# BAB I

# PENDAHULUAN

## A. Latar Belakang

Dengan kemajuan teknologi informasi yang sedang berlangsung di berbagai sektor saat ini, termasuk baik dalam aspek perangkat lunak maupun perangkat keras, terutama dalam komputer dan bidang *multimedia*, tuntutan akan teknologi juga semakin meningkat. Teknologi ini diperlukan untuk mendukung beragam keperluan manusia, tidak hanya dalam mempermudah aktivitas sehari-hari, namun juga dalam pendidikan dan hiburan. Salah satu wujud hiburan dalam dunia digital adalah permainan komputer, yang dapat dinikmati oleh berbagai kalangan, termasuk anak-anak maupun orang dewasa. Khususnya pada anak-anak, permainan ini menjadi salah satu bentuk hiburan paling diminati pada masa kini, karena menawarkan daya tarik dalam hal kesenangan dan aksesibilitas yang mudah.

Perkembangan teknologi, khususnya dalam industri *game* digital, sering dimanfaatkan sebagai sarana hiburan bagi kalangan anak-anak, bahkan hingga kalangan dewasa sehingga tidak sedikit juga anak-anak yang kecanduan dalam bermain *game*. Kebiasaan bermain *game* ini membuat anak anak sekarang lebih banyak menghabiskan waktu bermain *game* daripada belajar sehingga menyebabkan mereka lebih mudah paham akan *game* ketimbang pelajaran mereka sendiri. Kebiasaan ini dapat dimanfaatkan dengan memperkenalkan alat-alat tradisional daerah mereka sendiri melalui *game* tersebut.

Belakangan ini, terdapat pertumbuhan yang substansial dalam industri permainan, di mana teknologi *multimedia* dimanfaatkan secara luas pada berbagai *platform,* mencakup permainan berbasis seluler, permainan komputer pribadi (PC), dan permainan konsol. Selain itu, terdapat pula perkembangan permainan dalam lingkungan maya, yang lebih dikenal sebagai permainan daring (*online games*). Dari berbagai macam Jenis *game* yang banyak diminati oleh kalangan anak-anak yaitu *game* petualangan (*Adventure),* kondisi ini dapat dimanfaatkan sebagai acuan dalam memilih jenis *game* yang akan di rancang.

Topik ini dipilih untuk memberikan suatu solusi pembelajaran pengenalan alat tradisional daerah Sulawesi Selatan melalui sarana *game*. Maka dari itu judul pada penilitian ini adalah **"Aplikasi *game* petualangan pengenalan alat tradisional Sulawesi Selatan ".** Diharapkan pembuatan *game* ini dapat memberikan pelajaran mengenai pengenalan alat tradisional derah terkhusus pada daerah sulawesi selatan

## B. Rumusan Masalah

Berdasarkan dari uraian latar belakang diatas, maka rumusan masalah yang akan dibahas yaitu :

Bagaimana Merancang sebuah *game* petualangan yang menarik dan sekaligus sebagai sarana pengenalan alat-alat tradisional di Sulawesi Selatan ?

## C. Tujuan Penelitian

Penelitian ini bertujuan untuk Memperkenalkan alat-alat tradisional Sulawesi Selatan melalui sebuah *game* yang dapat dimainkan pada *Personal computer*.

## D. Batasan Masalah

Penelitian ini memiliki beberapa batasan masalah untuk memastikan focus yang tepat. Berikut adalah Batasan-batasan dalam penelitian ini:

1. Aplikasi *game* ini hanya dapat bekerja pada piranti *Personal computer*.
2. Aplikasi *game* ini berbasis 3D.
3. Aplikasi *game* hanya dapat dimainkan offline.
4. Alat alat tradisonal yang akan diperkenalkan dalam *game* ini hanya terbatas pada senjata tradisional, pakaian adat tradisional, dan makanan tradisional Sulawesi Selatan.

## E. Manfaat Penelitian

Adapun manfaat yang diharapkan dalam pembuatan sistem ini yaitu:

1. Manfaat bagi Pembuat Sistem

   Menambah pengetahuan mengenai Pengembangan *Game* dan mampu mengimplementasikan ilmu pengetahuan yang di dapat dari bangku perkuliahan khususnya pembelajaran ilmu komputer.

2. Manfaat bagi Pengguna

   Dengan penyusunan *game* ini diharapkan dapat memperkenalkan sedikit tentang alat-alat tradisional Sulawesi Selatan.

# BAB II

# TINJAUAN PUSTAKA

## A. Kajian Teori

### *1. Personal Computer*

Gambar 2.1 *Personal computer*

Istilah "*Personal computer*" Merupakan kata yang berasal dari Bahasa inggris yang d terjemahkan ke Bahasa indonesia sebagai "komputer pribadi." Terdapat pemahaman bahwa "*personal computer*" adalah suatu perangkat komputer yang digunakan secara individual. Umumnya, komputer ini dapat ditemukan di berbagai lingkungan seperti rumah, kantor, dan toko, serta dapat dibawa ke mana saja. Keberadaannya semakin dimudahkan dengan bentuk yang semakin sederhana, seperti laptop, yang memungkinkan mobilitas tinggi. Selain itu, harga yang terjangkau membuat "*personal computer*" menjadi lebih mudah diakses oleh berbagai kalangan, sehingga berbagai variasi bentuknya hampir dapat ditemukan di berbagai tempat.

Tujuan utama dari *personal computer* (PC) adalah untuk mengolah data *input* dan menghasilkan *output* berupa informasi sesuai dengan kebutuhan pengguna. Proses pengolahan data dimulai dari tahap memasukkan data (*input*)

dan berlanjut hingga menghasilkan informasi yang relevan (*output*). Untuk menjalankan proses ini, komputer mengandalkan sistem yang terdiri dari berbagai komponen yang saling terhubung dan tidak dapat dipisahkan satu sama lain.

Seiring berlalunya waktu dan perjalanan sejarah, terjadi perubahan dan kemajuan yang signifikan dalam berbagai aspek kehidupan, termasuk perkembangan teknologi hingga era modern saat ini. Awalnya dimulai dengan teknologi *personal computer* (komputer pribadi) yang sederhana, seiring waktu teknologi ini berkembang hingga menciptakan teknologi laptop pada tahun 1992. Dari sana, evolusi teknologi komputer terus berlanjut hingga pada saat ini, dengan pencapaian yang jauh lebih maju dan canggih.

.Semakin tinggi ram PC (minimal ram 1 gb) maka semakin baik pula kinerja PC. Selain itu, tampilan pada PC pada saat ini memiliki tampilan yang sangat *elegant* sesuai selera *user*, dan masih banyak lagi tambahan fitur yang semakin ditingkatkan dan mempercanggih sebuah komputer pada zaman modern ini. (Syahrul, 2010)

2. Unity


Gambar 2.2 Unity

Unity adalah sebuah perangkat lunak pengembangan permainan yang menawarkan sebuah mesin render yang kuat yang terintegrasi dengan

seperangkat alat intuitif dan alur kerja yang efisien untuk menciptakan konten 3D interaktif. Aplikasi ini juga menyediakan fasilitas penerbitan untuk berbagai *platform* dengan kesederhanaan, serta beragam fitur yang berguna. Selain itu, Unity memiliki toko aset internal yang memungkinkan pengguna untuk mengakses berbagai sumber daya dan komponen untuk memperkaya proyek mereka. Pendekatan ini mendukung kolaborasi dan berbagi pengetahuan melalui komunitas pengembang yang aktif.

Dalam pengembangan *game* 3D, Unity menawarkan berbagai aset 3D yang dapat digunakan oleh pengembang untuk mempercepat proses pembuatan *game*. Aset-aset ini mencakup model karakter, lingkungan, animasi, dan efek visual yang dapat diimpor langsung ke dalam proyek *game*. Salah satu template yang populer adalah *"Third Person Core,"* yang menyediakan dasar-dasar mekanik dan kontrol untuk *game third-person*, di mana pemain melihat karakter dari sudut pandang orang ketiga. Template ini biasanya mencakup karakter yang dapat dikendalikan, pengaturan kamera *third-person* yang mengikuti karakter, sistem kontrol untuk mengatur gerakan dan aksi, serta deteksi tabrakan dan fisika yang mengatur interaksi karakter dengan lingkungan.

Selain itu, Unity menyediakan fitur *Terrain* yang sangat penting dalam pembuatan pemandangan atau lanskap dalam *game*. *Terrain* memungkinkan pengembang untuk membuat permukaan tanah yang luas dan kompleks, seperti gunung, bukit, lembah, dan dataran, dengan menggunakan heightmap dan alat sculpting. Fitur ini juga mendukung penambahan tekstur yang dapat disesuaikan untuk menciptakan transisi yang halus antara berbagai jenis

permukaan, seperti rumput, pasir, dan salju. Selain itu, pengembang dapat menambahkan detail objek seperti rumput, semak, dan bunga, serta pohon yang dapat merespons angin, untuk memberikan kesan lebih hidup dan realistis pada pemandangan. *Terrain* di Unity juga mendukung pencahayaan dinamis dan bayangan untuk menciptakan suasana yang mendalam. Dengan adanya fitur ini, pengembang dapat menciptakan lingkungan yang besar dan detail secara efisien, tanpa harus membuat setiap elemen secara manual. Semua ini membuat Unity menjadi alat yang sangat kuat dan fleksibel untuk pengembangan *game*, memungkinkan penciptaan pengalaman yang kaya dan imersif bagi pemain.

