PERANCANGAN PERILAKU *NON-PLAYABLE CHARACTER* PADA *GAME* "THE MA'HAD" MENGGUNAKAN METODE *BEHAVIOUR TREE* BERBASIS *DECISION SUPPORT SYSTEM*

SKRIPSI

Oleh :
FARHAN RAFIF AZZUFAR
NIM. 19650108

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023

PERANCANGAN PERILAKU *NON-PLAYABLE CHARACTER* PADA *GAME* "THE MA'HAD" MENGGUNAKAN METODE *BEHAVIOUR TREE* BERBASIS *DECISION SUPPORT SYSTEM*

SKRIPSI

Oleh :
FARHAN RAFIF AZZUFAR
NIM. 19650108

Diajukan kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2023

HALAMAN PERSETUJUAN

PERANCANGAN PERILAKU *NON-PLAYABLE CHARACTER* PADA *GAME* "THE MA'HAD" MENGGUNAKAN METODE *BEHAVIOUR TREE* BERBASIS *DECISION SUPPORT SYSTEM*

SKRIPSI

Oleh :
FARHAN RAFIF AZZUFAR
NIM. 19650108

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 23 September 2023

Pembimbing I,	Pembimbing II,
Dr. Yunifa Miftachul Arif, M.T	Hani Nurhayati, M.T
NIP. 19830616 201101 1 004	NIP. 19780625 200801 2 006

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan, M.MT, IPM
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

PERANCANGAN PERILAKU *NON-PLAYABLE CHARACTER* PADA *GAME* "THE MA'HAD" MENGGUNAKAN METODE *BEHAVIOUR TREE* BERBASIS *DECISION SUPPORT SYSTEM*

SKRIPSI

Oleh :
FARHAN RAFIF AZZUFAR
NIM. 19650108

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 23 November 2023

Susunan Dewan Penguji

Ketua Penguji	: Dr. M. Faisal, M.T NIP. 19740510 200501 1 007	( )
Anggota Penguji I	: Agung Teguh Wibowo Almais, M.T NIDT. 19860103 20180201 1 235	( )
Anggota Penguji II	: Dr. Yunifa Miftachul Arif, M.T NIP. 19830616 201101 1 004	( )
Anggota Penguji III	: Hani Nurhayati, M.T NIP. 19780625 200801 2 006	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan, M.MT, IPM
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Farhan Rafif Azzufar
NIM	: 19650108
Fakultas / Jurusan	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Perancangan Perilaku *Non-Playable Character* Pada *Game* "THE MA'HAD" Menggunakan Metode *Behaviour Tree* Berbasis *Decision Support System*

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 23 November 2023
Yang membuat pernyataan,

Farhan Rafif Azzufar
NIM.19650108

MOTTO

... Tidak ada yang tidak bisa dilakukan, anda hanya belum bisa ...

HALAMAN PERSEMBAHAN

Saya bersyukur dan berterima kasih kepada Allah SWT karena telah memberikan rahmat dan petunjuk-Nya, sehingga saya berhasil menyelesaikan skripsi ini. Skripsi ini saya persembahkan untuk kedua orang tua saya, yang selalu memberikan dukungan dan mendoakan kelancaran penulisan skripsi saya. Ibu Choirunnisa yang selalu memberikan doa, dukungan, perhatian, kasih sayang setiap waktu. Bapak Akhmad Akbar yaang selalu memberikan arahan, motivasi, serta mejadi pedoman saya sebagai laki-laki dalam mengambil setiap pilihan serta tanggung jawab, serta menanamkan prinsip-prinsip yang wajib dimiliki seorang laki-laki, anak, dan seorang contoh kakak yang baik bagi adik-adik saya. Kepada kedua adik saya Fariz Rizky Alfarizy dan Fairuz Nazhif Akbar, yang telah menjadi adik, teman bermain saat dirumah, sertam memberikan banyak dukungan dan doa kepada saya sehingga dapat menyelesaikan skripsi ini.

Terima kasih juga kepada Keluarga besar, dosen, para sahabat dan semua pihak yang telah memberikan saran, bantuan, motivasi, semangat serta dukungan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini. Terima kasih sedalam-dalam nya saya ucapkan bagi segala dugungan yang telah diberikan.

KATA PENGANTAR

Assalamu'alaikum Wr.Wb.

