BAB I
PENDAHULUAN

1.1 Latar Belakang

Sejak dirilisnya *video game* pertama yang diciptakan oleh seorang fisikawan bernama William Higinbotham pada tahun 1958, dunia *video game* telah mengalami berbagai macam revolusi. Salah satu hasil dari revolusi di dunia *video game* adalah terlahirnya berbagai macam *genre-genre* yang memiliki berbagai macam *video game* yang dapat dipilih dan dimainkan oleh masyarakat, hal ini merupakan refleksi dari perkembangan teknologi dan kreaktivitas dari manusia. Industri *video game* telah menjadi salah satu industri terbesar di dunia hiburan dan hanya akan terus berkembang seiring dengan berjalannya waktu.

Menurut Elder (1971) *game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan dibatasi oleh konteks tertentu. Jadi, pada dasarnya sebuah *video game* dapat diartikan sebagai sebuah perangkat lunak interaktif yang dapat digunakan untuk sarana hiburan, permainan peran, dan simulasi. *"Any game which allows a number of players to assume the roles of imaginary characters and operate with some degree of freedom in an imaginary environment"*(Lortz, 1979) .

Salah satu game paling populer berdasarkan genrenya adalah game *role-playing game* (*RPG*). Sebuah *game RPG* adalah sebuah dunia dimana pemain akan berinteraksi dengan dunia yang telah disusun oleh *game developer* dengan menggunakan suatu karakter atau *avatar* yang mereka pilih. Pemilihan karakter yang tepat adalah salah satu hal terpenting yang dapat dilakukan oleh pemain dalam perjalanan mereka dalam sebuah *game RPG*. Ketika pemain mengetahui kekuatan dan kelemahan dari suatu karakter yang tersedia, maka pemain akan lebih mampu melihat bagaimana cara mengembangkan karakter yang akan mereka pilih.

Selama bermain *game RPG,* pemain juga akan berinteraksi dengan karakter yang disebut *Non-Player Character* (*NPC*). *NPC* atau yang disebut juga dengan *Autonomous Agent* adalah karakter atau entitas yang berada di dalam komputer animasi dan media interaktif seperti *game* dan *Virtual Reality* di mana memiliki kemampuan untuk melakukan gerakan ataupun interaksi secara otomatis oleh komputer dan tidak dikontrol oleh pemain (Du et al., 2021).

Sebuah *NPC* dibuat dengan tujuan untuk mensimulasikan perilaku dari manusia agar dapat meningkatkan realisme dan imersi dari dunia *game.* Untuk meningkatkan kualitas dari reaksi *NPC,* maka diimplementasikan algoritma *decision making* (Warpefelt et al., 2016). *Decision making* adalah serangkaian algoritma yang akan memberikan *NPC* kemampuan untuk mengambil keputusan sebagai respon dari pengembangan narasi atau pemain. Dalam pembuatan NPC dan perilakunya di dalam game, algoritma yang banyak digunakan adalah *Finite state method (FSM)* (Fathoni et al., 2020; Refnaldi, *nethod* (Sekhavat, 2017; Uludağlı & Oğuz, 2023), *Logic-based* ıl., 2019), *Goal oriented*(Suyikno & Setiawan, 2019), dan *Artificial*)(Sagredo-Olivenza et al., 2017). Dari semua metode ini, metode gunakan untuk memberi kemampuan pengambilan keputusan ah *Tree-based methods* karena metode ini mudah dipahami, dikembangkan.

Sebagai model yang membantu *decision making* bagi NPC, model-model di atas juga dapat diterapkan pada *decision making* lainnya yang ada pada game yaitu *character progression.* Penelitian mengenai pemanfaatan model pengambilan keputusan dalam *character progression* masih terbatas. Automasi pada sistem pengambilan keputusan di *character progression* dapat membantu pemain dalam progresi pengembangan karakter *game RPG* yang akan meningkatkan *user experience* dalam memainkan video game.

Telah banyak terjadi kasus dimana pemain tidak mengembangkan karakter yang mereka mainkan dengan efisien sehingga terjadi ketidakseimbangan antara pemain dengan karakter musuh. Hal ini telah banyak terjadi pada *game-game* seperti *Dark Souls* dan *Bloodborne*, kedua game yang memfokuskan pemain untuk mengembangkan dan menguasai karakter yang mereka mainkan. Jika pemain gagal dalam dua hal tersebut, ada kemungkinan pemain akan berhenti bermain atau yang disebut *"rage quit*".

Dengan memperhatikan masalah ini, penelitian ini akan fokus dalam menerapkan metode *Euclidean Distance* dalam sistem *character* progression. Hasil dari penelitian ini diharapkan dapat menjadi dasar dalam pembuatan sistem *character progression* otomatis dengan menggunakan algoritma *decision-making* dan dapat meningkatkan kepuasan pengguna video game. Dengan demikian, penelitian ini tidak hanya memperkaya wawasan teoritis dalam pengembangan *game*, tetapi juga memiliki dampak positif langsung pada pengalaman bermain pemain *game RPG*.