Di Unity, pengembangan *game* sangat bergantung pada penggunaan *Game Object*, yang merupakan komponen dasar dari semua objek dalam *game*. Setiap elemen di dalam *game*, seperti karakter, kamera, lampu, dan bahkan partikel efek, adalah *Game Object*. *Game Object* dapat diberi berbagai komponen untuk menentukan fungsionalitasnya, seperti *collider* untuk deteksi tabrakan, *renderer* untuk menampilkan grafik, dan skrip untuk perilaku khusus. Untuk menciptakan gerakan dan animasi, Unity menggunakan *Animator*, yang mengontrol animasi dari *Game Object*. Animator memungkinkan pengembang untuk membuat dan mengatur berbagai animasi, seperti berjalan, melompat, atau menyerang, serta transisi antara animasi-animasi tersebut berdasarkan kondisi tertentu.

Unity memiliki *Input System* yang sangat fleksibel dan dapat dikonfigurasi untuk menangani berbagai jenis input dari pemain, seperti *keyboard, mouse*, kontroler, atau perangkat sentuh. *Input System* ini memungkinkan pengembang untuk menetapkan aksi tertentu terhadap *input*

yang diterima, seperti bergerak ke arah tertentu, melompat, atau melakukan interaksi dengan objek di dalam *game*. Pengembang juga dapat mengatur sensitivitas dan *dead zones*, serta menangani input dari berbagai perangkat dengan mudah, yang sangat penting dalam memastikan bahwa *game* dapat dimainkan dengan nyaman dan responsif pada berbagai *platform*. Dengan alat-alat ini, Unity mempermudah pengembangan *game* yang interaktif dan dinamis, memungkinkan penciptaan pengalaman bermain yang lebih hidup dan menarik.

Pada saat mendesain sebuah *game* ada beberapa *tool* yang digunakan beberapa di antaranya yaitu :

a. *File*

*File* digunakan umtuk membuat project baru, menyimpan project, serta membuka proyek *game* yang sebelumnya telah ada.

b. *New Scane*

Berguna untuk membuat tempat penyimpanan adegan *game* baru

c. *Asset*

Digunakan untuk *Import* dan *export assets*

d. *Game Object*

Digunakan untuk membuat *object* pada pembuatan *game*, *object* disini dapat berupa,karakter pemain, arena, serta benda-benda yang di butuhkan untuk melengkapi isi *game* tersebut

e. *Physics*

*Physics* berguna untuk menambahkam komponen fisik pada objek salah satu contohnya agar objek dapat menyentuh objek lain.

(Pranata Baskara Arya,Pamodjie Andre Kurniawan,Sanjaya Ridwan,2018.)

3. Bahasa Pemograman C#

Gambar 2.3 Bahasa Pemograman C#

C# (diucapkan: Csharp atau seesharp) umumnya dianggap sebagai penerus C++ atau versi kompleks dari C++, karena diasumsikan bahwa simbol # adalah kombinasi dari 4 tanda tambah yang disusun menjadi simbol hash. Namun, terlepas dari apakah asumsi ini benar atau tidak, C# adalah bahasa pemrograman yang sangat menjanjikan.

C# merupakan bahasa pemrograman yang dirancang dan dikembangkan oleh Microsoft, yang berfokus pada pemrograman berorientasi objek, dan juga merupakan bahasa yang mensupport pemrograman .NET melalui *Visual Studio*. Dasar C# berasal dari bahasa pemrograman C++, dan C# juga memiliki sejumlah kesamaan dengan bahasa pemrograman lain seperti *Visual Basic*, Java, Delphi, dan tentu saja C++. C# menggabungkan kemudahan sintaksis yang mirip dengan Visual Basic, serta kestabilan yang serupa dengan Java dan C++. Kombinasi ini memberikan keuntungan bagi para pengembang dengan latar belakang pemrograman yang beragam, sehingga mereka dapat menguasai bahasa ini dengan cepat. Kelebihan lain dari C# adalah kesederhanaan sintaksisnya, yang mempermudah proses penulisan

kode. Selain itu, C# juga dianggap lebih mudah dibanding bahasa program lain seperti C++ dan Java.

C# dirancang oleh seorang *programer* dari *microsoft*, Anders Hajlsberg.

Anders merancang C# karena mengetahui kekurangan C++, Delphi, Java, dan *Smaltalk*, sehingga Anders membuat bahasa C# yang lebih kuat. Hal ini dapat memperjelas mengapa C# mempunyai beberapa kesamaan dengan bahasa tersebut.

Dalam ekosistem pengembangan program .NET, bahasa C# tergantung pada *Common Language Runtime* (CLR) sebagai sumber pustaka, dan setiap program yang ditulis dalam C# memerlukan CLR (dan oleh karena itu .NET Framework) untuk dieksekusi. Analogi ini mirip dengan kebutuhan runtime library pada *Visual Basic* 6 untuk menjalankan aplikasi. C# memiliki fleksibilitas yang luas dalam menciptakan berbagai jenis solusi perangkat lunak, mulai dari aplikasi Windows, aplikasi konsol, hingga solusi berbasis web. Ketergantungan C# pada CLR dan .NET Framework mendorong keseragaman dan interoperabilitas dalam ekosistem .NET, memungkinkan pengembangan aplikasi yang efisien dan seragam di berbagai *platform*.

Berikut beberapa alasan yang menjadi dasar pemilihan bahasa C# untuk membuat aplikasi, yaitu:

a. C# beriorantasi objek

   Kemampuan C# menunjukkan bahwa itu adalah bahasa beorientasi objek. C# dengan mudah membuat objek, kelas, melakukan *enkapsulasi*, *inheritance*, dan *polimorfisme*.

b. C# cukup sederhana

Bahasa C# memiliki sifat kesederhanaan yang berasal dari dasar-dasar bahasa C, C++, dan bahkan Java. Namun, C# berhasil menghadirkan tingkat kesederhanaan yang lebih tinggi dibandingkan bahasa-bahasa tersebut, karena C# dirancang dengan mengeliminasi kelemahan yang ada pada dasar-dasarnya.

c. Mampu membuat berbagai aplikasi

Kita dapat membuat berbagai aplikasi dengan C#, seperti pengolah kata, grafik spreedsheets, atau bahkan compiler bahasa pemograman..

d. Efisien

Bahasa C# termasuk bahasa pemograman yang memiliki sedikit *keywords* karena mengandalkan *library* yang sangat lengkap. Jadi para pengembang dapat mengingatnya dan memahami kegunaanya dengan baik.

e. C# bersifat *modular*

Bahasa C# sangat *modular*, karena setiap kelas disimpan dalam *namespace* dan dapat digunakan kembali oleh program lain yang membutuhkannya. (Asy'arie Faris, 2017)

4. Blender

Gambar 2.4 Blender

Blender lebih dikenal sebagai aplikasi pembuatan objek 3D gratis dengan sumber terbuka, tetapi juga mendukung semua konsep 3D dan dapat membuat aplikasi dan permainan video 3D interaktif. Dengan *pipeline* terpadu dan proses pengembangan yang responsif, Blender sangat cocok untuk pengguna individu atau studio kecil..

Blender adalah salah satu *program* yang jadi rekomendasi untuk pembuatan karakter, item, dan *maps* karna blender sudah menyiapkan berbagai *tool* untuk mendesainnya contoh *tool* yang sering digunakan *game develop*:

*a. Snap 3D Cursor*

Secara prinsip, Kursor 3D berperan dalam menentukan lokasi suatu objek, sementara Snap berfungsi sebagai alat untuk menemukan titik referensi. Penggunaan Kursor 3D dilakukan dengan menekan kombinasi tombol SHIFT+S.

*b. Adding object*

"*Adding object*" merupakan opsi yang berguna untuk memasukkan objek ke dalam lingkungan tiga dimensi. Anda dapat melakukan ini

dengan menekan kombinasi tombol SHIFT + A atau melalui pilihan menu "*Add*" yang terletak pada bagian "*Header*".

*c. Transformation*

Transformasi adalah upaya untuk memindahkan, memutar, atau mengubah ukuran suatu objek. Tool ini sangat penting dalam mengatur posisi dan orientasi objek dalam ruang 3D.

*d. Screen Layouts*

Saat Anda membuka Blender, layar akan ditampilkan seperti yang ditunjukkan di atas secara *default*. Anda dapat mengubah tampilan sesuai dengan kebutuhan Anda, misalnya jika Anda perlu merancang simulasi *visual*, Anda dapat memilih tata letak animasi, yaitu tampilan yang digunakan untuk menangani animasi dan sebagainya. Alat ini terletak di sudut kiri atas layar dan merupakan bagian dari *Header* Utama, yang merupakan bilah menu di bagian atas tampilan Blender. (Akbar, 2021)

Cara pembuatan suatu objek di Blender melibatkan beberapa langkah dasar. Pertama, pengguna biasanya memulai dengan membuat objek dasar seperti kubus atau bola menggunakan opsi "*Adding object*". Setelah objek dasar dibuat, pengguna dapat menggunakan *tool* "*Transformation*" untuk memindahkan, memutar, atau mengubah ukuran objek sesuai kebutuhan. Untuk penempatan yang lebih presisi, pengguna bisa memanfaatkan "*Snap 3D Cursor*". Akhirnya, untuk efisiensi kerja, pengguna dapat menyesuaikan "*Screen Layouts*" agar sesuai dengan jenis tugas yang sedang dikerjakan, seperti pemodelan, teksturing, atau animasi.