Puji syukur penulis panjatkan kepada Allah SWT yang senantiasa memberikan rahmat serta kesehatan, sehingga penulis mampu menyelesaikan penelitian ini dengan baik. Penulis menyampaikan ucapan terima kasih kepada semua pihak yang pernah terlibat langsung maupun tidak langsung dalam menyelesaikan penelitian ini, bukan hanya karena usaha keras dari penulis sendiri, akan tetapi karena adanya dukungan dari berbagai pihak. Oleh karena itu penulis berterima kasih kepada:

1. Prof. Dr. M. Zainuddin, M.A., selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Prof. Dr. Hj. Sri Harini, M.Si., selaku dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Fachrul Kurniawan M.MT., IPM selaku Ketua Prodi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang
4. Dr. Yunifa Miftachul Arif, M.T selaku Dosen Pembimbing 1 yang telah membimbing serta memberikan arahan serta motivasi dalam penulisan skripsi dari awal hingga akhir.
5. Hani Nurhayati, M.T selaku Dosen Pembimbing 2 yang telah memberikan bimbingan, arahan serta bantuan dalam terwujudnya karya tulis skripsi ini dari awal hingga akhir.

6. Dr. M. Faisal, M.T selaku penguji I dan Agung Teguh Wibowo Almais, M.T selaku penguji II yang telah meluangkan waktunya untuk menguji dan dengan sabar memberi arahan dan saran dalam menyelesaikan skripsi ini.
7. Segenap civitas akademik Program Studi Teknik Informatika, dan seluruh dosen yang telah memberikan ilmu serta arahan semasa kuliah.
8. Kedua orang tua penulis, Bapak Akhmad Akbar dan Ibu Choirunnisa, serta keluarga besar yang telah memberikan banyak dukungan, doa serta selalu menjadi semangat sehingga penulis mampu menyelesaikan masa studi hingga mencapai gelar sarjana.
9. Kedua adik penulis, Fariz Rizky Alfarizy dan Fairuz Nazhif Akbar, yang telah memberikan dukungan, menjadi teman bermain saat dirumah, doa, serta dukungan yang selalu menjadi semangat sehingga penulis dapat menyelesaikan masa studi hingga mencapai gelar sarjana.
10. Teman-teman Alliance of Informatics Engineering yang telah memberikan semangat dan juga dukungan kepada penulis
11. Teman-teman (HHTK) dan Rafikir Boys yang telah mendukung, memberikan bantuin, menjadi teman nongkrong dan bertukar pikiran sehingga penulis dapat menyelesaikan skripsi dengan baik.
12. Teman-teman yang telah memberikan bantuan yang sangat besar dalam pengumpulan dataset penelitian saya, serta yang menjadi teman diskusi dan teman belajar.
13. Terakhir penulis ingin berterima kasih kepada diri sendiri karena telah menyelesaikan skripsi dengan baik, terima kasih karena sudah berhasil

merealisasikan ide, merancang, hingga membangun game yang menjadi sumber penelitian serta penulisan skripsi sesuai berdasarkan apa yang telah diimajinasikan. Terimakasih karena telah berhasil melalui segala masa-masa sulit tanpa pernah berhenti berusaha. Dan terakhir terima kasih karena telah menjadi diri sendiri hingga akhir.

Skripsi yang telah ditulis ini masih jauh dari kata sempurna, oleh karena itu penulis sangat menghargai dan senang jika terdapat kritik dan saran yang diberikan. Semoga skripsi ini dapat memberikan manfaat.

Wassalamu alaikum, Wr. Wb.

Malang, 23 November 2023

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Azzufar, Farhan Rafif. 2023. **Perancangan Perilaku *Non-Playable Character* pada *Game* "THE MA'HAD" Menggunakan Metode *Behaviour Tree* Berbasis *Decision Support System***. Skripsi. Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (1) Dr. Yunifa Miftachul Arif, M.T (2) Hani Nurhayati, M.T

Kata Kunci *: Behavior Tree, Decision Support System, Non-Playable Character, Perancangan Perilaku,*