1.2 Rumusan Masalah

Rumusan masalah pada pembuatan sistem *Character Progression* adalah sebagain berikut:

1. Bagaimana cara mengimplementasikan metode *Euclidean Distance* dalam sistem *Character Progression*?
2. Bagaimana cara membangun sistem *Character Progression*?

1.3 Tujuan Penelitian

Adapun tujuan penelitian sebagai berikut:

1. Mengimplementasi metode *Euclidean Distance* dalam sistem *Character Progression.*
2. Merancang dan membangun sistem *Character Progression.*

1.4 Manfaat Penelitian

Adapun manfaat dari penelitian yang dilakukan adalah menambahkan wawasan dan pengetahuan dalam cara pengimplementasian metode *Euclidean Distance* dan bagaimana cara merancang dan membangun suatu sistem *Character Progression*.

1.5 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. Penelitian ini hanya akan menggunakan *Game Engine Unity* sebagai alat
   ınya, sehingga tidak akan menbahas implementasi pada *platform*

   hanya akan mengembangkan sistem pada *platform Personal*
   , sehingga tidak membahas optimalisasi untuk mobile atau konsol

   dikembangkan tidak melibatkan fitur *multiplayer online* atau
   ta daring, semua proses berlangsung secara local.

4. Sistem yang dirancang hanya untuk *single-player,* jadi tidak mempertimbangkan interaksi antar pemain.
5. Penelitian ini hanya membahas perancangan dan implementasi sistem *Character Progression* dan pemrogramannya dengan menggunakan *Game Engine Unity.*

1.6 Tinjauan Pustaka

1.6.1 Video Game

Game dapat diterjemahkan dari bahasa Inggris sebagai sebuah permainan. Sedangkan, *video* merupakan suatu media visual elektronik. Jadi, dapat diartikan bahwa *video game* adalah suatu media permainan elektronik. Menurut (Elder, 1971),*"a game is an activity among two or more independent decision-makers seeking to achieve their objectives in some limiting context*." Sebuah *game* diartikan sebagai sebuah kegiatan yang menyangkut dua atau lebih pemain. Dimana, pemain akan melakukan suatu aktivitas yang memiliki suatu tujuan dan batasan-batasan tertentu.

Jadi, berdasarkan dua definisi tersebut, sebuah *video game* dapat diartikan sebagai suatu perangkat elektronik yang dapat dipergunakan sebagai objek hiburan yang akan memiliki aspek-aspek seperti keputusan pemain, tujuan, dan batasan-batasan objektif.

Menurut Sulistyo (2010) sebuah g*ame* dapat dibagikan berdasarkan dari jenis-jenis *platform* yang tersedia. Dimana, *platform* merupakan suatu sistem hasil kombinasi dari perangkat lunak dan perangkat keras yang memungkinkan sebuah *game* untuk dapat beroperasi dengan optimal.

Berdasarkan *platform* yang digunakan, *video game* dapat dibagikan menjadi beberapa jenis:

1. *Arcade games* yaitu *game* dimainkan menggunakan mesin yang dibuat khusus untuk *game* tertentu.
2. *PC games* yaitu *game* yang dapat dimainkan dengan menggunakan komputer atau laptop.
3. *Console games* yaitu *game* yang dimainkan dengan menggunakan perangkat khusus seperti *Playstation* atau *XBOX.*
4. *Handheld games* yaitu *game* yang dimainkan dengan perangkat seperti *Nintendo Switch.*
5. *Mobile games* yaitu *game* dimainkan menggunakan perangkat s*martphone* (Hashim et al., 2008).

Selain pembagian berdasarkan *platform, video game* juga dapat dibagikan berdasarkan dari *genre* nya. Dimana, *genre* dari *video game* adalah salah satu cara kalsifikasi dan mengelompokkan *video game* berdasarkan elemen-elemen desain tertentu yang dimiliki masing-masing *game.*

Video game memiliki beberapa genre. Menurut (Henry, 2010) genre-genre video game adi:

ng mengharuskan pemain untuk menyelesaikan tantangan yang butuhkan koordinasi dan reaksi tangan-mata yang baik. *Game* i biasanya bertempo cepat dan paling disukai oleh mereka yang *game*.