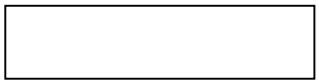
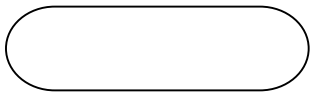
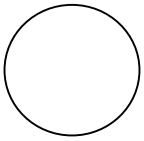
Selain menggunakan Blender, pengembang *game* juga dapat memanfaatkan Mixamo, sebuah *marketplace* yang menyediakan berbagai objek 3D siap pakai. Mixamo menawarkan perpustakaan karakter dan animasi yang dapat diunduh dan langsung digunakan dalam proyek *game*. Ini sangat membantu, terutama bagi pengembang yang membutuhkan aset berkualitas tinggi namun memiliki keterbatasan waktu atau sumber daya untuk membuat semuanya dari awal.

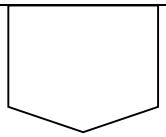
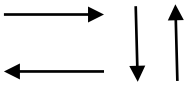
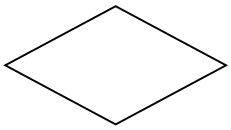
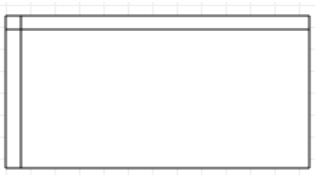
Dengan memadukan kemampuan Blender dalam pembuatan objek 3D dan pemanfaatan aset dari Mixamo, pengembang dapat mempercepat proses pengembangan *game*. Blender digunakan untuk membuat dan menyesuaikan objek 3D, sementara Mixamo menyediakan karakter dan animasi yang dapat diintegrasikan ke dalam Unity untuk menciptakan pengalaman bermain yang lebih kaya dan interaktif. Kombinasi alat ini memungkinkan pengembang untuk fokus pada desain dan logika permainan, memastikan *game* yang dihasilkan tidak hanya fungsional tetapi juga menarik secara visual dan edukatif.

5. Bagan Alir Program

Bagan alur program, juga dikenal sebagai *flowchart*, adalah representasi visual yang merinci langkah-langkah dari suatu proses program. Bagan alur program dibangun berdasarkan diagram alur sistem yang telah diverifikasi. Dalam pembuatan bagan alur program, simbol-simbol berikut ini digunakan :

*Tabel 2.1 Simbol Flowchart*

| SIMBOL | KETERANGAN |
|---|---|
|  | Simbol deklarasi brfungsi memberikan nilai-nilai awal pada variabel |
|  | Simbol *input*/*output* digunakan untuk menggambarkan data yang dimasukkan atau keluar dari program. |
|  | Simbol proses, berperan dalam mengilustrasikan tindakan atau langkah yang dilakukan dalam suatu proses. |
|  | Simbol proses terdefinisi, digunakan untuk menunjukkan operasi khusus yang detailnya dijelaskan di tempat lain. |
|  | Simbol terminal berfungsi untuk memulai dan mengakhiri suatu rangkaian proses. |
|  | Simbol penghubung dimanfaatkan untuk menunjukkan hubungan terputus pada halaman yang sama dalam alur program. |

| SIMBOL | KETERANGAN |
|---|---|
|  | Simbol penghubung digunakan untuk menunjukkan koneksi online yang terputus di halaman lain |
|  | Simbol garis alir, menunjukkan arus dari proses |
|  | Simbol keputusan, digunakan unuk suatu penyelesaian kondisi didalam program |
|  | Simbol Penyimpanan *Internal* digunakan apabila suatu data disimpan secara internal |
|  | Simbol disk storange digunakan apabila *input* berasal dari *disk* |

6. UML *(Unified Modelling Language)*

*Unified Modeling Language* (UML) adalah suatu "bahasa" yang telah menjadi standar industri dalam visualisasi, perancangan, dan dokumentasi sistem perangkat lunak. UML berfungsi sebagai alat untuk membuat abstraksi

dari sistem atau perangkat lunak yang berbasis objek. Pada tahun 1997, UML dikembangkan oleh kelompok manajemen dan sejak itu digunakan secara luas.

Dalam pengembangan berorientasi objek, terdapat prinsip-prinsip utama yang perlu dipahami, termasuk konsep objek, kelas, abstraksi, enkapsulasi, pewarisan, dan polimorfisme.

Metode *Unified Modeling Language* (UML) mengadopsi tiga elemen fundamental untuk menggambarkan sistem atau perangkat lunak yang akan dikembangkan, yaitu:

a) Sesuatu *(things)*

Ada empat *things* dalam *Unified Modelling Language* (UML) :

1) "*Structural things*," merujuk pada bagian-bagian yang bersifat relatif statis dan dapat mencakup elemen-elemen baik dalam bentuk fisik maupun konseptual.
2) "*Behavioral things*," mengacu pada komponen yang dinamis dan sering kali terdiri dari kata kerja dalam model UML, yang merefleksikan perilaku sepanjang ruang dan waktu.
3) "*Grouping things*," merupakan komponen penting dalam UML. Dalam skenario model UML yang kompleks, penggunaan konsep "*package*" digunakan untuk menyederhanakan representasi model. Paket-paket ini dapat diperinci lebih lanjut. Penggunaan paket berguna dalam mengelompokkan elemen, seperti model-model dan subsistem-subsistem.

4) "*Notational things,*" merupakan elemen yang mengklarifikasi model UML. Bagian ini mampu mencakup catatan atau komentar yang menjelaskan fungsi serta atribut dari setiap elemen dalam model UML.

a) Relasi *(relationship)*

Sembilan jenis diagram yang tersedia dalam UML adalah sebagai berikut:

1) Diagram Kelas, Jenis diagram ini bersifat statis dan menampilkan kumpulan kelas, antarmuka, kolaborasi, serta relasi di antara mereka. Diagram ini umum digunakan dalam pemodelan sistem berbasis objek. Walaupun bersifat statis, diagram kelas sering juga mencakup elemen-elemen aktif.
2) Diagram Objek Diagram ini juga bersifat statis dan memvisualisasikan objek-objek serta relasi antar objek. Diagram objek menggambarkan instalasi statis dari segala unsur yang terdapat dalam diagram kelas.
3) Diagram *Use-Case* Dalam sifat statisnya, diagram ini menggambarkan rangkaian kasus penggunaan dan aktor (sebuah jenis kelas). Diagram ini memegang peranan yang sangat penting dalam mengatur dan memodelkan perilaku sistem yang dibutuhkan dan diharapkan oleh pengguna.
4) Diagram Urutan (*Sequence Diagram*) Dalam sifat dinamisnya, diagram ini menyoroti pengiriman pesan pada waktu tertentu. Diagram urutan adalah representasi interaksi yang menitikberatkan pada proses pengiriman pesan antara objek-objek.

5) Diagram Kolaborasi (*Collaboration Diagram*): Memiliki sifat dinamis, diagram ini merupkan bentuk interaksi yang fokus pada struktur organisasi objek yang saling berkomunikasi dengan mengirim dan menerima pesan

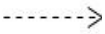
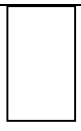
6) *Statechart Diagram* Bersifat dinamis. Diagram state ini memperlihatkan state-state pada sistem memuat *state, transisi, event,* serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka, kelas, kolaborasi dan terutama penting pada pemodelan sistem-sistem yang reaktif.

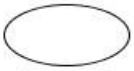
7) *Activity Diagram* Bersifat dinamis. Diagram aktivitas ini adalah tipe khusus dari diagram state yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi dalam suatu sistem dan memberi tekanan pada aliran kendali antar objek.

8) *Component Diagram* Bersifat statis. Dengan kemponen ini memperlihatkan organisasi serta kebergantungan sistem atau perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini terkait dengan diagram kelas di mana komponen biasanya ditugaskan ke satu atau lebih kelas, antarmuka, dan kolaborasi.

9) *Deployment Diagram* Bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan. Diagram ini membuat simpul simpul beserta komponen-komponen yang ada di dalamnya. Diagram ini sangat berguna pada banyak mesin. (N, Sora .2015)

*Tabel 2.2. Simbol Use Case Diagram*

| NO | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| **1** |  | *Actor* | Mengidentifikasi kelompok peran yang diambil oleh pengguna ketika berinteraksi dengan kasus penggunaan. |
| **2** |  | *Dependency* | Hubungan di mana perubahan terjadi pada faktor yang bersifat mandiri dan mempengaruhi elemen-elemen yang tergantung pada faktor tersebut, yang dikenal sebagai elemen independen atau independen. |
| **3** |  | *Generalization* | Ini merujuk pada hubungan di mana objek turunan memiliki pengaruh terhadap perilaku dan struktur data dari objek yang berada di atasnya dalam hierarki. Ini memerinci kasus penggunaan sumber secara eksplisit. |
| **4** |  | *Include* | Menspesifikasikan secara eksplisit *Use Case* sumber. |
| **5** |  | *Extend* | Menspesifikasikan bahwa *Use Case* target memperluas perilaku *Use Case* sumber ke titik tertentu. |
| **6** |  | *Association* | Menghubungkan satu objek ke objek lainnya. |
| **7** |  | *System* | Menentukan paket system dengan cara terbatas. |

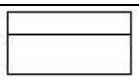
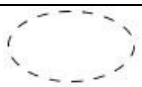
| NO | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| **8** |  | *Use Case* | Deskripsi rangkaian tindakan-tindakan yang dieksekusi oleh sistem untuk menghasilkan suatu hasil yang dapat diukur bagi seorang aktor tertentu. |
| **9** |  | *Collaboration* | Kolaborasi antara aturan-aturan dan elemen-elemen lain yang saling berinteraksi untuk memberikan perilaku yang lebih besar daripada jumlah dan komponennya secara individu, yang dikenal sebagai sinergi |
| **10** |  | *Node* | Elemen fisik yang hadir dan beroperasi saat aplikasi dieksekusi, yang mencerminkan sumber daya komputasi. |