Non_Playable Character merupakan karakter pendukung yang dirancang untuk menambah kesan realistis di dalam permainan. Terdapat dua jenis NPC yaitu bersifat statis dan dinamis. NPC yang bersifat dinamis dirancang untuk membantu pemain menyelesaikan alur maupun tugas yang diberikan dalam permainan. Sebaliknya, NPC yang bersifat statis dirancang hanya untuk memberikan kesan nyata pada permainan. Penampilan kesan realistis pada NPC yang bersifat statis terkadang kurang tersampaikan kepada pamain, maka dari itu di dalam permainan dibutuhkan NPC yang bersifat dinamis. *Behaviour Tree* merupakan metode yang umum digunakan dalam perancangan perilaku pada suatu objek. Metode ini menggunakan tingkatan prioritas dalam mengeksekusi perilaku yang diinisiasi. *Decision Support System* merupakan metode yang digunakan untuk menentukan alternatif terbaik berdasarkan kriteria yang telah ditetapkan sebagai parameter pengambilan data. Perilaku NPC di dalam permainan "THE MA'HAD" dirancang untuk memberikan respon terhadap pemain berdasarkan data yang didapatkan dari serangkaian tes yang dilakukan dalam permainan. Berdasarkan pengujian yang dilakukan menggunakan 50 data dengan input yang berbeda, metode *Behaviour Tree* dan *Decision Support System* yang diterapkan dalam perancangan perilaku NPC menghasilkan perilaku yang sesuai dengan yang telah ditetapkan secara akurat, sehingga memberikan kesan yang lebih dinamis dan realistis.

ABSTRACT

Azzufar, Farhan Rafif. 2023. **Designing Non-Playable Character Behaviour in Game "THE MA'HAD" Using Behaviour Tree Method Based on Decision Support System**. Thesis. Department of Informatics Engineering, Faculty of Science and Technology, Maulana Malik Ibrahim Islamic State University Malang. Supervisor : (1) Dr. Yunifa Miftachul Arif, M.T (2) Hani Nurhayati, M.T

A Non-Playable Character (NPC) is a supporting character designed to enhance a realistic impression within a game. There are two types of NPCs: static and dynamic. Dynamic NPCs are designed to assist players in completing the storyline or tasks given in the game. Conversely, static NPCs are designed solely to create a lifelike atmosphere in the game. The realistic impression conveyed by static NPCs is sometimes not effectively conveyed to the player; therefore, dynamic NPCs are needed in the game. A Behavior Tree is a method commonly used in designing the behavior of an object. This method employs priority levels in executing initiated behaviors. The Decision Support System is a method used to determine the best alternatives based on predefined criteria as data retrieval parameters. The behavior of NPCs in the game "THE MA'HAD" is designed to respond to players based on data obtained from a series of tests conducted in the game. Based on testing with 50 different input data, the Behavior Tree and Decision Support System methods applied in designing NPC behavior produce behavior that accurately aligns with the predetermined criteria, thus providing a more dynamic and realistic impression.

Keywords: *Behavior Design, Non-Playable Character, Behaviour Tree, Decision Support System*

مستخلص البحث

الزفار، فرحان رافيف. 2023. تصميم سلوك الشخصية غير القابلة للعب في لعبة THE MA'HAD " باستخدام طريقة شجرة السلوك على أساس نظام دعم القرار. البحث الجامعي. قسم الهندسة المعلوماتية، كلية العلوم والتكنولوجيا بجامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف الأول: د. يونيفة مفتاح العارف، الماجستير. المشرف الثاني: هاني نورحياتي، الماجستير.

الكلمات الرئيسية: تصميم السلوك، الشخصية غير القابلة للعب، شجرة السلوك، نظام دعم القرار.

الشخصية غير القابلة للعب هي شخصية داعمة مصممة لإضافة انطباع واقعي إلى اللعبة. هناك نوعان من الشخصية غير القابلة للعب، ثابتة وديناميكية. تم تصميم الشخصية غير القابلة للعب الديناميكية لمساعدة اللاعبين على إكمال التدفق والمهام الواردة في اللعبة. في المقابل، تم تصميم الشخصية غير القابلة للعب الثابتة فقط لإعطاء اللعبة إحساسا حقيقيا. في بعض الأحيان لا يتم نقل الانطباع الواقعي عن الشخصية غير القابلة للعب الثابتة إلى اللاعبين، لذلك هناك حاجة إلى الشخصية غير القابلة للعب الديناميكية في اللعبة. شجرة السلوك هي طريقة شائعة الاستخدام في تصميم السلوك على كائن. تستخدم هذه الطريقة مستويات الأولوية في تنفيذ السلوك الذي تم بدؤه. نظام دعم القرار هو طريقة تستخدم لتحديد أفضل بديل بناء على المعايير التي تم تعيينها كمعلمات استرجاع البيانات. تم تصميم سلوك NPC في "THE MA'HAD " للاستجابة للاعبين بناء على البيانات التي تم الحصول عليها من سلسلة من الاختبارات التي أجريت في اللعبة. استنادا إلى الاختبارات التي أجريت باستخدام 50 بيانات بمدخلات مختلفة، فإن طريقة شجرة السلوك ونظام دعم القرار المطبقة في تصميم سلوك NPC تنتج سلوكا يطابق ما تم تعيينه بدقة، مما يعطي انطباعا أكثر ديناميكية وواقعية.

