.Pun;
using Photon.Realtime;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UnityEngine;
using ExitGames.Client.Photon;

namespace Perangonline.PhotonQuis.Controlelrs
{
    public class PlayerRPCHandler : MonoBehaviour, IOnEventCallback
    {

        public float StartDelay = 1.5f;
        [SerializeField]
```

```
        private PhotonView _photonView;

        private GPlayer _player;
        private bool _canAnswer = false;
        private bool _answered = false;


        private readonly byte RequestAnswer = 0;
        private readonly byte RightAnswer = 1;
        private readonly byte WrongAnswer = 2;

        private bool masterNext = false;

        public bool CanAnswer
        {
            get
            {
                return _canAnswer;
            }

            set
            {
                _canAnswer = value;
            }
        }

        private void Start()
        {


            _player = GetComponent<GPlayer>();

            if (_photonView.IsMine)
                _photonView.RPC("SetPlayer", RpcTarget.All,
PlayerPrefs.GetString("PlayerName"));

            if (PhotonNetwork.IsMasterClient)
            {
                PreparePlayers();
                _photonView.RPC("PreparePlayers", RpcTarget.Others);
            }

        }


        public void OnEnable()
        {
            PhotonNetwork.AddCallbackTarget(this);
        }


        public void OnDisable()
        {
            PhotonNetwork.RemoveCallbackTarget(this);
        }

        [PunRPC]
        public void SetPlayer(string name)
        {
            if (_player == null)
```

```
                _player = GetComponent<GPlayer>();

            _player.PlayerName = name;
        }



        [PunRPC]
        public void PreparePlayers()
        {

GameUIController.instance.PreparePlayers(PhotonNetwork.PlayerList);


            QuestionController.instance.LoadQuestions();







            StartCoroutine(WaitStart(StartDelay));
        }



        private IEnumerator WaitStart(float delay)
        {
            yield return new WaitForSeconds(delay);

            QuestionController.instance.questionIndex = 0;
            QuestionController.instance.Next(0);

            if (PhotonNetwork.IsMasterClient)
                QuestionController.instance.Next(0);


            CanAnswer = true;
        }


        IEnumerator hideWrongAnswerNotif()
        {
            yield return new WaitForSeconds(2);

GameUIController.instance.WrongAnswerNotif.SetActive(false);
        }


        public void OnEvent(EventData photonEvent)
        {

            Debug.Log("OnEvent code: " + photonEvent.Code);

            if (!_photonView.IsMine)
                return;

            if (photonEvent.Code > 3)
                return;
```

```
            string playerName = ((object[])photonEvent.CustomData)[0]
as string;

            switch (photonEvent.Code)
            {
                case 0:

                    Player[] players = PhotonNetwork.PlayerList;

                    for (int i = 0; i < players.Length; i++)
                    {
                        if (playerName == PhotonNetwork.NickName)
                        {
                            if (CanAnswer)
                            {


//PhotonRoom.instance.RingBellNotifOnPlayer(playerName);
                                //StopTimerAnswerAllPlayer
                                PhotonRoom.instance.photonView.RPC
("RingBellNotifOnPlayer",  RpcTarget.All, playerName);


GameUIController.instance.EnableAnswers(true);


GameUIController.instance.StartAnswerTimer();
                                _answered = true;
                                return;

                            }
                        }
                    }
                    CanAnswer = false;
                    GameUIController.instance.EnableBuzzer(false);


                    break;
                case 1:

                    PhotonRoom.instance.AddScore(playerName,
GameSettings.instance.AnswerPoints);
                    QuestionController.instance.QuestionIndex += 1;

                    //int index = PhotonNetwork.IsMasterClient ?
QuestionController.instance.QuestionIndex - 1 :
QuestionController.instance.QuestionIndex;
                    //QuestionController.instance.Next(index);


QuestionController.instance.Next(QuestionController.instance.Question
Index);



                    CanAnswer = true;
                    PhotonRoom.instance.  NumOfWrongAnswerPlayer = 0;

                    _answered = false;
```

```
//GameUIController.instance.WrongAnswerNotif.SetActive(false);

                    StartCoroutine("hideWrongAnswerNotif");

                    for (int i = 0; i<
GameUIController.instance.PlayerBellNotif.Count; i++ )