*Tabel 2.3. Symbol Activity Diagram*

| NO | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| **1** |  | *Actifity* | Merupakan visualisasi tentang cara setiap kelas dan antarmuka saling berinteraksi satu sama lain. |
| **2** |  | *Action* | Keadaan sistem yang mencerminkan pelaksanaan dari suatu aksi atau tindakan. |
| **3** |  | *Initial Node* | Proses bagaimana suatu objek dibentuk atau dimulai dalam konteks pengembangan perangkat lunak. |

| NO | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| **4** |  | *Actifity Final Node* | Bagaimana objek dibuat dan dihapus (dihancurkan) dalam siklus hidupnya dalam konteks pengembangan perangkat lunak. |
| **5** |  | *Fork Node* | Suatu aliran yang pada tahap tertentu membagi menjadi beberapa aliran yang berbeda. |

*Tabel 2.4.* Simbol *Sequence Diagram*

| NO | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| **1** |  | *Generalization* | Hubungan di mana objek anak (descendant) mewarisi perilaku dan struktur data dari objek induk (ancestor) yang berada di atasnya dalam hierarki. |
| **2** |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek. |
| **3** |  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |
| **4** |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang dilakukan oleh sistem yang menghasilkan hasil yang dapat diukur bagi seorang aktor. |
| **5** |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. |

| NO | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| **6** |  | *Dependency* | Hubungan di mana perubahan pada elemen independen mempengaruhi elemen yang bergantung pada elemen non-independen. |
| **7** |  | *Association* | Menghubungkan satu objek ke objek lainnya. |

**7.** Pengujian Sistem `

Pengujian adalah suatu proses yang dilakukan terhadap aplikasi atau program dengan tujuan menemukan kesalahan dan potensi masalah sesuai dengan spesifikasi perangkat lunak yang telah ditetapkan sebelum aplikasi tersebut diserahkan kepada pengguna. Ada beberapa metode yang berbeda untuk melaksanakan pengujian perangkat lunak, namun untuk menguji aplikasi ini, metode yang digunakan adalah *white box* dan *black box testing*.

*a. Black box Testing*

*Black box test* terfokus pada apakah unit program memenuhi kebutuhan (*required*) yang disebutkan dalam spesifikasi. Pengujian *black box* hanya menjalankan metode pengujian dengan menjalankan atau menjalankan unit atau modul dan mencatat apakah hasil unit sesuai dengan proses yang diinginkan.

Adapun teknik-teknik yang digunakan dalam pengujian *black box* adalah sebagai berikut:

1) Digunakan untuk menguji fungsi perangkat lunak khusus.

2) Efektivitas perangkat lunak yang diuji hanya diverifikasi berdasarkan hasil yang dihasilkan.
3) Kemampuan program Anda untuk memenuhi kebutuhan Anda dan mengidentifikasi kesalahan.

*b. White box Testing*

Uji coba *white box testing* merupakan metode perancangan *testcase* yang mengunakan struktural untuk mendapatkan *testcase*, test ini digunakan untuk meramal cara kerja perangkat lunak secara rinci kepada *logic path* (jalur logika), perangkat lunak di tes dengan kondisi dan perulangan secara fisik. (Kuliahkomputer, 2018)

## B. Kajian Hasil Penelitian Terdahulu

Diperlukan lebih banyak penelitian terdahulu yang dapat digunakan sebagai data pendukung. Salah satu data pendukung yang peneliti yakini harus menjadi subjek bagian lain adalah pekerjaan sebelumnya yang terkait dengan topik yang dibahas dalam penelitian ini. Dalam hal ini peneliti melakukan temuan-temuan tertentu dalam bentuk risalah/risalah akhir atau jurnal internet, karena tujuan dari penelitian sebelumnya digunakan sebagai acuan yang berkaitan dengan masalah teknologi informasi..

1. Penelitian dalam sebuah skripsi yang dilakukan oleh chaeruddin (2018) dari Parepare yang berjudul "*Treadmill virtual Reality*", pada penelitian ini berfokus pada sistem penggerak menggunakan *virtual reality* sehingga dapat merasakan sensasi berjalan pada sebuah *game* 3D

2. Penelitian dalam sebuah skripsi yang dilakukan oleh Dedi Sumardi (2018) dari Parepare yang berjudul "Pembuatan *Game* Rey-Man Menggunakan Construct 2 Berbasis Android", Perancangan *game platform* yang mengharuskan pemain mengarahkan suatu objek dengan melalui berbagai tahap atau tingkatan area untuk menyerang dan menghindar dari serangan musuh
3. Penelitian dalam sebuah skripsi yang di lakukan oleh Rasnah (2018) dari Parepare dengan judul "Aplikasi penggerak *Scrollbar* Dengan Identifikasi Gerak dan Arah Objek Menggunakan Kamera", penelitian yang dilakukan ini berfokus pada bagaimana menggerakkan sebuah *scrollbar* dengan mengikuti isyarat objek tangan menggunakan kamera.

# BAB III

# METODE PENELITIAN

## A. Jenis Penelitian

Jenis penelitian yang digunakan adalah penelitian eksperimental, di mana data objek diambil langsung melalui percobaan yang dilakukan. Eksperimen dalam penelitian ini terdiri dari pembuatan aplikasi *game* petualangan sebagai sarana pengenalan alat tradisional di Sulawesi selatan .

## B. Lokasi dan Waktu

1. Penelitian ini dilaksanakan di Jalan Kesuma, Kelurahan Kampung Baru, Kecamatan Bacukiki Barat, Kota Parepare, Sulawesi Selatan.
2. Waktu yang digunakan untuk pelaksanaan penelitian adalah sekitar 2 bulan.

## C. Alat dan Bahan

Untuk melakukan proses penelitian dalam pembuatan aplikasi, diperlukan penggunaan perangkat keras dan perangkat lunak sebagai pendukung kegiatan tersebut. Berikut adalah penjelasan mengenai *hardware* dan *software* yang digunakan dalam pembuatan aplikasi *game* ini.

1. Perangkat Keras (*Hardware*)

Tabel 3.1 Spesifikasi Perangkat Keras

| **Spesifikasi** | |
|---|---|
| Merek *Laptop* | Lenovo V14 |
| *Processor Laptop* | Amd Athlon Gold 3150U With Radeon Graphics CPU 2.4 GHz (4 CPUs) |
| *RAM Laptop* | 4 GB |
| *Hardisk* | 256 GB SSD |

Pada Tabel 3.1 ditampilkan spesifikasi perangkat keras laptop yang digunakan dalam penelitian ini. Laptop yang digunakan adalah Lenovo V14, dilengkapi dengan prosesor AMD Athlon Gold 3150U yang memiliki kecepatan 2.4 GHz dengan empat inti CPU dan grafis Radeon. Laptop ini juga memiliki kapasitas RAM sebesar 4 GB dan penyimpanan *internal* berupa SSD dengan kapasitas 256 GB. Spesifikasi ini menunjukkan bahwa perangkat yang digunakan cukup memadai untuk menjalankan aplikasi dan program yang dibutuhkan selama penelitian, khususnya untuk tugas-tugas pemrograman dan pengolahan data yang ringan hingga sedang.

2. Perangkat Lunak (*Software*)

Tabel 3.2 Spesifikasi Perangkat Lunak

| **Spesifikasi** | |
|---|---|
| Sistem Operasi | Windows 10 Pro 64-Bit |
| Aplikasi pembuat *game* | Unity 2020.2.7f1 |

| | |
|---|---|
| Aplikasi Desain Karakter | Blender 2.9 |
| Bahasa Pemrograman | *C#* |

Pada Tabel 3.2 ditampilkan spesifikasi perangkat lunak yang digunakan dalam penelitian ini. Sistem operasi yang digunakan adalah Windows 11, yang mendukung berbagai aplikasi dan perangkat lunak pengembangan. Bahasa pemrograman yang dipilih untuk proyek ini adalah C#, yang dikenal karena kemampuannya dalam pengembangan aplikasi dan *game*. Untuk *code editor*, digunakan *Visual Studio Code*, Blender, dan Unity. *Visual Studio Code* merupakan editor teks yang ringan namun kuat, Blender digunakan untuk pemodelan 3D, dan Unity adalah mesin *game* yang sering digunakan untuk mengembangkan *game* interaktif dan aplikasi 3D. Kombinasi perangkat lunak ini memberikan fleksibilitas dan kekuatan dalam pengembangan proyek penelitian yang mencakup pemrograman dan pembuatan konten visual.

## D. Teknik Pengumpulan Data

Untuk mempermudah pengumpulan data, penulis menggunakan beberapa metode, seperti :

1. Tinjauan Pustaka (*Literature Review*) dilakukan dengan cara mengumpulkan data teoritis yang terkait dengan masalah yang menjadi objek penelitian dari berbagai literatur dan buku-buku yang tersedia di perpustakaan, yang digunakan sebagai dasar teori.

2. Analisis web sejenis, proses pengumpulan data dilakukan untuk mendapatkan informasi yang diperlukan melalui *web* atau *game* yang sama dan membandingkannya.