Penerjemah, M.Mubasysyir Munir, MA NIDT:19860513201802011215	Tanggal 13/11/2023	Validasi Kepala PPB, Prof. Dr. H. M. Abdul Hamid, MA NIP: 19730201 1998031007

BAB I

PENDAHULUAN

1.1 Latar Belakang

Permainan atau *Game* merupakan sebuah sarana atau media yang biasanya digunakan untuk menghilangkan rasa penat maupun stress dari kesibukan sehari-hari yang dialami. Permainan biasa dimainkan selepas melakukan aktifitas yang cukup menguras tenaga maupun pikiran. Pada dasarnya seseorang bermain *game* dengan tujuan untuk mendapatkan kesenangan maupun kepuasan diri, sehingga rasa lelah setelah beraktifitas menjadi hilang. Tidak hanya itu, apabila ditinjau dari sisi lain *game* memiliki banyak peran positif lainnya, *game* dapat memotivasi *player* untuk ikut serta dalam pembelajaran yang terdapat di dalam konten permainan tersebut (Arif et al., 2021). Pembelajaran yang dapat diambil dari sebuah permainan seperti, media untuk mengasah otak, bagaimana mengambil keputusan yang tepat disaat tertentu dan lain sebagainya.

Menurut Stenros (2017) dalam penelitiannya, definisi yang saat ini diterima secara mayoritas berdasarkan apa yang telah diusulkan oleh Clark C. Abt, yaitu: "setiap kontes (permainan) di antara musuh (pemain) yang beroperasi di bawah kendali (aturan) untuk suatu tujuan (kemenangan, pembayaran kemenangan)". Sejalan dengan definisi tersebut dapat dikatakan bahwa sebuah *game* memiliki konsep dimana seseorang atau lebih yang biasa disebut dengan istilah *player*, menjalankan atau mengikuti aturan untuk mencapai suatu target yang telah ditetapkan, aturan tersebut berperan sebagai petunjuk serta menjadi alur agar *player* tetap mengikuti konsep permainan tersebut.

Dalam *game* terdapat salah satu unsur pendukung yang sangat berperan dalam meningkatkan kualitas serta *value* dari permainan yaitu *non-playable character* atau yang biasa disebut dengan NPC. NPC berperan penting dalam *game* sebagai objek *humanoid* (objek dalam bentuk manusia). Implementasi NPC ke dalam sebuah *game* biasanya untuk menambahkan kesan realistis sekaligus menghidupkan sebuah permainan agar terlihat lebih menarik dan tidak membosankan.

Menurut Warpefelt (2016) di dalam buku nya, *non-player character* atau NPC merupakan karakter yang terdapat dalam permainan komputer yang dikontrol oleh komputer, bukan dengan *player*. *Non-Playable Character* (NPC) merupakan objek yang bersifat dinamis yang tidak berada dibawah kontrol oleh *player,* dan beroperasi dalam sebuah wilayah pada *game* secara mandiri (Ibrahim, 2020).

Dengan adanya NPC, sebuah *game* akan terlihat seperti kehidupan pada umumnya yang dimana terdapat banyak manusia. Menurut Mackay dalam (Warpefelt, 2016), NPC biasanya ditemukan di dalam permainan, contohnya pada *role playing games (*RPGs*),* dimana karakter nya dikontrol oleh *Dungeon Master* (DM). Di dalam *game* sendiri pada umumnya NPC terbagi menjadi dua jenis, yaitu NPC yang hanya berperan sebagai pelengkap, yang dimana *humanoid object* tersebut hanya ditempatkan di dalam sebuah *game* hanya untuk melengkapi *environment* dengan tujuan memberikan kesan hidup pada permainan dan tidak melakukan *state* atau perilaku apapun.

Selanjutnya, terdapat juga NPC yang memang ditempatkan dalam *game* dengan tujuan untuk berinteraksi bahkan sebagai pemandu utama dalam

menjalankan *game* tersebut, NPC tipe ini biasanya dapat melakukan interaksi dengan *player* serta dapat melakukan beberapa *state* seperti bergerak sesuai kondisi yang sudah ditetapkan, memberikan arahan kepada *player* dalam menjalankan permainan, hingga menemani *player* sampai akhir permainan. Implementasi NPC yang bersifat kompleks ini, membutuhkan analisis yang kompleks pula dalam menginisiasikannya, kompleksitas nya berbanding lurus seiring dengan kerumitan dalam perancangannya, namun dampak nya tentunya lebih besar dan lebih berpengaruh di dalam *game* tersebut.