2. *Arcade games*
   Tipe *game* yang memiliki cerita sederhana dan lebih memfokuskan kepada *gameplay* dari *video game* tersebut.
3. *Fighting games*
   Tipe *game* di mana pemain akan bertarung melawan karakter lain yang dikendalikan oleh pemain lain atau kecerdasan buatan (AI) *game* tersebut.
4. *Racing games*
   Tipe *game* yang melibatkan pemain mengendalikan suatu kendaraan dengan tujuan untuk mencapai garis finis sebelum orang lain.
5. *Strategy games*
   Tipe *game* yang akan membuat pemain mengandalkan strategi, yang akan mentest kemampuan otak pemain dalam menentukan jalannya permainan. Artinya, keterampilan pengambilan keputusan pemain mempunyai arti penting dalam menentukan hasil dari permainan.
6. *Simulation games*
   Tipe *game* yang mereplikasi aspek-aspek dari dunia nyata yang kemudian ditanamkan dalam sebuah *game*. Permainan-permainan yang ada pada *genre* ini sering kali sangat mendetail dan bergantung pada strategi dan keterampilan untuk berhasil.
7. *Role-Playing Games*
   Tipe *game* dimana pemain akan berperan sebagai karakter yang dapat berinteraksi dengan dunia yang telah disusun oleh pembuat *game*

1.6.2. Role-Playing Game

Role-Playing Game (*RPG*) adalah salah satu *genre* yang terlahir seiring dengan perkembangan *video game*. *Game-game* dalam *genre* ini biasanya memfokuskan perkembangan cerita yang kemudian akan dibagi dalam beberapa misi atau level. Seorang pemain akan mengontrol satu atau lebih karakter dengan memberikan perintah, yang kemudian akan membuat karakter berinteraksi dengan dunia *game* (Zagal & Deterding, 2018) . Sepanjang permainan berlanjut, terdapat atribut-atribut tertentu yang dimiliki oleh karakter yang akan mengalami perkembangan. Dimana, perkembangan tersebut akan menentukan perubahan dari kepribadian karakter.

Sebuah *Role-Playing Games* biasanya juga memiliki mekanisme interaksi yang kompleks dan dinamis, yang ditentukan dan dikembangkan antara karakter pemain dan dunia di mana mereka hidup (Lee et al., 2021). Hal ini mencakup interaksi dengan lingkungan dunia, serta pemain non-karakter lainnya.

Sebuah *Role-Playing Games* pada umumnya akan memiliki elemen-elemen penyusun, seperti:

1. Cerita dan latar.

n *inventory.*
ression dan sistem level.

grafik (Bjarnason, 2020).

Gambar 1. The Witcher 3: Wild Hunt

Gambar 2. Honkai: Star Rails

1.6.3. *Character Progression*

Character Progression dapat diartikan sebagai perkembangan aspek-aspek suatu karakter dalam cerita (Sheldon, 2022). Baik dalam bentuk kepribadian, kemampuan, pengetahuan, atau hubungan dengan karakter lain atau dunia *game* itu sendiri. Perkembagan ini akan terjadi seiring dengan perjalanan cerita yang telah disusun oleh *game developers*.

Suatu karakter tidak hanya dibentuk berdasarkan kelebihan yang menjadikannya menonjol, namun juga kelemahan dan kekurangan yang ia miliki (Anderson & Johnson, 2021). Karakter yang memiliki latar belakang, keyakinan, ketakutan, harapan, akan membuat karakter terlihat semakin nyata, *relatable*, dan simpatik kepada pemain. Seiring berjalannya *game*, para karakter akan ditantang untuk menghadapi berbagai konflik yang

dengan kelemahan mereka sendiri. Respons mereka terhadap
entukan bagaimana mereka mengatasi kelemahan mereka dan
mereka berkembang. Ketika karakter mengalami beberapa
ıereka menghadapi beberapa konflik, itu berarti karakter tersebut
cara bertahap.

ıgan karakter yang berdasarkan dengan perkembangan cerita,
apat mengalami perkembangan dalam bidang *gameplay.* Dimana,

perkembangan ini akan menunjukkan progresi dari kemampuan pemain dalam mengendalikan karakter yang ia mainkan. Contoh perkembangan karakter dengan Gambar 3 dan Gambar 4.

Gambar 3. Karakter level 1

Gambar 4. Karakter level 166

1.6.4. *Stat Points*

Stat Points adalah nilai-nilai numerik yang akan diberikan kepada entitas pada suatu *video game*. Nilai-nilai ini akan diberikan dan dibagikan kepada atribut-atribut tertentu yang dimiliki oleh entitas dalam dunia *game,* dimana atribut-atribut tersebut akan memberikan peningkatan kinerja atau kemampuan mereka didunia *game*.

suatu *game* yang menggunakan *stat points* akan menggunakan berikut:

ᵓ) mewakili jumlah serangan yang dapat diterima karakter sebelum

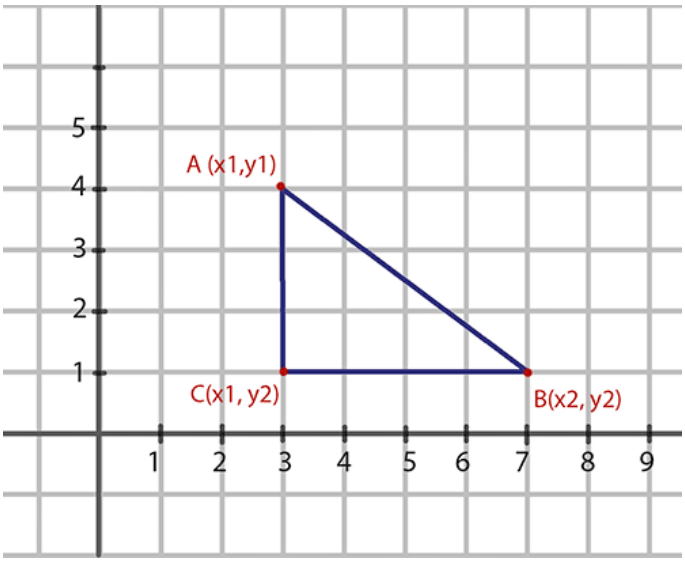
ᵓ) mewakili jumlah dan tipe serangan sihir yang dapat digunakan