GameUIController.instance.PlayerBellNotif[i].SetActive(false);

                    break;
                case 2: //wrong answer


                    GameUIController.instance.StopAnswerTimer();
                    //StopTimerAnswerAllPlayer

PhotonRoom.instance.photonView.RPC("StopTimerAnswerAllPlayer",
RpcTarget.All);



                    PhotonRoom.instance.NumOfWrongAnswerPlayer++;

                    if (PhotonRoom.instance.NumOfWrongAnswerPlayer <
2)
                    {
                        if (_answered)
                        {

GameUIController.instance.WrongAnswerNotif.SetActive(true);

GameUIController.instance.ActivateAnswers(false);



                            return;
                        }


                        GameUIController.instance.EnableBuzzer(true);

                        CanAnswer = true;


                    }
                    else
                    {

GameUIController.instance.WrongAnswerNotif.SetActive(true);

                        QuestionController.instance.QuestionIndex +=
1;


QuestionController.instance.Next(QuestionController.instance.Question
Index);
```

```
                        CanAnswer = true;

                        _answered = false;

                        PhotonRoom.instance.NumOfWrongAnswerPlayer =
0;

                        GameUIController.instance.HurryUp = false;


//GameUIController.instance.WrongAnswerNotif.SetActive(false);
                        StartCoroutine("hideWrongAnswerNotif");

                        for (int i = 0; i <
GameUIController.instance.PlayerBellNotif.Count; i++)

GameUIController.instance.PlayerBellNotif[i].SetActive(false);

                    }
                    break;

                case 3: //timeout
                    object[] data = (object[])photonEvent.CustomData;
                    int x = (int)data[0];

                    QuestionController.instance.QuestionIndex = x;

                    QuestionController.instance.QuestionIndex += 1;


QuestionController.instance.Next(QuestionController.instance.Question
Index);

                    CanAnswer = true;

                    _answered = false;

                    PhotonRoom.instance.NumOfWrongAnswerPlayer = 0;


                    GameUIController.instance.HurryUp = false;


//GameUIController.instance.WrongAnswerNotif.SetActive(false);
                    StartCoroutine("hideWrongAnswerNotif");

                    for (int i = 0; i <
GameUIController.instance.PlayerBellNotif.Count; i++)

GameUIController.instance.PlayerBellNotif[i].SetActive(false);
                    break;

            }
        }


    }
}
```

6. Dialog Manager.cs

```
using System.Collections;
using Photon.Pun;
using Photon.Realtime;
using UnityEngine;
using TMPro;
using UnityEngine.UI;
using System.Collections.Generic;

public class DialogManager : MonoBehaviourPunCallbacks
{
    public GameObject panelEnterQuiz;
    public GameObject dialogPanel;
    public TMP_Text npcNameText;
    public Image npcImage;
    public TMP_Text dialogText;
    public Button continueButton;

    private int roomCounter = 1;

    private string[] dialogLines;
    private int currentLine = 0;
    private bool isTyping = false;
    private string currentSentence = "";
    private Coroutine typingCoroutine;
    private bool isDisplayingText = false;


    private static DialogManager instance;
    private NPCController currentNPC;

    public static DialogManager Instance
    {
        get
        {
            if (instance == null)
            {
                instance = FindObjectOfType<DialogManager>();
            }
            return instance;
        }
    }

    private void Start()
    {
        dialogPanel.SetActive(false);
    }

    public void SetCurrentNPC(NPCController npc)
    {
        currentNPC = npc;
    }

    public void ShowDialog(string npcName, Sprite image, string[]
lines)
    {
        npcNameText.text = npcName;
        npcImage.sprite = image;
        dialogLines = lines;
```

```
        currentLine = 0;
        DisplayNextLine();
        dialogPanel.SetActive(true);
    }

    public void DisplayNextLine()
    {
        if (currentLine < dialogLines.Length && !isDisplayingText)
        {
            currentSentence = dialogLines[currentLine];
            currentLine++;
            StartCoroutine(TypeSentence(currentSentence));
        }
        else if (currentLine >= dialogLines.Length)
        {
            EndDialog();
        }
    }

    private IEnumerator TypeSentence(string sentence)
    {
        isDisplayingText = true;  // Mulai menampilkan teks
        dialogText.text = "";

        for (int i = 0; i < sentence.Length; i++)
        {
            dialogText.text += sentence[i];
            yield return new WaitForSeconds(0.05f);
        }