Dalam penelitiannya Fink et al. (2007) berpendapat bahwa mendesain NPC merupakan hal yang sulit, karena pemain *game* atau *player* menginginkan NPC semirip mungkin dengan manusia, menantang atau membantu *player* layaknya manusia pada umumnya. Perancangan *Non-Playable Character* (NPC) yang memiliki perilaku yang lebih dari satu *state* dapat dilakukan pada tahap inisiasi awal dengan cara memberikan beberapa kondisi serta kemungkinan berdasarkan pilihan *player* di dalam *game.* Guna menentukan pemilihan *state* terbaik atau yang paling memungkinkan, dapat menggunakan metode *behaviour tree,* dimana dengan menginisiasikan beberapa *state* pada tahap awal perancangan lalu memberikan pengkondisian yang bergantung kepada pilihan yang dipilih oleh *player* di dalam *game.*

Menurut Trianto et al. (2017) terkait penelitiannya dalam merancang agen cerdas, *behavior tree* adalah salah satu metode pemetaan perilaku agen cerdas dasar yang menggunakan task sebagai elemen penyusunnya. Dengan menggunakan metode ini maka NPC akan memiliki beberapa *state* cadangan yang berbeda yang

nantinya *state* tersebut akan dipilih berdasarkan *priority level* yang sudah diinisiasikan sejak awal agar sesuai dengan perkembangan permainan.

Ditinjau dari sudut pandang *simple-game* pada umumnya, unsur NPC biasanya hanya terlihat statis atau tidak memiliki perilaku, keberadaan NPC ini hanya untuk menampilkan keberadaan objek manusia lain yang terdapat di dalam *game*. Hal ini dapat dikatakan NPC hanya berperan sekedar menambah kesan realistis pada permainan, akan tetapi tidak membuatnya lebih hidup. Menurut penelitian Sugianto & Utama (2021) mengatakan, Pada *Genre Action*, setiap tokoh yang tidak dimainkan (*Non Player Character*/NPC) dapat melakukan sesuatu agar memiliki kesan hidup. Dalam hal ini diperlukan konsep Kecerdasan Buatan (*Artificial Intelligence*), maka di dalam permainan dibutuhkan NPC yang memiliki sebuah perilaku yang berperan layaknya manusia pada umumnya.

Maka dari itu dibutuhkan inisiasi *state* pada NPC yang dikombinasikan dengan *Decision Support System* (DSS) untuk menghasilkan *state* yang akan dijalankan secara *realtime* sesuai dengan kondisi maupun perkembangan di dalam *game* tersebut. Penggunaan DSS akan sangat membantu dalam penentuan tingkatan prioritas *state* yang akan dijalankan terlebih dahulu agar sesuai dengan data dari *game,* hal ini memungkinkan *player* memiliki banyak pengalaman yang berbeda sesuai dengan pilihan yang dipilih di dalam *game* tersebut. Untuk menganalisis tingkat kepentingan tersebut maka digunakan metode *Decision Support System* (DSS) untuk menentukan tingkat kepentingan atau melakukan perankingan terhadap banyaknya *state* yang ada.

Menurut Anto et al. (2019) dalam penelitiannya, sistem pendukung keputusan (SPK) adalah sistem yang di dalamnya terdapat alternatif, kriteria dan bobot yang digunakan untuk menentukan solusi terbaik. *Decision Support System* (DSS) atau dalam bahasa indonesia sistem pendukung keputusan (SPK), merupakan suatu sistem pembantu pemilihan keputusan berdasarkan tingkat kepentingan tertinggi yang didapatkan dari analisis data yang sebelumnya sudah dikumpulkan. Data tersebut akan dianalis dan dilakukan perankingan berdasarkan tingkat kepentingan tertinggi yang nantinya akan menghasilkan suatu keputusan yang paling akurat berdasarkan kondisi yang sesuai dengan data dalam bentuk solusi alternatif. *Decision support system* memiliki beberapa metode di dalamnya yang berisi perbedaan model analisis data, namun memiliki tujuan yang sama yaitu untuk menghasilkan suatu keputusan yang paling sesuai dan krusial terhadap kondisi yang dialami. Salah satu metode yang terdapat dalam DSS yang digunakan untuk melakukan perankingan serta menghasilkan keputusan terbaik yaitu *Simple additive Weighting* (SAW). Metode ini memiliki sistem analisis yang menggunakan kriteria berdasarkan data yang telah ditentukan, menginisiasikan bobot serta alternatif, hingga mendapatkan alternatif solusi yang dapat diaplikasikan ke dalam berbagai permasalahan.