- *Strength* (*STR*) mewakili kekuatan serangan fisik yang ditimbulkan karakter.
- *Speed* (*SPD*) mewakili seberapa cepat karakter dan frekuensi dari serangan dan peluang untuk menghindari serangan.
- *Endurance* (*END*) mewakili seberapa kuat pertahanan karakter yang kemdian akan mengurangi kekuatan serangan musuh (Kelly, 2020).

Stat Points juga dapat digunakan untuk melihat progresi dari karakter yang dimainkan oleh pemain. Seperti yang dapat dilihat pada Gambar 3 dan Gambar 4. *Stat Points* bagi karakter yang masih level 4 dengan karakter level 166 dapat dibandingkan dengan jelas.

1.6.5. *Euclidean Distance*

Euclidean distance adalah perhitungan jarak dari 2 buah titik dalam *Euclidean space*. *Euclidean space* diperkenalkan oleh Euclid, seorang matematikawan dari Yunani sekitar tahun 300 B.C.E. untuk mempelajari hubungan antara sudut dan jarak. *Euclidean* ini berkaitan dengan Teorema Phytagoras dan biasanya diterapkan pada 1, 2, dan 3 dimensi. Ilustrasi dari pengukuran jarak *Euclidean* dapat dilihat pada Gambar 5.



Gambar 5. Ilustrasi pengukuran jarak Euclidean

Sebuah jarak *Eucledian* dapat dipergunakan dalam suatu sistem *game* untuk mengukur perbedaan antara dua entitas berdasarkan atribut tertentu, sehingga membantu dalam pengambilan keputusan yang lebih akurat dalam mekanisme *gameplay*.

Dimana, jarak *Euclidean* dapat dipergunakan dalam proses pengambilan keputusan akan mentukan target serangannya. Metode jarak *Euclidean* akan menghitung posisi terdekat dari target-target yang tersedia. Rumus *Euclidean* seperti berikut:

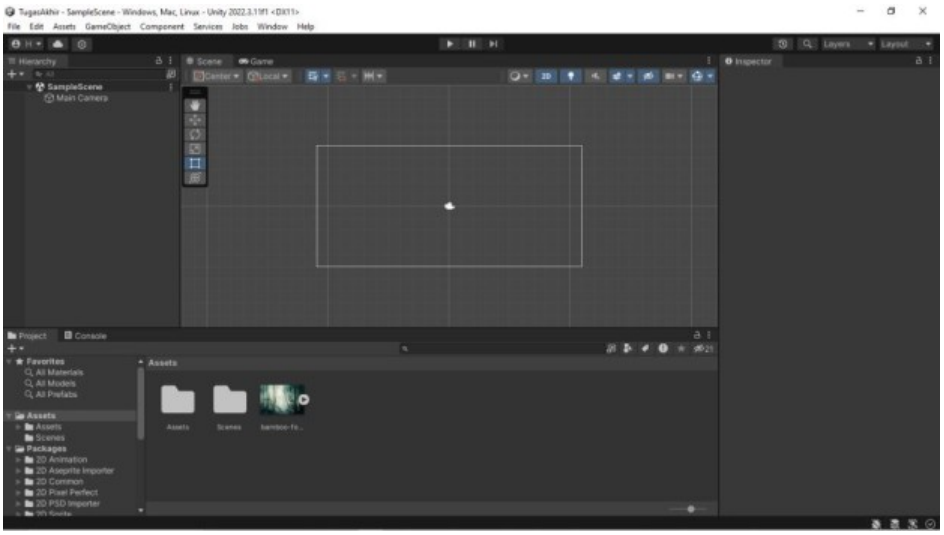
$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- **(x_1, y_1)** adalah posisi karakter.
- **(x_2, y_2)** adalah posisi target.
- **d** adalah jarak *Euclidean* antara karakter dan target.

1.6.6. Unity Engine

Unity Engine merupakan suatu *game engine* yang dikembangkan dan dirilis pada tahun 2005 oleh *Unity Technologies* yang bersifat *cross-platform.* Menurut Michael Lewis (2002) *game engine* adalah kumpulan modul kode simulasi yang secara tidak langsung akan menentukan perilaku *game* (logika game) atau lingkungan *game* (data level). Sistem ini akan mencakup modul yang menangani *input, output* (*Render 3D*, gambar *2D*, suara), dan *generic fisika*/dinamika untuk dunia *game*.