        continueButton.gameObject.SetActive(true);
        isDisplayingText = false;  // Selesai menampilkan teks
    }



    private void EndDialog()
    {
        dialogText.text = "";
        continueButton.gameObject.SetActive(false);
        currentNPC = null;
        dialogPanel.SetActive(false);
        panelEnterQuiz.SetActive(true);
    }

    public void ContinueDialog()
    {
        Debug.Log("Continue button pressed.");
        if (isTyping)
        {
            StopCoroutine(typingCoroutine);
            dialogText.text = currentSentence;  // Tampilkan teks
penuh
            isTyping = false;
        }
        else
        {
            DisplayNextLine();
        }
    }
```

```
    public void JoinQuizForNPC()
    {
        int nextRoomCounter = GetNextRoomCounter();
        panelEnterQuiz.SetActive(false);
        UIController.instance.SetLoading(true);
        UIController.instance.OnPressedCancel(true);
        string nameRoom = currentNPC.npcName;
        RoomOptions roomOptions = new RoomOptions();
        roomOptions.MaxPlayers = 2;
        PhotonNetwork.JoinOrCreateRoom(nameRoom, roomOptions,
TypedLobby.Default);
        //Debug.Log(nameRoom);
    }

    int GetNextRoomCounter()
    {
        int nextRoomCounter = roomCounter;
        roomCounter++;
        return nextRoomCounter;
    }

    public override void OnJoinRoomFailed(short returnCode, string
message)
    {
        base.OnJoinRoomFailed(returnCode, message);
        //Debug.LogError("Gagal Bergabung Ke Ruangan: " + message);
        JoinQuizForNPC();
    }

    public override void OnCreateRoomFailed(short returnCode, string
message)
    {
        base.OnCreateRoomFailed(returnCode, message);
        //Debug.LogError("Gagal Membuat Ruangan: " + message);
        JoinQuizForNPC();
    }
}
```

7. NPCController.cs

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;

public class NPCController : MonoBehaviour
{
    public static NPCController instance;

    private void Awake()
    {

        if (instance == null)
        {
            instance = this;
        }
```

```
    }

    public string npcName;
    public Sprite npcImage;
    public string[] dialogLines;

    private float chatDistance = 2f;
    private bool playerInRange = false;
    private GameObject player;
    public Button chatButton;

    private void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        chatButton.onClick.AddListener(StartDialog);
        chatButton.gameObject.SetActive(false);
    }

    private void Update()
    {
        float distanceToPlayer = Vector3.Distance(transform.position,
player.transform.position);
        playerInRange = distanceToPlayer < chatDistance;

        if (playerInRange)
        {
            chatButton.gameObject.SetActive(true);
            DialogManager.Instance.SetCurrentNPC(this);
        }
        else
        {
            chatButton.gameObject.SetActive(false);
        }
    }

    private void StartDialog()
    {
        DialogManager.Instance.ShowDialog(npcName, npcImage,
dialogLines);
    }
}
```

8. ManagerInit.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.EventSystems;
using TMPro;
using UnityEngine.UI;
using UnityEngine;
using UnityEngine.SceneManagement;
using Photon.Pun;

public class ManagerInit : MonoBehaviour
{
    [Header("UI")]
    [SerializeField] GameObject mainMenu;
```

```
    [SerializeField] GameObject panelLogout;
    [SerializeField] GameObject panelTentang;
    [SerializeField] GameObject panelIsiNama; // Panel untuk isi nama
    [SerializeField] TMP_InputField inputPlayerName;
    [SerializeField] TMP_Text userName;

    [Header("Button")]
    [SerializeField] Button buttonPlay;
    [SerializeField] Button buttonLogout;
    [SerializeField] Button buttonInfo;
    [SerializeField] Button buttonCloseTentang;

    public int SceneBuildIndexToLoad;

    private void Start()
    {
        EventTrigger trigger =
userName.gameObject.AddComponent<EventTrigger>();
        EventTrigger.Entry entry = new EventTrigger.Entry { eventID =
EventTriggerType.PointerClick };
        entry.callback.AddListener((eventData) => {
ShowPanelIsiNama(); });
        trigger.triggers.Add(entry);
        // Cek apakah nama sudah tersimpan di PlayerPrefs
        if (PlayerPrefs.HasKey("Username"))
        {
            // Jika ada, tampilkan nama
            string savedName = PlayerPrefs.GetString("Username");
            userName.text = savedName;
        }
    }