Menurut Fauzan et al. (2018), metode *Simple Additive Weighting* (SAW) adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua kriteria. Metode ini menganalisis sebuah data yang sudah didapatkan kemudian menghitung tingkat kepentingannya berdasarkan *value* dari tiap-tiap alternatif yang ada. Setelah mendapatkan hasil dari analisis perhitungan tiap

alternatif maka akan dilakukan perankingan pada tiap alternatif berdasar pada nilai tertinggi. Setelah dilakukan perankingan maka akan didapatkan suatu kesimpulan yang berupa alternatif atau solusi yang dapat digunakan dalam kondisi tersebut. Pemilihan penggunaan metode *Simple Additive Weighting* (SAW) dalam pengimplementasian *decision support system* pada penelitian kali ini dikarenakan metode SAW merupakan salah satu metode yang paling mudah serta dengan tahapan nya yang tidak terlalu panjang dan rumit sampai pada tahap mendapatkan keputusan terbaik yang dibutuhkan yang berupa alternatif.

Terkait hubungan antara *behaviour tree* dengan *decision support system* model *simple additive weighting,* metode SAW digunakan untuk menentukan alternatif yang sesuai berdasarkan data yang telah didapatkan dari *player* setelah menyelesaikan serangkaian *pretest.* Setelah mendapatkan alternatif yang sesuai, maka akan ditentukan tingkat prioritas perilaku yang telah diinisiasikan menggunakan metode *behaviour tree.* Pada dasarnya *behaviour tree* merupakan metode pengeksekusian *state* berdasarkan tingkat prioritas yang telah diinisiasikan, maka dari itu setelah mendapatkan alternatif yang sesuai dari hasil perhitungan menggunakan metode *simple additive weighting* maka akan disesuaikan tingkat prioritas perilaku yang akan dijalankan dengan metode *behaviour tree.* Penggunaan dua tahap penentuan ini dilakukan agar *output* perilaku NPC yang akan dijalankan mendapatkan hasil yang sesuai dengan kondisi data *player* di dalam *game* tersebut.

Data yang diambil merupakan data perkembangan *player* di dalam game, data yang telah tersimpan pada *player* akan di ambil atau dibaca oleh NPC dan dilakukan analisis guna menentukan perilaku yang sesuai yang nantinya akan

dijalankan terlebih dahulu dalam bentuk *state.* Tiap perbedaan data perkembangan pada *player* memiliki kemungkinan *output* perilaku yang berbeda pada NPC. Dengan hal ini maka *player* atau setiap user yang memainkan game ini akan merasakan pengalaman yang berbeda-beda bergantung pada pilihan mereka masing-masing. Implementasi DSS ke dalam ini juga memberikan pengalaman yang lebih *realtime* terhadap penentuan kondisi yang akan dipilih *player* di dalam *game.*

Dalam penelitian ini, perancangan *game* "THE MA'HAD" bertujuan untuk memberikan gambaran simulasi kehidupan di dalam mahad. Berdasarkan pengertiannya simulasi adalah suatu proses peniruan terhadap hal yang nyata beserta keadaan disekitarnya (dengan suatu keadaan) Faisal et al., (2016). Maksud dari simulasi kehidupan di ma'had yaitu agar mahasiswa lebih memahami sekaligus mengetahui kehidupan serta pembelajaran yang terdapat di dalam ma'had itu sendiri. Alasan penggunaan media *game* sebagai sarana pembelajaran dikarenakan *game* adalah suatu bentuk aplikasi yang paling banyak digunakan dan dinikmati oleh pengguna media elektronik Nurhayati et al., (2017).