## E. Teknik Analisis Data

Penelitian ini melibatkan beberapa tahap, yaitu persiapan, studi literatur, pengumpulan data, analisis, perancangan, pengujian, dan implementasi. Penjelasan mengenai setiap langkah tersebut adalah sebagai berikut :

1. Persiapan Penelitian

   Pada tahap ini, peneliti melakukan persiapan untuk penelitian. Persiapan tersebut mencakup penyiapan gambaran awal desain *character* animasi, laptop dan perangkat lunak / *software* yang digunakan selama penelitian yaitu unity dan blender yang mendukung pembuatan aplikasi.

2. Studi Literatur

   Pada tahap ini, peneliti menelaah pustaka dengan meneliti sejumlah buku referensi. seperti Cara membuat *game* sendiri dengan Unity. Selain itu juga mempelajari jurnal hasil penelitian sejenis sebelumnya yang pernah dilakukan oleh orang lain seperti pembuatan *game*.

3. Analisis

   Pada tahap analisis, peneliti meninjau sistem yang saat ini diterapkan, kemudian mengidentifikasi masalah utama yang menjadi fokus penelitian untuk merumuskan solusi alternatif.

4. Perancangan

Peneliti kemudian merancang permainan yang akan di buat serta juga merancang desain *character*.

5. Pengujian

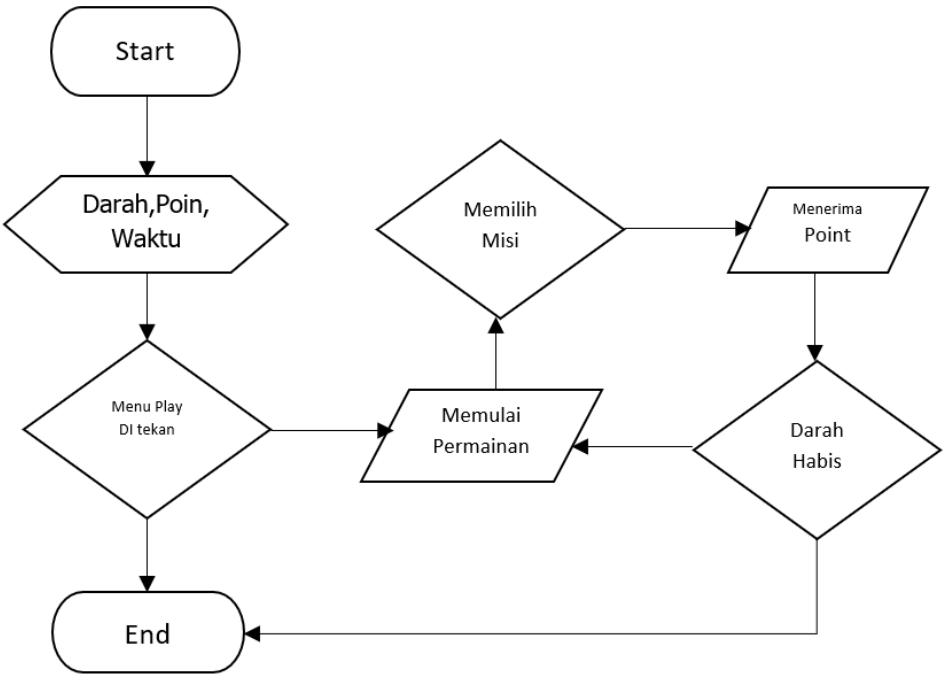
Setelah perancangan selesai, peneliti menguji hasil rancangan tersebut pada PC. Jika ditemukan kekurangan atau kelemahan, proses kembali ke tahap analisis.

6. Implementasi

Setelah selesai pada tahap perancangan dan tidak ada kekurangan yang ditemukan, aplikasi siap untuk dimainkan oleh pengguna.

## F. Diagram Alir

***1.*** *Flowchart Game*




Gambar 3.1 *Flowchart Game*

Pada gambar 3.1 menggambarkan alur permainan dari awal hingga akhir. Permainan dimulai dengan inisialisasi variabel penting seperti darah, poin, dan waktu, yang menentukan kondisi awal pemain. Selanjutnya, pemain diberi opsi untuk memilih menu *play*. Jika pemain memilih untuk tidak menekan menu *play*, permainan akan langsung berakhir. Namun, jika pemain menekan menu *play*, mereka akan diarahkan untuk memilih misi. Setelah misi dipilih, permainan dimulai, dan pemain menjalani proses *gameplay*.

Selama permainan berlangsung, ada dua kemungkinan hasil jika darah pemain habis, pemain harus memulai ulang permainan. Sebaliknya, jika pemain berhasil menyelesaikan misinya, mereka akan menerima poin sebagai bentuk penghargaan. Setelah menerima poin atau jika permainan diulang, pemain kembali ke langkah memilih misi untuk memilih misi baru atau mencoba lagi. Alur ini berakhir ketika pemain memilih untuk tidak melanjutkan dengan menekan end. Secara keseluruhan, *flowchart* ini menggambarkan siklus permainan yang berpusat pada inisialisasi, pilihan misi, pelaksanaan misi, evaluasi kondisi pemain, dan pemberian poin atau pengulangan permainan berdasarkan hasil *gameplay*..

## G. Gambar Rancangan Aplikasi

1. Tampilan Menu Awal

   Tampilan menu awal akan mempunyai tiga tombol yaitu tombol mulai (*play*) dimana tombol ini berfungsi mengarahkan ke *scene* untuk memulai permainan, tombol pengaturan (*setting*) berfungsi mengarahkan ke bagian

pengaturan pada menu awal , tombol berhenti (*quit*) berfungsi untuk menghentikan permainan


Gambar 3.2 Rancangan Main Menu

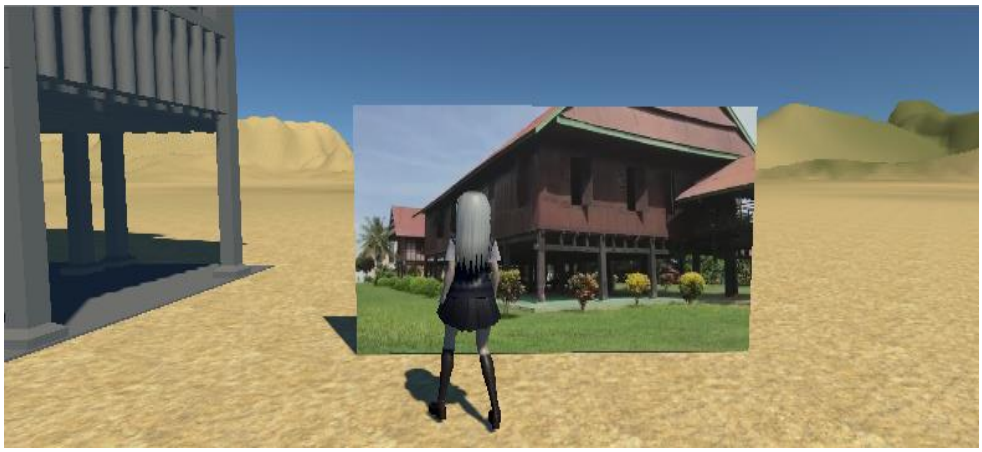
2. Tampilan Ketika Memulai Permainan

Tampilan merupakan *scene* awal *game* saat pemain menekan tombol *play* di main menu, pada tampilan ini menunjukkan kondisi arena yang luas dimana pemain dapat menjelajahi arena tersebut guna mencari objek untuk mengenal *object* tersebut

Gambar 3.3 Tampilan *Scene Playground*

3. Keika Pemain Menyentuh *Object*

Pada saat pemain menemukan objek yang telah di cari sebelumnya pemain dapat melihat informasi lebih lengkap terkait *object* tersebut guna mengenal lebih jauh terhadap objek tersebut seperti pada gambar di bawah ini