    private void ShowPanelIsiNama()
    {
        panelIsiNama.SetActive(true);
    }

    public void OnPressedPlay()
    {
        if (string.IsNullOrEmpty(userName.text))
        {
            // Nama belum diisi, tampilkan panel untuk mengisi nama
            panelIsiNama.SetActive(true);
        }
        else
        {
            // Nama sudah diisi, lanjut ke permainan
            StartGame();
        }
    }


    private void StartGame()
    {
        StartCoroutine("loadScene");
    }


    IEnumerator loadScene()
    {
```

```
        yield return new WaitForSeconds(0.1f);
        PhotonNetwork.LeaveRoom();
        PhotonNetwork.Disconnect();
        SceneManager.LoadScene(SceneBuildIndexToLoad);
        if (PhotonRoom.instance.gameObject != null)
        {
            Destroy(PhotonRoom.instance.gameObject);
        }
    }

    public void SetNama()
    {
        panelIsiNama.SetActive(false);
        string newName = inputPlayerName.text;
        userName.text = newName;  // Tampilkan nama di UI
        PlayerPrefs.SetString("Username", newName);
    }
    public void OnpressedLogout()
    {
        panelLogout.SetActive(true);
        mainMenu.SetActive(false);
        buttonLogout.gameObject.SetActive(false);
        buttonInfo.gameObject.SetActive(false);
        buttonPlay.gameObject.SetActive(false);
    }

    public void OnCancelLogout()
    {
        panelLogout.SetActive(false);
        mainMenu.SetActive(true);
        buttonPlay.gameObject.SetActive(true);
        buttonLogout.gameObject.SetActive(true);
        buttonInfo.gameObject.SetActive(true);
    }

    public void OnCancelInfo()
    {
        panelTentang.SetActive(false);
        mainMenu.SetActive(true);
        buttonPlay.gameObject.SetActive(true);
        buttonLogout.gameObject.SetActive(true);
        buttonInfo.gameObject.SetActive(true);
    }

    public void OnpressedInfo()
    {
        panelTentang.SetActive(true);
        mainMenu.SetActive(false);
        buttonPlay.gameObject.SetActive(false);
        buttonLogout.gameObject.SetActive(false);
        buttonInfo.gameObject.SetActive(false);
    }

    public void OnExitGame()
    {
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#else
        Application.Quit();
#endif
```

```
    }
}
```

9. GameUIManager.cs

```
using BagasDev.PhotonQuis;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using Photon.Realtime;
using ExitGames.Client.Photon;
using Photon.Pun;

public class GameUIController : MonoBehaviour {

    public static GameUIController instance;

    [SerializeField]
    public List<GameObject> PlayerBellNotif;

    [SerializeField]
    public GameObject WrongAnswerNotif;
    public GameObject CorrectAnswerNotif;


    [SerializeField]
    public GameObject PanelResult;

    public Button ButtonExitRoom;

    [SerializeField]
    private Text QuestionText;
    [SerializeField]
    private Button Answer1;
    [SerializeField]
    private Button Answer2;
    [SerializeField]
    private Button Answer3;



    [SerializeField]
    private Button Answer4;

    [SerializeField]
    private Image player1;
    [SerializeField]
    private Image player2;
    [SerializeField]
    private Image player3;

    [SerializeField]
    private Button BuzzerButton;
    [SerializeField]
    private Text Timer;
```

```
    [SerializeField]
    private Text QuestionTime;

    private Text scorePlayer1;
    private Text scorePlayer2;
    private Text scorePlayer3;


    private readonly byte RequestAnswer = 0;
    private readonly byte RightAnswer = 1;
    private readonly byte WrongAnswer = 2;

private void Awake()
    {
        instance = this;
    }

    private void Start()
    {
        scorePlayer1 =
player1.transform.GetChild(1).GetComponent<Text>();
        scorePlayer2 =
player2.transform.GetChild(1).GetComponent<Text>();
        //scorePlayer3 =
player3.transform.GetChild(1).GetComponent<Text>();

        EnableAnswers(false);
        EnableBuzzer(false);


        ButtonExitRoom.onClick.AddListener(() => {
            PhotonRoom.instance.LeaveRoom();
        });
    }

    public void EnableAnswers(bool enabled)
    {
        Answer1.enabled = enabled;
        Answer2.enabled = enabled;
        Answer3.enabled = enabled;
        Answer4.enabled = enabled;