Tujuan *Unity* dipergunakan dalam proses pembuatan *game* karena akan memberikan kemudahan bagi pengembang permainan karena menyediakan fungsi-fungsi inti dari sebuah permainan, misalnya grafika (menghasilkan grafika 2-dimensi atau 3-dimensi), fisika (menghitung dan menyimulasikan hukum-hukum gerak dan hukum fisika lainya), audio, atau kecerdasan buatan. *Game engine Unity* dapat dilihat pada Gambar 6.



Gambar 6. Unity Version 2022

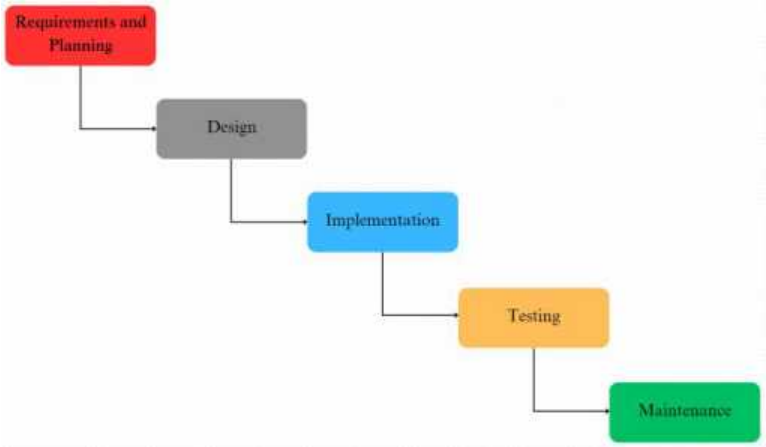
Unity memiliki beberapa komponen utama yang digunakan dalam membuat *game* seperti *scene, C#, game object* dan *component*. Berikut penjelasan dari beberapa komponen *Unity:*

ı *window* yang digunakan untuk membangun *game*. Di dalamnya ıan mengatur *object* dalam sebuah *scene*. Dalam halaman resmi h tempat kerja dan tempat mengatur objek-objek yang ada. *Scene* ɔ pada *browser internet* dimana *game developer* dapat berkreasi erbagai macam objek seperti rintangan, dekorasi didalamnya (Unity.

b. C# (C-*Sharp*)
   C# (C-*Sharp*) merupakan sebuah bahasa pemrograman yang dibuat oleh *Microsoft* dan salah satu bahasa pemrograman yang dapat digunakan dalam dunia pengembangan *game*. C# (C-Sharp) menjanjikan produktivitas, fleksibilitas serta kemudahan yang ada dari aplikasi sebelumnya yaitu *Visual Basic*, Java, dan C++ yang mengadopsi kemampuan dari penggabungan aplikasi sebelumnya.
c. *Game Object*
   Dalam *unity* segala jenis objek yang ada disebut sebagai *game object*. Dalam *Unity* (2022[b]) *Game object* merupakan sebuah *building block* pada *scene* di Unity, dan bertindak sebagai wadah untuk komponen fungsional yang menentukan bagaimana tampilan serta apa yang akan dilakukan oleh *game object* tersebut.
d. *Component*
   Seperti yang dijelaskan sebelumnya pada *game object*, *game object* merupakan wadah dari *component*. *Component* yang dimaksud adalah bagian fungsional dari setiap *game object*. *Component* ini berisi properti yang dapat di edit untuk menentukan perilaku *game object* (Unity, 2022[c]).

1.6.7. Metode *Waterfall*

Metode *Waterfall* merupakan salah satu proses pengembangan perangkat lunak (Firmansyah & Jamilah, 2018) yang popular dan sering dipergunakan oleh para pengembang sistem, baik itu sistem informasi, aplikasi yang berbasis web maupun desktop, namun tidak hanya itu saja metode *waterfall* juga dapat diterapkan di berbagai jenis software lainnya, salah satunya adalah didalam pengembangan aplikasi *game*, hal ini dikarenakan tahapan dari metode *waterfall* sangatlah fleksibel untuk di implementasikan. Metode w*aterfall* dapat dilihat pada Gambar 7.



*/*aterfall

ɔ-tahapan yang ada pada metode *waterfall*, kita dapat menjelaskan bagai berikut:

Planning

Proses pengumpulan kebutuhan dilakukan secara intensif untuk memastikan apa saja perangkat lunak yang dibutuhkan oleh *user*. Spesifikasikan kebutuhan perangkat lunak pada tahap ini perlu untuk di dokumentasikan.

2. *Design*
   Tahap ini adalah proses yang memfokuskan pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka dan prosedur pengkodean.

3. *Implementation*
   Pada tahap ini dimulai proses pengembangan berdasarkan dokumen desain. Tahap ini melibatkan pemrograman, pembuatan aset game (karya seni, efek suara, musik), desain level, dan pengintegrasian berbagai komponen.