Gambar 3.4 Tampilan Informasi *Object*

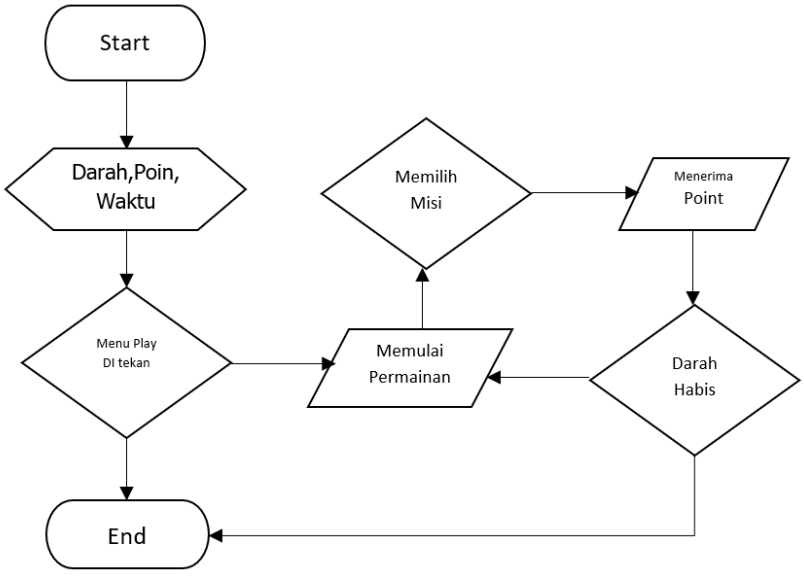
# BAB IV

# HASIL DAN PEMBAHASAN

## A. Hasil

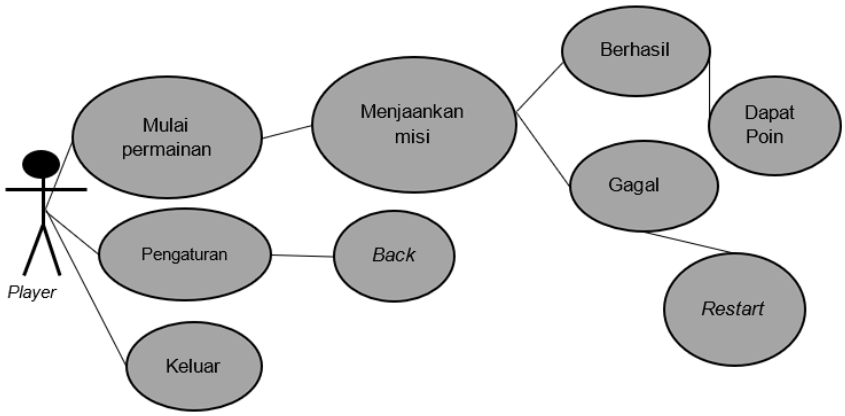
1. *Flowchart* Sistem

*Flowchart* sistem adalah ilustrasi yang menunjukkan bagaimana proses bekerja dan menggambarkan urutan langkah-langkah yang terlibat dalam keseluruhan sistem. Ini menguraikan langkah-langkah dalam prosedur yang ada dalam sistem. Diagram ini hanya mencakup satu komponen, yang disebut *pemain.*




Gambar 4.1 *Flowchart* Pemain

2. Analisis Sistem yang Diajukan




Gambar 4.2 *Use Case* Diagram Aplikasi

Pada Gambar 4.2 diagram use case ini menunjukkan interaksi pemain dengan fitur permainan. Pemain dapat memilih mulai permainan untuk menjalankan misi, yang dapat berakhir dengan berhasil atau gagal. Jika berhasil, pemain mendapat poin, sedangkan jika gagal, pemain dapat memilih restart untuk mencoba lagi. Selain itu, pemain bisa mengakses pengaturan untuk mengubah preferensi atau memilih keluar untuk meninggalkan permainan. Opsi back memungkinkan pemain kembali ke menu sebelumnya. Diagram ini menggambarkan berbagai pilihan yang tersedia bagi pemain selama bermain.

*3. Activity Diagram*

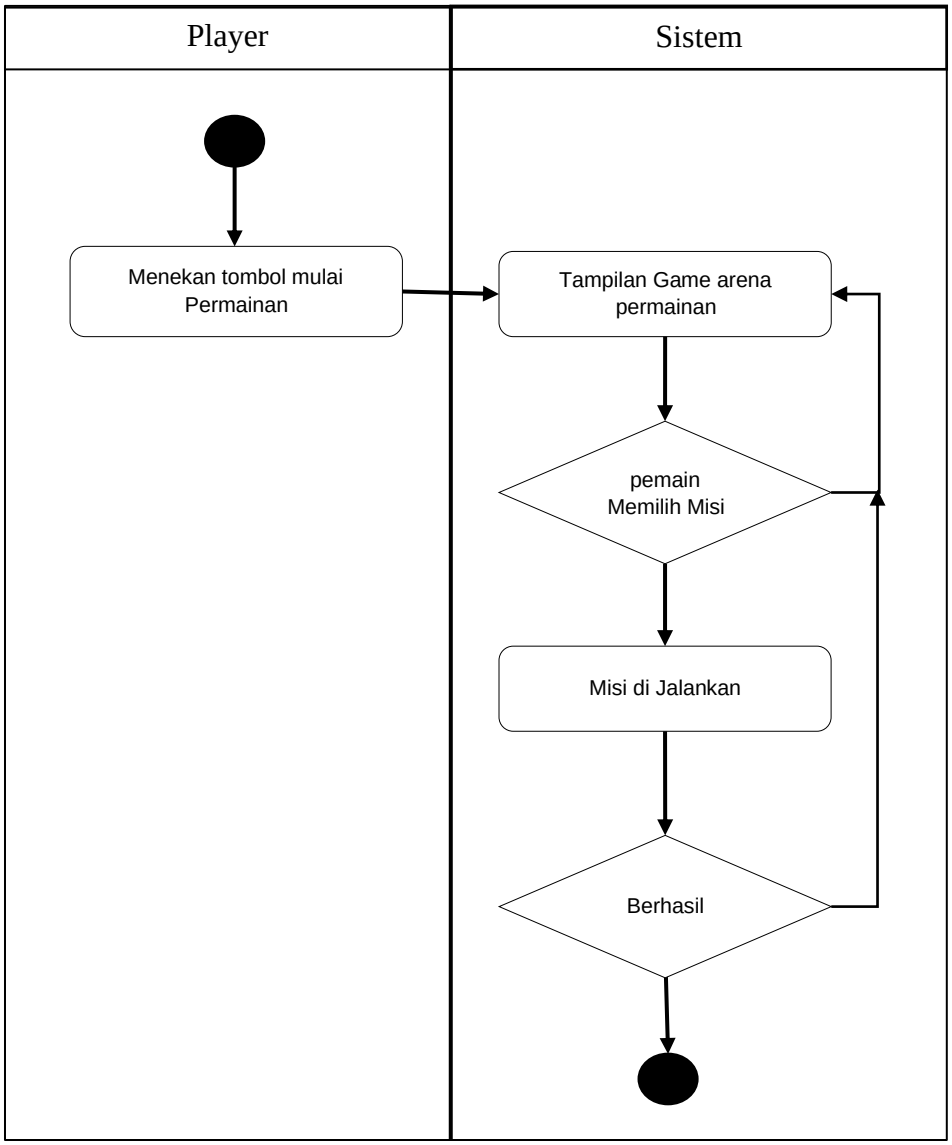
*Activity Diagram* ini mengilustrasikan urutan proses kegiatan dalam sebuah sistem.

a. *Activity Diagram Play*er

*Activity* ini merinci urutan proses yang diikuti oleh pemain saat melakukan pengelolaan data di *Menu* Utama. Proses ini meliputi tiga

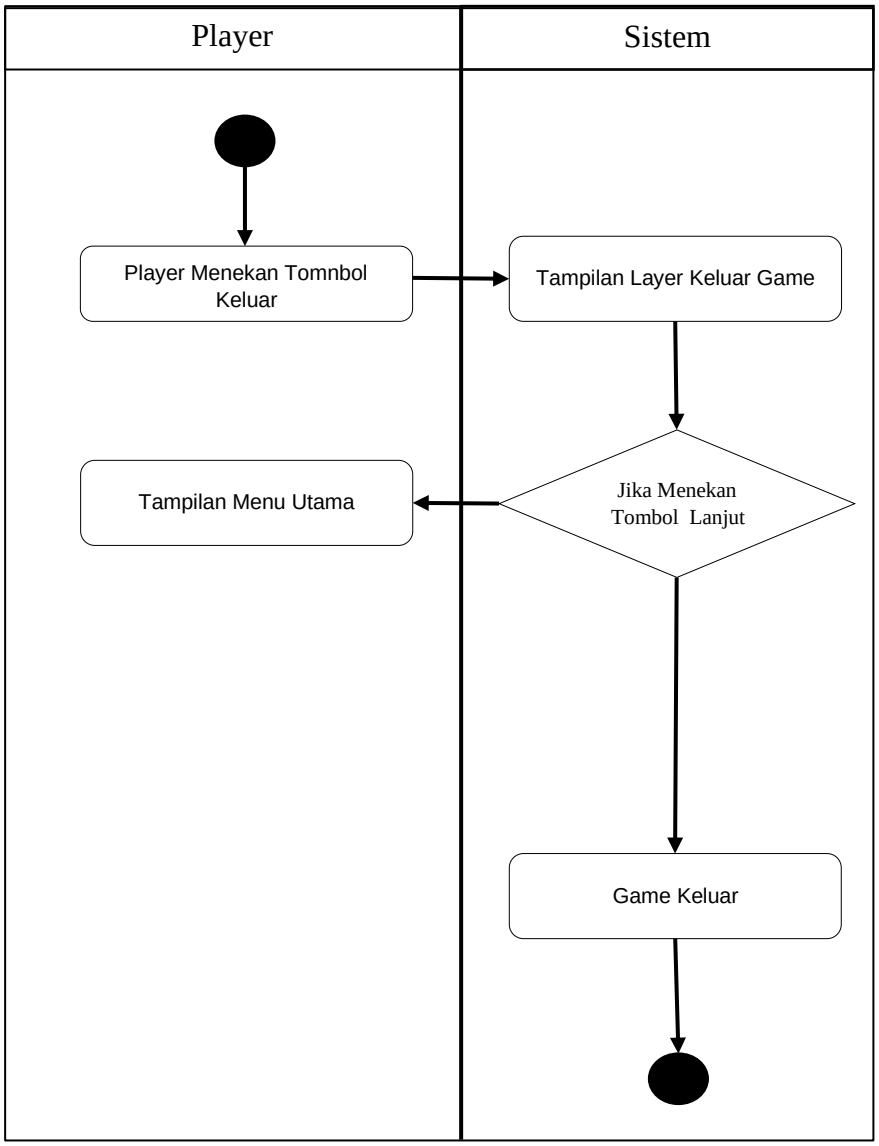
aktivitas utama, yaitu memulai permainan, memproses data, dan keluar dari sistem. Dalam konteks ini, diagram aktivitas menggambarkan langkah-langkah yang diambil oleh pemain dari awal hingga akhir saat berinteraksi dengan pilihan menu yang tersedia.