    }

    public void ActivateAnswers(bool active)
    {
        Answer1.gameObject.SetActive(active);
        Answer2.gameObject.SetActive(active);
        Answer3.gameObject.SetActive(active);
        Answer4.gameObject.SetActive(active);

        if (!active)
        {
            QuestionText.gameObject.SetActive(false);
        }
    }
```

```
    public enum  ResultType{   AllAnswered, Timeup, PlayerLeave,
Disconnected };
    private bool isDraw = true;

    [HideInInspector]
    public bool ResultIsShown = false;

    public void ShowResult(int resulttype = 0, string WinnerNickname =
"", string runningAwayNickname = "")
    {
        if(!ResultIsShown)
        {
            PanelResult ps = PanelResult.GetComponent<PanelResult>();
            int BestScore = 0;

            if (resulttype == (int)ResultType.AllAnswered)
            {
                ps.ResultDesc.text = "Selesai!";

                if (PhotonRoom.instance.playerScore[0] ==
PhotonRoom.instance.playerScore[1])
                {
                    isDraw = true;
                }
                else
                {
                    Player[] _players = PhotonNetwork.PlayerList;

                    BestScore =
Math.Max(PhotonRoom.instance.playerScore[0],
PhotonRoom.instance.playerScore[1]);
                    int winner = -1;

                    for (int i = 0; i <
PhotonRoom.instance.playerScore.Length; i++)
                    {
                        if (BestScore ==
PhotonRoom.instance.playerScore[i])
                        {
                            winner = i;
                            break;
                        }
                    }

                    if (winner != -1)
                    {
                        isDraw = false;
                        WinnerNickname = _players[winner].NickName;
                    }
                }


                if (isDraw)
                {
                    ps.ResultDesc.text += "\r\n" + "\r\n" + "Seri!";
                }
                else
                {
                    if (WinnerNickname ==
PlayerPrefs.GetString("Username"))
```

```
                        ps.ResultDesc.text += "\r\n" + "\r\n" + "Kamu
Menang!";
                    else
                        ps.ResultDesc.text += "\r\n" + "\r\n" + "Kamu
Kalah!";

                    ps.ResultDesc.text += "\r\n" + "\r\n" + "Skor
Terbaik: " + BestScore + ". Pemenang: " + WinnerNickname;
                }
            }
            else if (resulttype == (int)ResultType.Timeup ||
resulttype == (int)ResultType.Disconnected)
            {
                if(resulttype == (int)ResultType.Timeup)
                    ps.ResultDesc.text = "Waktu Berakhir!";
                else
                    ps.ResultDesc.text = "Koneksi Terputus!";

                if (PhotonRoom.instance.playerScore[0] ==
PhotonRoom.instance.playerScore[1])
                {
                    isDraw = true;
                }
                else
                {
                    Player[] _players = PhotonNetwork.PlayerList;

                    BestScore =
Math.Max(PhotonRoom.instance.playerScore[0],
PhotonRoom.instance.playerScore[1]);
                    int winner = -1;

                    for (int i = 0; i <
PhotonRoom.instance.playerScore.Length; i++)
                    {
                        if (BestScore ==
PhotonRoom.instance.playerScore[i])
                        {
                            winner = i;
                            break;
                        }
                    }

                    if (winner != -1)
                    {
                        isDraw = false;
                        WinnerNickname = _players[winner].NickName;
                    }
                }
                if (isDraw)
                {
                    ps.ResultDesc.text += "\r\n" + "\r\n" + "Seri!";
                }
                else
                {
                    if (WinnerNickname ==
PlayerPrefs.GetString("Username"))
                        ps.ResultDesc.text += "\r\n" + "\r\n" + "Kamu
menang!";
                    else
```