4. *Testing*
   Tahap pengujian akan fokus pada perangkat lunak secara dari segi lojik dan fungsional untuk memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir munculnya kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan diinginkan.

5. *Maintenance*
   Setelah *game* dirilis, terus berikan dukungan dengan mengatasi masalah apa pun yang dilaporkan oleh pemain, merilis *patch* atau pembaruan, dan kemungkinan menambahkan fitur atau konten baru berdasarkan masukan pemain.

BAB II
METODE PENELITIAN

2.1 Waktu dan Lokasi Penelitian

Penelitian ini akan dilakukan pada Maret 2024 hingga Mei 2024 di Laboratorium Rekayasa Perangkat Lunak Universitas Hasanuddin.

2.2 Tahapan Penelitian

Pada penelitian ini akan menggunakan metode *waterfall.* Dimana, tahap pertama dari metode ini adalah tahap perencanaan konsep yang kemudian dilanjutkan dengan tahap pemodelan, berikutnya tahap implementasi sistem, kemudian tahap pengujian, dan diakhiri dengan tahap pemeliharaan.

Penelitian ini menggunakan metode *Waterfall* karena metode ini akan memberikan lebih banyak kontrol saat proses pengembangan sistem karena metode ini dilakukan secara bertahap (Fagarasan et al., 2021). Dengan metode ini, segala kesalahan atau kekurangan dari tahap sebelumnya dapat diidentifikasi dan diperbaiki sebelum berpindah ke tahap berikutnya. Metode waterfall terdiri dari beberapa tahapan yaitu analisis kebutuhan, desain sistem, implementasi, pengujian, dan pengoperasian serta *maintenance* (A. A. Adenowo & Adenowo, 2013). Meskipun demikian, penelitian ini dilaksanakan hanya hingga tahapan pengujian.

2.2.1 Analisis Kebutuhan

Tahap ini akan dimulai dengan mengidentifikasikan segala materi yang diperlukan untuk merancang suatu *game RPG.* Dalam merancang suatu *game,* pertama dibutuhkan suatu *game engine* yang akan menjadi *platform* yang akan digunakan untuk menyusun dan menjalan *game.* Berikutnya, akan dikumpulkan segala a*sset* yang akan digunakan dalam *game*. *Asset* yang dibutuhkan untuk penelitian ini berupa *character sprite, audio, animasi,* dan sebagainya. Contoh dari *character sprite* yang akan digunakan dapat dilihat seperti pada Gambar 8.

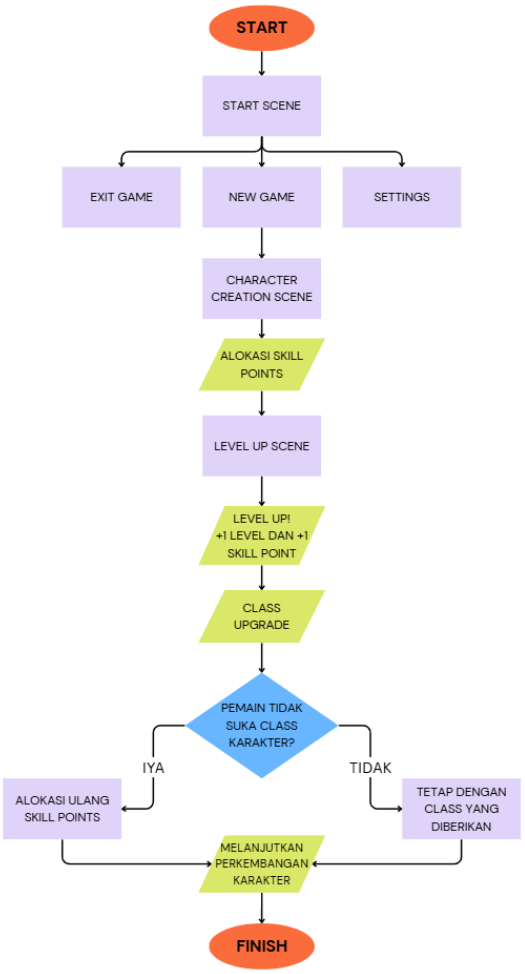
venturer

2.2.2 Perancangan Sistem

Pada tahap ini akan dimulai proses perancangan *gameplay,* perancangan *leveling system,* perancangan algoritma *decision making,* pembuatan *user interface,* dan penulisan dari s*cript.*