Selain memberikan gambaran terkait kehidupan ma'had, *game* ini juga memberikan pengetahuan terkait pembelajaran agama islam yang patut diketahui oleh setiap umat muslim dalam bentuk media yang baru yang diharapkan dapat memberikan kesan berbeda dari yang lain. Di dalam islam sendiri telah di jelaskan dalam (*Q.S at-Taubah: 122*), terkait pentingnya menuntut ilmu. Berikut firman Allah SWT. dalam (*Q.S at-Taubah: 122*) :

وَمَا كَانَ ٱلْمُؤْمِنُونَ لِيَنفِرُواْ كَآفَّةً ۚ فَلَوْلَا نَفَرَ مِن كُلِّ فِرْقَةٍ مِّنْهُمْ طَآئِفَةٌ لِّيَتَفَقَّهُواْ فِى ٱلدِّينِ وَلِيُنذِرُواْ قَوْمَهُمْ إِذَا رَجَعُوٓاْ إِلَيْهِمْ لَعَلَّهُمْ يَحْذَرُونَ

"Tidak sepatutnya bagi mukminin itu pergi semuanya (ke medan perang). Mengapa tidak pergi dari tiap-tiap golongan di antara mereka beberapa orang untuk memperdalam pengetahuan mereka tentang agama dan untuk memberi peringatan kepada kaumnya apabila mereka telah kembali kepadanya, supaya mereka itu dapat menjaga dirinya."

Berdasarkan penafsiran dari Ibnu Katsir yang merupakan hafidz, ulama sekaligus pemikir, menafsirkan ayat diatas sebagai berikut. Tatkala kaum Mukminin dicela oleh Allah bila tidak ikut ke medan perang kemudian Nabi ﷺ mengirimkan sariyahnya, akhirnya mereka berangkat ke medan perang semua tanpa ada seorang pun yang tinggal, maka turunlah firman-Nya berikut ini: (Tidak sepatutnya bagi orang-orang yang mukmin itu pergi) ke medan perang (semuanya. Mengapa tidak) (pergi dari tiap-tiap golongan) suatu kabilah (di antara mereka beberapa orang) beberapa golongan saja kemudian sisanya tetap tinggal di tempat (untuk memperdalam pengetahuan mereka) yakni tetap tinggal di tempat (mengenai agama dan untuk memberi peringatan kepada kaumnya apabila mereka telah kembali kepadanya) dari medan perang, yaitu dengan mengajarkan kepada mereka hukum-hukum agama yang telah dipelajarinya (supaya mereka itu dapat menjaga dirinya) dari siksaan Allah, yaitu dengan melaksanakan perintah-Nya dan menjauhi larangan-Nya (*Tafsir Ibnu Katsir 4.2.Pdf*, n.d.). Sehubungan dengan ayat ini Ibnu Abbas r.a. memberikan penakwilannya bahwa ayat ini penerapannya hanya khusus untuk sariyah-sariyah, yakni bilamana pasukan itu dalam bentuk sariyah lantaran Nabi ﷺ tidak ikut. Sedangkan ayat sebelumnya yang juga melarang seseorang tetap

tinggal di tempatnya dan tidak ikut berangkat ke medan perang, maka hal ini pengertiannya tertuju kepada bila Nabi ﷺ berangkat ke suatu ghazwah.

1.2 Pernyataan Masalah

Berdasarkan latar belakang masalah yang telah dijelaskan sebelumnya, terdapat sebuah masalah yang diangkat dalam penelitian kali ini, yakni:

1. Bagaimana implementasi metode *Decision Support System* dengan model *Simple Additive Weighting* dalam menentukan perilaku *Non-Playable Character* pada *game* "THE MA'HAD"
2. Bagaimana tingkat akurasi perubahan perilaku *Non-Playable Character* terhadap perkembangan *status player* pada *game* "THE MA'HAD".

1.3 Tujuan Penelitian

Tujuan dari penelitian ini sesuai dengan identifikasi masalah yang telah dijelaskan, yakni: untuk

1. Mengetahui implementasi metode *Decision Support System* dengan model *Simple Additive Weighting* dalam menentukan perilaku *Non-Playable Character* pada *game* "THE MA'HAD"
2. Mengetahui tingkat akurasi perubahan perilaku *Non-Playable Character* terhadap perkembangan *status player* pada *game* "THE MA'HAD".

1.4 Batasan Masalah

Dalam menjadikan penelitian yang dilakukan tetap fokus dan terarah, terdapat batasan-batasan masalah yang diterapkan sebagai berikut.

1. Penelitian akan menggunakan *Multi Criteria Decision Making* (MCDM) yaitu *Simple Additive Weighting* (SAW)
2. NPC yang digunakan dibatasi menjadi dua NPC yang berperan sebagai sistem objektif dalam *game.*

1.5 Manfaat Penelitian

Harapan dari penelitian yang dilakukan yakni dapat memberikan manfaat dan maslahat dikemudian hari antara lain sebagai berikut.