1) *Activity Diagram* Mulai Permainan




Gambar 4.3 Tampilan *Activity diagram* mulai permainan

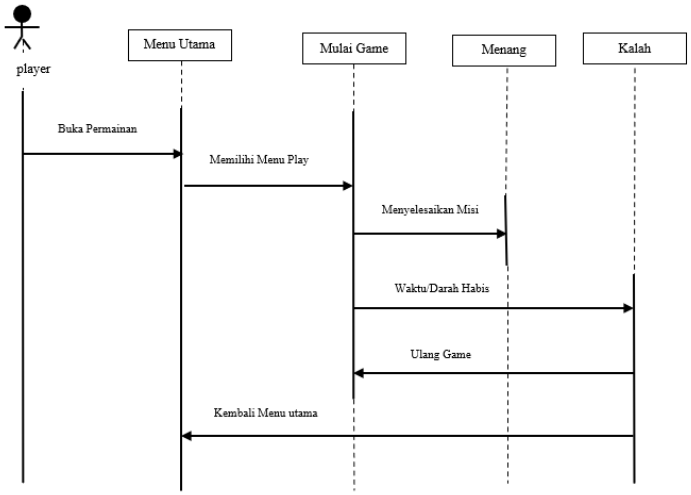
2) *Activity Diagram* Keluar




Gambar 4.4 Tampilan *Activity Diagram* Keluar

4. *Sequence Diagram*

a. *Sequence Diagram player*




Gambar 4.5 Tampilan *Sequence Diagram Player*

Pada gambar 4.5 menggambarkan alur interaksi pemain dengan sistem dalam permainan. Interaksi dimulai dengan pemain memilih opsi buka permainan, yang kemudian membawa pemain ke menu utama. Di sini, pemain dapat memilih menu *play* untuk memulai permainan. Selama permainan berlangsung, pemain harus menyelesaikan misi yang telah ditetapkan. Jika pemain berhasil, mereka akan mencapai status menang. Namun, jika waktu habis atau nyawa pemain berkurang hingga habis, kondisi waktu/darah habis akan terjadi, dan permainan akan berakhir dengan status Kalah. Dalam kedua kondisi, pemain memiliki pilihan untuk ulang *game* dan mencoba lagi atau kembali ke menu utama untuk memulai permainan baru atau memilih opsi lain. Diagram ini menjelaskan alur dan respon sistem terhadap aksi yang diambil oleh pemain dalam permainan.

## B. Pembahasan

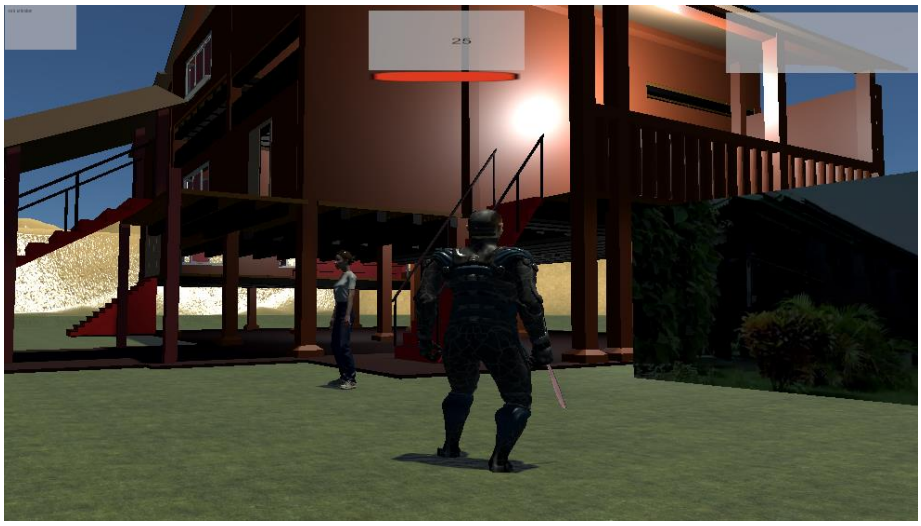
Pada bagian ini akan membahas hasil dan tampilan *Game* Petualangan Pengenalan Budaya Sulawesi Selatan

1. Tampilan Sistem

   a. Tampilan *Main Menu*


Gambar 4.7 Tampilan *Main Menu*

   b. Tampilan Permainan



Gambar 4.8 Tampilan *Playground*

c. Tampilan Papan Informasi Objek


Gambar 4.9 Tampilan Papan Informasi *Object*

d. Tampilan *full screen* Video objek

Gambar 4.10 Tampilan *Full screen* Video Informasi

2. Pengujian *Black box*

Tabel 4.1 *Black box* Tampilan Awal *Game*

| **Tes Faktor** | Hasil | **Kesimpulan** |
|---|---|---|
| Tampilan awal *game* | ✓ | Berhasil membuka *game* |
|  | | |

Pada table 4.1 pengujian *black box* untuk tampilan awal *game* menjelaskan bahwa fokus utamanya adalah memastikan tampilan awal *game* muncul dengan benar. Pengujian ini bertujuan untuk mengecek apakah saat *game* dibuka, tampilan awal yang diharapkan akan muncul, memungkinkan pemain untuk mengakses berbagai pilihan seperti *play*, *setting*, dan *back*. Hasil pengujian menunjukkan bahwa tampilan awal *game* berhasil dimunculkan dengan baik, yang ditunjukkan oleh tanda centang di kolom hasil. Gambar di bawah tabel memperlihatkan tampilan awal *game* yang menunjukkan pemandangan pegunungan dengan seorang penunggang kuda di depannya. Kesimpulannya, pengujian ini berhasil karena sistem berhasil menampilkan layar awal *game* seperti yang diharapkan.

Tabel 4.2 *Black box* Tampilan *Playground*

| **Tes Faktor** | Hasil | **Kesimpulan** |
|---|---|---|
| Tampilan menu *playground* saat Tombol *play* ditekan | ✓ | Berhasil mengarahkan ke menu *playground* |
|  | | |

Pada table 4.2 pengujian *black box* untuk menu *playground*, dijelaskan bahwa saat tombol *play* ditekan, tampilan yang diharapkan adalah menu *playground*. Pengujian ini bertujuan untuk memastikan bahwa ketika tombol *play* diklik, sistem berhasil mengarahkan pemain ke lingkungan *playground* dengan benar. Hasil pengujian menunjukkan bahwa tombol *play* berfungsi dengan baik, karena pemain langsung diarahkan ke tampilan *playground* seperti yang ditunjukkan oleh gambar karakter di dalam lingkungan terbuka dengan medan berpasir dan latar belakang pegunungan. Dari pengujian ini dapat disimpulkan bahwa sistem berhasil mengarahkan pemain ke Menu *playground*, sehingga tombol *play* telah bekerja sesuai fungsinya.

Table 4.3 *Black box Menerima Misi*

| **Tes Faktor** | Hasil | **Kesimpulan** |
|---|---|---|
| Pemain berbicara dengan NPC untuk menerima misi | ✓ | Berhasil menerima misi |
|  | | |

Pada table 4.3 pengujian *black box* untuk menerima misi, dijelaskan bahwa pemain diharuskan berbicara dengan NPC (*Non-Play*er *Character*) untuk menerima misi. Pengujian ini dilakukan untuk memastikan bahwa pemain dapat berinteraksi dengan NPC dan menerima misi sebagaimana yang diharapkan. Hasil pengujian menunjukkan bahwa sistem berhasil memproses interaksi antara pemain dan NPC, sehingga pemain dapat menerima misi. Hal ini terlihat dari gambar, di mana karakter pemain berinteraksi dengan NPC di depan bangunan, dan dialog NPC mengonfirmasi pemberian misi. Kesimpulannya, pengujian ini berhasil, karena sistem telah memungkinkan pemain untuk menerima misi dengan lancar setelah berinteraksi dengan NPC.

Tabel 4.4 *Black box* Menjalankan Misi

| **Tes Faktor** | Hasil | **Kesimpulan** |
|---|---|---|
| Pemain menjalankan misi yang telah diterima | ✓ | Berhasil menjalankan misi |
|  | | |

Pada table 4.4 pengujian *black box* untuk menjalankan misi, dijelaskan bahwa pemain harus menjalankan misi yang telah diterima sebelumnya. Pengujian ini bertujuan untuk memastikan bahwa setelah pemain menerima misi, mereka dapat melanjutkan dan menyelesaikan misi tersebut sesuai instruksi. Hasil pengujian menunjukkan bahwa pemain berhasil menjalankan misi yang diberikan. Hal ini ditunjukkan dalam gambar di mana karakter pemain sedang berada di dalam sebuah ruangan yang mungkin terkait dengan misi tersebut. Kesimpulannya, pengujian ini berhasil karena sistem memungkinkan pemain untuk melanjutkan dan menjalankan misi yang telah diterima dengan baik.

Tabel 4.5 *Black box* Menyelesaikan Misi

| **Tes Faktor** | Hasil | **Kesimpulan** |
|---|---|---|
| Pemain berhasil menjalankan misi dan menerima poin | ✓ | Berhasil menjalankan misi dan menerima poin |
|  | | |

Pada tabel 4.5 menjelaskan bahwa pemain harus menjalankan misi dan menerima poin sebagai hasilnya. Pengujian ini bertujuan untuk memastikan bahwa setelah pemain menjalankan misi, mereka juga dapat menerima poin sebagai bentuk pencapaian. Hasil pengujian menunjukkan bahwa pemain berhasil menyelesaikan misi dan mendapatkan poin sesuai yang diharapkan. Hal ini terlihat dalam gambar di mana karakter pemain berinteraksi dengan lingkungan permainan yang mungkin terkait dengan misi tersebut. Kesimpulannya, pengujian ini berhasil karena sistem memungkinkan pemain untuk menyelesaikan misi dan menerima poin dengan baik.