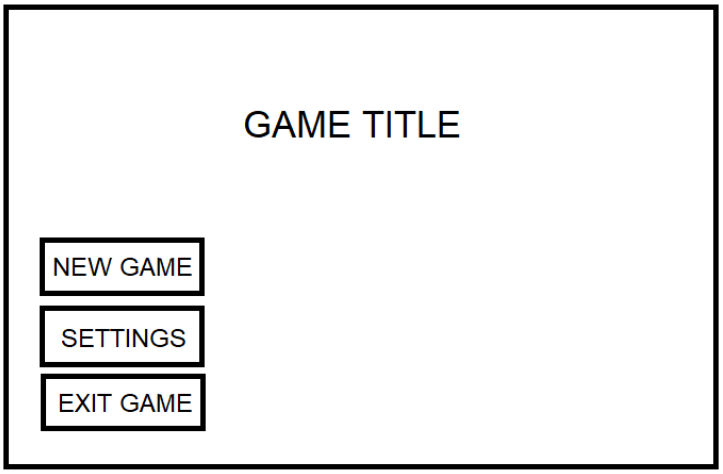
1. Perancangan *Gameplay*

Tahap pertama yang akan dilakukan adalah perancangan dari *gameplay.* Hal ini karena *gameplay* akan menjadi suatu pondasi dari permainan yang dibuat. Alur dari *game* dapat digambarkan dengan menggunakan *flowchart* seperti pada Gambar 9.



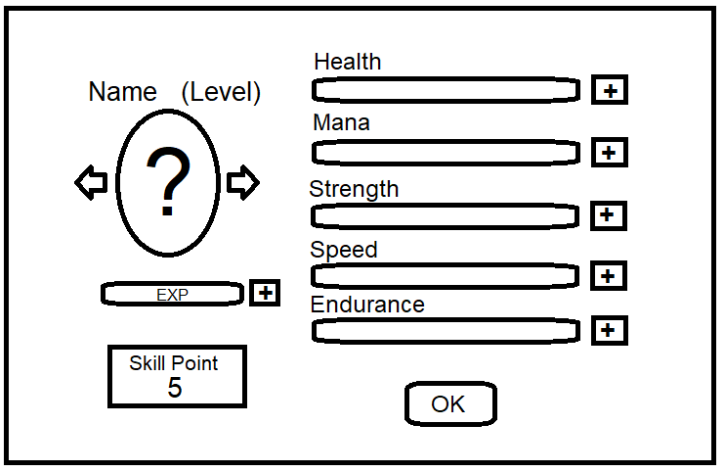
***Gambar 9** Flowchart alur game*

emiliki tiga tombol yang dapat dipilih oleh pemain. Tombol pertama untuk memulai permainan, tombol kedua adalah "*Settings*" untuk an kontrol dan pengaturan keras suara, dan tombol ketiga adalah ntuk mematikan *game.* Gambar dari *start scene* dapat dilihat pada



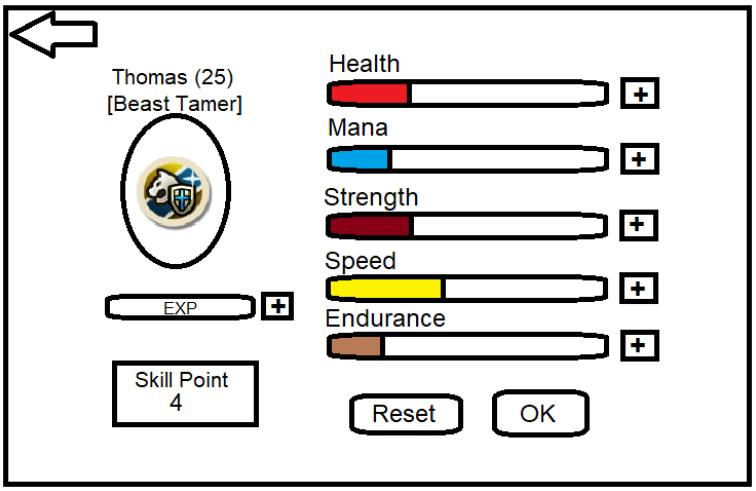
Gambar 10. Start Scene

Jika pemain menekan tombol "*New Game*", maka *game* akan mengarahkan pemain kepada *character creation screen* seperti pada Gambar 11.



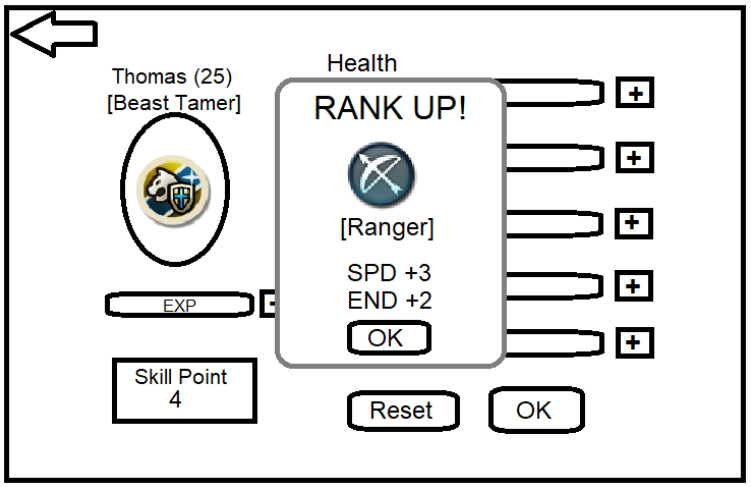
Gambar 11. Character creation

Pada *character creation,* pemain dapat memilih karakter yang ingin mereka mainkan dan menentukan pada *stat* mana s*kill point* akan diberikan. Setelah memilih karakter dan membagikan *stat point*, maka pemain dapat menekan tombol "OK" untuk melanjutkan ke *character screen. Character screen* dapat dilihat pada Gambar 12.