```
                        ps.ResultDesc.text += "\r\n" + "\r\n" + "Kamu
Kalah";

                    ps.ResultDesc.text += "\r\n" + "\r\n" + "Skor
Terbaik: " + BestScore + ". Pemenangnya: " + WinnerNickname;
                }
            }
            else if (resulttype == (int)ResultType.PlayerLeave)
            {
                ps.ResultDesc.text = "Pemain Meninggalkan Kompetisi!";


                if (runningAwayNickname ==
PlayerPrefs.GetString("Username"))
                {
                    ps.ResultDesc.text += "\r\n" + "\r\n" + "Kamu
Kalah!";
                }
                else
                {
                    ps.ResultDesc.text += "\r\n" + "\r\n" + "Kamu
Menang!";
                }
            }
            PanelResult.SetActive(true);
            ResultIsShown = true;
        }
    }

    public void ShowQuestion(Question question, List<Answer> answers)
    {
        StopAllCoroutines();
        StartCoroutine("QuestionTimer");

        if (question.IsImage)
        {
            QuestionText.gameObject.SetActive(false);
            //StartCoroutine(LoadImage(QuestionImage,
question.Description));
        }
        else
        {
            QuestionText.gameObject.SetActive(true);
            QuestionText.text = question.Description;
        }

        ActivateAnswers(true);

        Answer1.transform.GetChild(0).GetComponent<Text>().text =
answers[0].Description;
        Answer1.tag = answers[0].IsTrue ? "true_answer" : "Untagged";

        Answer2.transform.GetChild(0).GetComponent<Text>().text =
answers[1].Description;
        Answer2.tag = answers[1].IsTrue ? "true_answer" : "Untagged";

        Answer3.transform.GetChild(0).GetComponent<Text>().text =
answers[2].Description;
        Answer3.tag = answers[2].IsTrue ? "true_answer" : "Untagged";
```

```
        Answer4.transform.GetChild(0).GetComponent<Text>().text =
answers[3].Description;
        Answer4.tag = answers[3].IsTrue ? "true_answer" : "Untagged";

    }

/*    public static IEnumerator LoadImage(Image image, string url)
    {
        WWW www = new WWW(url);

        yield return www;

        www.LoadImageIntoTexture(image.mainTexture as Texture2D);
        www = null;


    }*/


    public void BellNotifOn(string playerName)
    {
        int playerNumber = GetPlayerNumber(playerName);
        //Debug.Log("number: " + playerNumber + " name: " + playerName
+ " ring bell: ");
        switch (playerNumber)
        {
            case 1:
                PlayerBellNotif[0].SetActive(true);
                break;
            case 2:
                PlayerBellNotif[1].SetActive(true);

                break;


        }
    }


    public void UpdateScore(string playerName, int score)
    {
        int playerNumber = GetPlayerNumber(playerName);

        //Debug.Log("number: " + playerNumber + " name: " + playerName
+ " score: " + score );

        switch (playerNumber)
        {
            case 1:
                scorePlayer1.text = score.ToString();
                break;
            case 2:
                scorePlayer2.text = score.ToString();

                break;

                /*
            case 3:
                scorePlayer3.text = score.ToString();
                break;
```

```
                */
        }
    }

    public void PreparePlayers(Player[] players)
    {
        player1.transform.GetChild(0).GetComponent<Text>().text =
players[0].NickName;
        player2.transform.GetChild(0).GetComponent<Text>().text =
players[1].NickName;
        //player3.transform.GetChild(0).GetComponent<Text>().text =
players[2].NickName;
    }

    public void OnBuzzerClick()
    {

        EnableBuzzer(false);


        object[] content = new object[] {
PlayerPrefs.GetString("Username") };
        RaiseEventOptions raiseEventOptions = new RaiseEventOptions {
Receivers = ReceiverGroup.All};
        SendOptions sendOptions = new SendOptions { Reliability = true
};
        PhotonNetwork.RaiseEvent(RequestAnswer, content,
raiseEventOptions, sendOptions);

    }

    public void EnableBuzzer(bool enabled)
    {
        BuzzerButton.enabled = enabled;
    }

    public int GetPlayerNumber(string playerName)
    {
        if (player1.transform.GetChild(0).GetComponent<Text>().text ==
playerName)
            return 1;
        if (player2.transform.GetChild(0).GetComponent<Text>().text ==
playerName)
            return 2;

        return 3;
    }

    public void OnAnswerClick(Button button)
    {

        EnableAnswers(false);
        StopCoroutine(AnswerTimer());
        Timer.text = "";
        Timer.gameObject.SetActive(false);