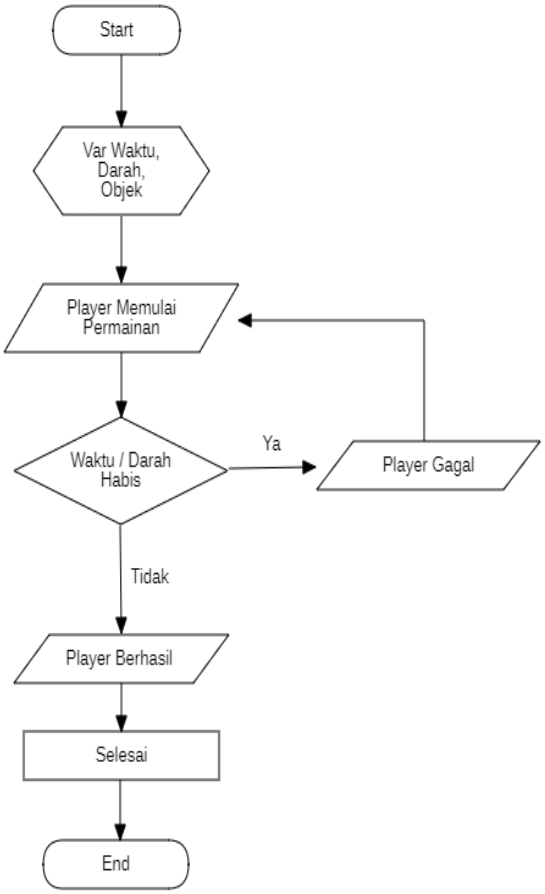
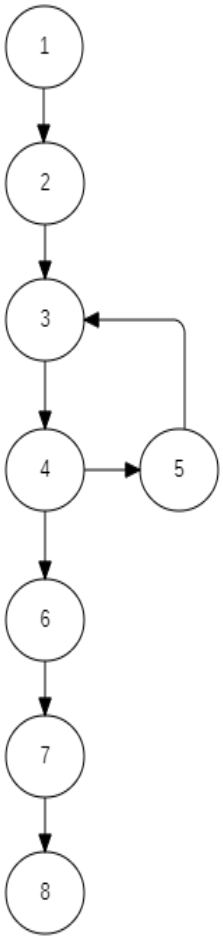
Tabel 4.6 *black box Game Over*

| **Test Faktor** | Hasil | **Kesimpulan** |
|---|---|---|
| Tampilan *game over* ketika darah pemain habis | ✓ | Berhasil mengarahkan ke tampilan *game over* saat darah pemain habis |
|  | | |

Pada table 4.6 Pengujian *black box* untuk tampilan *game over* dijelaskan bahwa pemain harus melihat tampilan *game over* ketika darah pemain habis. Pengujian ini bertujuan untuk memastikan bahwa setelah darah pemain habis, sistem akan mengarahkan pemain ke tampilan *game over* sesuai instruksi. Hasil pengujian menunjukkan bahwa sistem berhasil menampilkan tampilan *game over* yang diharapkan. Hal ini ditunjukkan dalam gambar di mana karakter pemain terlihat pada layar *game over* dengan opsi *quit* dan *restart*. Kesimpulannya, pengujian ini berhasil karena sistem memungkinkan tampilan *game over* muncul dengan baik saat darah pemain habis.

3. Pengujian *White box*

Tabel 4.7 *White Box*

| *Flowchart* | *Flowgraph* |
| --- | --- |
|  |  |

Menghitung jumlah *Region*, *Cyclomatic Complexity*, *Independent Path*, dan *Grafik Matriks.*

a. *Independent Path* pada *Flowgraph* atas adalah:

*Path* 1 = 1-2-3-4--6-7-8

*Path* 2 = 1-2-3-4-5-3-4-6-7-8

b. Menghitung Kompleksitas *Cyclomatic dari Edge* (E) dan Node (N): Rumus untuk *Cyclomatic Complexity* adalah

V(G) = E - N + 2

| | |
|---|---|
| *E* | = 8 |
| N | = 8 |
| Dengan Predikat *Node* | = 2 |

Jadi

| | |
|---|---|
| V(G) | = E – N + 2 |
| | = 8 – 8 + 2 |
| | = 2 |
| Predikat *Node* | = P + 1 |
| | = 1 + 1 |
| | = 2 |

c. Berdasarkan perhitungan *Cyclomatic Complexity* dari *flowgraph* tersebut, nilai V(G) adalah 2, menunjukkan bahwa terdapat 2 *region* dalam *flowgraph* tersebut.

# BAB V

# PENUTUP

## A. Kesimpulan

Penelitian ini menunjukkan bahwa aplikasi *game* yang dikembangkan telah berhasil berfungsi sesuai dengan yang diharapkan, dengan kemampuan menjalankan misi-misinya secara optimal. Objek-objek yang disertakan dalam game ini tidak hanya memberikan tantangan, tetapi juga berhasil menyampaikan pengetahuan tambahan kepada pemain, menjadikannya sebuah pengalaman bermain yang edukatif.

Hasil pengujian membuktikan bahwa pemain mampu memahami dan menyelesaikan misi-misi yang disediakan dengan baik, yang pada gilirannya meningkatkan kualitas pengalaman bermain. *Game* ini tidak hanya menyediakan hiburan, tetapi juga menjadi alat pembelajaran yang efektif dengan menyatukan elemen *edukatif* dan *gameplay* yang menarik. Melalui kombinasi yang tepat dari kedua elemen ini, *game* tersebut mampu menciptakan keseimbangan yang ideal antara kesenangan bermain dan proses pembelajaran, menjadikannya sarana yang bermanfaat bagi pengembangan pengetahuan pemain..

## B. Saran

Dari hasil penelitian ini, ada beberapa hal yang dapat diperbaiki dan dikembangkan. Pertama, *game* ini sebaiknya ditambah dengan lebih banyak misi dan materi edukasi agar bisa mencakup lebih banyak topik dan menjangkau lebih

banyak pemain. Kedua, penting untuk melakukan evaluasi lebih lanjut guna melihat seberapa besar pengetahuan pemain meningkat setelah bermain. Selain itu, menambahkan fitur interaktif seperti mode *multiplayer*. Pengembang juga perlu memperhatikan umpan balik dari pengguna untuk terus meningkatkan kualitas *game*. Terakhir, agar lebih banyak orang bisa mengakses *game* ini, sebaiknya dibuat versi yang bisa dimainkan di berbagai perangkat seperti tablet, dan ponsel.

# DAFTAR PUSTAKA


Akbar, 2021 " Ap aitu Blender 3D? "(online) (https://akbarproject.com/apa-itu-blender/) ( diakses pada tanggal 10 juli 2021)

Ardhianto, Eka, Wiwien, Hadikurniawati & Edy, Winarno. 2012. "Augmented Reality Objek 3 Dimensi dengan Perangkat Artoolkit dan Blender. Jurnal Teknologi Informasi".

Asy'arie Faris, 2017,"Mengenal Bahasa C# dan Kegunaannya Dalam Bahasa Pemograman"(online) (https://www.Farisasyarie.com/2017/09/mengenal-bahasa-c-c-sharp-dan.html?m=1) (diakses pada tanggal 10 Juli 2021) .

Budiutomo Nanang (2017), Simbol *Flowchart* Beserta Fungsi,gambar dan Keterangannya Lengkap (online) (https://bukubiruku.com/simbol-*flowchart*-dan-fungsinya) (diakses pada 10 juli 2021)

Chaeruddin,, 2018. "Treadmill virtual reality", Parepare:Universitas Muhammadiyah Parepare.

Dedi Sumardi,2020, "Pembuatan *Game* Rey-Man Menggunakan Constuct 2 Berbasis Android" Parepare:Universitas Muhammadiyah Parepare

Kuliahkomputer, 2018,"Pengujian Sistem Informasi *Black box*/*White box*".(online)(http://www.kuliahkomputer.com )(diakses pada tanggal 10 julis 2021)

N, Sora .2015 . "Pengertian UML dan jenis – jenisnya" (online) (http://www.pengertianku.net/2015/09/pengertian-uml-dan-jenis-jenisnya-serta-contoh-diagramnya.html.(diakses pada tanggal 10 juli 2021)

Pranata Baskara Arya,Pamodjie Andre Kurniawan,Sanjaya Ridwan.2018."Mudah Membuat *Game* dan Potensi Finansialnya Dengan Unity 3D". Elex Media Komputindo: Jakarta.

Rasnah, 2017, "Aplikasi Penggerak Scroolbar Dengan Identifkasi Gerak dan Arah Objek Menggunakan Kamera",Parepare: Universitas Muhammadiyah Parepare.

Sianipar R.H, 2017, "Panduan Praktis Pemograman C# bagi Pemula".Andi Offset : Yogyakarta

Sumardi Dedi, (2018) "Pembuatan *Game* Rey-Man Menggunakan Construct 2 Berbasis Android",Parepare:Universitas Muhammadiyah Parepare.

Syahrul, 2010, "Organisasi Dan Arsitektur Komputer" Andi Offset : Yogyakarta