Gambar 12. LevelUp Scene

Pada *character screen*, pemain dapat membagikan *skill points* yang ia terima setiap kali karakter yang dimainkan *level up.* Jika pemain telah mencapai *level* tertentu dan jika karakter yang dimainkan memiliki jumlah *stat point* tertentu, akan muncul sebuah *floating screen* seperti pada Gambar 13.



Gambar 13. Class upgrade

Jika pemain tidak menyukai *gameplay* dari karakter, maka pemain dapat menekan mengembalikan karakter ke *class* "*adventurer*" dan membagikan ɟ dimiliki untuk mencoba mendapatkan tipe karakter yang berbeda. ı mengakhiri permainan, pemain dapat menekan tombol *left arrow* ujung atas s*creen* untuk kembali ke *main menu.*

2. Perencanaan Metode *Euclidean Distance*

Euclidean Distance akan aktif apabila karakter yang dimainkan memenuhi 2 variabel yang ditetapkan untuk mengalami *rank up.* Variabel pertama yang harus dimiliki karakter adalah, karakter yang dimainkan telah *level up* hingga *level* 11, 31, atau 51. Variabel kedua adalah karakter yang dimainkan telah memiliki s*tat points* yang mendekati dengan *stat points* yang diperlukan untuk menerima *rank up* berikutnya.

Setelah mecapai level yang telah di tentukan, sistem akan melakukan pengecekan s*tat point* untuk melihat tipe karakter yang dapat direkomendasikan ke pemain. Jika terdapat 2 tipe karakter dapat direkomendasian ke pemain, maka sistem akan mencek *stat point* yang memiliki jarak terdekat diantara karakter-karakter tersebut. Bila *stat point* karakter yang dimainkan lebih dekat dengan s*tat point* karakter tipe A daripada s*tat point* karakter tipe B, maka system akan memilih karakter tipe A untuk direkomendasikan ke pemain.

Bila karakter yang dimainkkan mencapai level 11 dengan *stat point* seperti berikut: HP = 7, MP = 3, STR = 4, SPD 3, dan END = 3. Dan terdapat 3 tipe karakter yang yang memiliki *stat point* seperti berikut:

- Opsi A: STR = 10, SPD = 5.
- Opsi B: MP = 12, HP = 6.
- Opsi C: END = 8, STR 8.

Dengan menggunakan metode jarak *Euclidean* dan akan didapatkan hasil perhitungan seperti berikut:

- Jarak ke Opsi A**:** $\sqrt{(10-4)^2+(5-3)^2} = \sqrt{6^2+2^2} = \sqrt{36+4} = \sqrt{40} = 6.32$
- Jarak ke Opsi B:$\sqrt{(12-3)^2+(6-7)^2} = \sqrt{9^2+(-1)^2} = \sqrt{81+1} = \sqrt{82} = 9.06$
- Jarak ke Opsi C: $\sqrt{(8-4)^2+(8-3)^2} = \sqrt{4^2+5^2} = \sqrt{16+25} = \sqrt{41} = 6.40$

Akan ditemukan bahwa jarak *Euclidean* tipe A terdekat dengan nilai 6.32 jadi karakter yang dimainkan akan diubah ke karakter tipe A.

3. Pembuatan *UI* (*User Interface*)

Pembuatan *UI* merupakan salah satu faktor penting dalam pembuatan *video game,* karena *UI* memberikan kontrol kepada pemain dalam dunia *game.* Jadi, dianjurkan untuk membuat suatu *UI* yang mudah dipahami dan dikendalikan pemain.

4. Penulisan *Script*

Sebuah *video game* akan membutuhkan s*cript* yang sesuai dikarenkan s*cript* akan berfungsi sebagai pengatur perilaku, interaksi, dan berbagai macam elemen lain dalam *video game.*

2.2.3 Implementasi Sistem

Pada tahap ini, akan dimulai proses pengabungan dan pembangunan dari segala materi yang telah dibuat dan dikumpulkan pada tahap-tahap sebelumnya. Segala macam a*ssets,* rancangan *gameplay*, dan *UI* akan digambungankan untuk menjadi suatu *game*.

2.2.4 Pengujian

Pada tahap ini, *video game* yang telah dibuat akan diberikan kepada seorang pemain yang disebut "*beta tester*". Tugas dari *beta tester* ini adalah untuk melihat dan mencoba segala fitur yang dimiliki g*ame*, mencari apa ada *bug* atau e*xploit* pada *game*, dan memberikan kritik serta masukan tentang pengalaman saat bermain (Bécares et al., 2017). Tapi, inti utama dari tahap ini adalah untuk melihat apakah pemain akan merasa puas memainkan g*ame* yang telah dibuat.