        RaiseAnswerEvent(button.tag == "true_answer" ? RightAnswer :
WrongAnswer);
```

```
    }

    public void RaiseAnswerEvent(byte code)
    {
        object[] content = new object[] {
PlayerPrefs.GetString("Username") };
        RaiseEventOptions raiseEventOptions = new RaiseEventOptions {
Receivers = ReceiverGroup.All };
        SendOptions sendOptions = new SendOptions { Reliability = true
};

        PhotonNetwork.RaiseEvent(code, content, raiseEventOptions,
sendOptions);
    }

    private bool AnswerTimeStarted = false;


    [HideInInspector]
    public bool HurryUp;// true if answertime remaining is less than
QuestionTImeRemainingtime

    public void StartAnswerTimer()
    {
        AnswerTimeStarted = true;
        Timer.gameObject.SetActive(true);
        HurryUp = false;

        if (GameSettings.instance.AnswerTime <
QuestionTImeRemainingtime)
            AnswerTimeRemaining = GameSettings.instance.AnswerTime;
        else
        {
            HurryUp = true;

            //PhotonRoom.instance.photonView.RPC("setHurryUpToAll",
RpcTarget.All, true);


            AnswerTimeRemaining = QuestionTImeRemainingtime;
        }



        StartCoroutine("AnswerTimer");
    }

    public void StopAnswerTimer()
    {
        AnswerTimeStarted = false;
        Timer.gameObject.SetActive(false);




        StopCoroutine("AnswerTimer");
    }

    float AnswerTimeRemaining = 0f;
```

```
    private IEnumerator AnswerTimer()
    {
        if(AnswerTimeStarted)
        {
            do
            {
                AnswerTimeRemaining -= Time.deltaTime;
                Timer.text = Math.Round(AnswerTimeRemaining,
1).ToString();
                //Timer.text = "" + timeRemaining;
                yield return null;

            } while (AnswerTimeRemaining >= 0f);


            if(AnswerTimeRemaining <= 0f)
            {
                //Debug.Log("timeRemaining = " + timeRemaining);
                AnswerTimeStarted = false;

                if(!HurryUp)
                    RaiseAnswerEvent(WrongAnswer); // di sini ada
fungsi next question, jangan balapan dengan questiontimer


                StopAnswerTimer();
            }


        }

    }

    float QuestionTImeRemainingtime;

    private IEnumerator QuestionTimer()
    {
         QuestionTImeRemainingtime =
GameSettings.instance.QuestionTime;

        do
        {
            QuestionTImeRemainingtime -= Time.deltaTime;

            QuestionTime.text =
((int)QuestionTImeRemainingtime).ToString();

            yield return null;
        } while (QuestionTImeRemainingtime >= 0);

        //
        //PhotonRoom.instance.NumOfWrongAnswerPlayer = 1;
        //RaiseAnswerEvent(WrongAnswer);

        if(true) //(HurryUp)
        {

            object[] content = new object[] {
QuestionController.instance.QuestionIndex };//send current question
index
```

```
            RaiseEventOptions raiseEventOptions = new
RaiseEventOptions { Receivers = ReceiverGroup.All };
            SendOptions sendOptions = new SendOptions { Reliability =
true };

            PhotonNetwork.RaiseEvent(3, content, raiseEventOptions,
sendOptions);


            //one player send broadcast to all
            //PhotonRoom.instance.NumOfWrongAnswerPlayer += 1;
            //RaiseAnswerEvent(3); //send to all

            /*
            //ShowResult((int) ResultType.AllAnswered, "", "");

            QuestionController.instance.QuestionIndex += 1;


QuestionController.instance.Next(QuestionController.instance.QuestionI
ndex);
            GameUIController.instance.EnableBuzzer(true);


            PhotonRoom.instance.NumOfWrongAnswerPlayer = 0;
            */


        }
        else
        {
            /*
            if(PhotonRoom.instance.NumOfWrongAnswerPlayer<2)
            {
                 //RaiseAnswerEvent(3);
                //every player load next question individually


                QuestionController.instance.QuestionIndex += 1;


QuestionController.instance.Next(QuestionController.instance.QuestionI
ndex);
                GameUIController.instance.EnableBuzzer(true);


                PhotonRoom.instance.NumOfWrongAnswerPlayer = 0;

            }

            */
        }
        GameUIController.instance.WrongAnswerNotif.SetActive(false);
        for (int i = 0; i <
GameUIController.instance.PlayerBellNotif.Count; i++)

GameUIController.instance.PlayerBellNotif[i].SetActive(false);
    }
}
```