1. Memberikan gambaran terkait lingkungan serta kegiatan di dalam Ma'had Sunan Ampel 'Al-Aly, serta memberikan gambaran serta pengajaran terkait materi yang terdapat di dalam *game* "THE MA'HAD".
2. Hasil penelitian ini dapat dijadikan sebagai bahan acuan lebih lanjut untuk penelitian selanjutnya mengenai perancangan *Non-Playable Character* - menggunakan metode *Behaviour Tree* berbasis *Simple Additive Weighting.*

BAB II

STUDI PUSTAKA

2.1 Penelitian Terdahulu

Pada panelitian Pratiwi et al., (2021) yang berjudul Membangun Game 3D *Side Scroll* dan Menerapkan Model *Behaviour Tree* pada NPC *Enemy* dalam game "MAVERICK". Tujuan dari penelitian ini adalah menerapkan metode *behaviour tree* pada NPC agar lebih adaptif terhadap perilaku pemain dalam permainan ini. Pengujian metode *behaviour tree* pada penelitian ini akan diimplementasikan pada *autonom agent* sebagai pengendali perilakunya, perubahan perilaku NPC berdasarkan kondisi yang sedang terjadi dalam *game.* Berdasarkan hasil pengujian, dapat ditarik kesimpulan bahwa penggunaan *behaviour tree* dalam agen cerdas dapat menghasilkan perilaku yang adaptif terhadap pemain.

Pada penelitian Hogg et al (2020) dengan judul *Evolving Behaviour Trees For Supervisory Control of Robot Swarms,* bertujuan untuk menghasilkan kawanan (*swarms*) terbaik yang dapat memahami informasi yang diberikan oleh manusia yang berperan sebagai operator. Selain itu penelitian ini juga bertujuan untuk menghasilkan *swarms* yang memiliki efektifitas tertinggi dalam hal kecepatan dalam menemukan solusi ketika dihadapkan dengan permasalahan terbaru. Berdasarkan permasalahan yang kerap terjadi terkait dengan penelitian yang serupa, dinyatakan bahwa objek percobaan (*swarms*) masih terdapat ketidakmampuan *swarms* untuk menerima informasi atau perintah dari operator, sehingga dibutuhkan perubahan pada algoritma, parameter, *environment,* serta kontrol pada agen. Pada penelitian ini dilakukan implementasi metode *behaviour*

tree dalam menginisiasikan perilaku-perilaku pada *swarms* agar dapat melakukan tugas yang diberikan oleh operator. Penggunaan metode *behaviour tree* dalam menginisiasikan perilaku terhadap *swarms* bertujuan agar operator dapat kontrol terhadap *swarms* dalam berbagai kondisi dan scenario. Merujuk berdasarkan implementasi metode *behaviour tree* terhadap *swarms*, operator berhasil melakukan kontrol terhadap perilaku *swarms*. Hasil dari penelitian ini menyatakan bahwa peneliti berhasil menghasilkan strategi yang dikembangkan berdasarkan *environment* sistematis dan spesialisasi dari berbagai skenario. Dalam kasus ini peneliti dapat mengembangkan kondisi yang lebih luas yang akan digunakan dalam mengontrol *swarms* dan menggali lebih dalam strategi yang berpengaruh sesuai yang dikembangkan.

Pada penelitian Alamsyah et al., (2019) tentang Implementasi Algoritma *Collision Detection* dan *Finite State Machine* Untuk Karakter Musuh Pada Game Bertipe MetroidVania. Penelitian ini bertujuan untuk merancang karakter musuh yang adaptif berdasarkan perkembangan game dan level dari permainan. Penggunaan metode *collision detection* digunakan untuk pengecekan dua atau lebih objek yang saling bertumpukan, dalam metode ini terdapat dua tipe yaitu *priori detection* dan *post detection* yang dimana *priori detection* adalah pengecekan sebelum tumpukan terjadi, dan *post detection* adalah pengecekan setelah terjadinya tumpukan. Metode ini diterapkan pada musuh sebagai pendeteksian *player* dalam jangkauan musuh yang telah ditetapkan, setelah *player* berada dalam jangkauan musuh maka akan digunakan metode *finite state machine* (FSM) guna menginisiasikan perilaku atau tindakan musuh tersebut terhadap *player*.

Berdasarkan pengujian serta pengimplementasian metode *collision detection* dan *finite state machine* pada karakter musuh dalam game, dapat dikatakan telah berhasil menghasilkan perilaku musuh yang realistis dan dapat menyesuaikan dengan keadaan yang terjadi.

Pada penelitian Chen (2021), yang berjudul Application of Analytic Hierarchy Process (AHP) and Simple Additive Weighting (SAW) Methods In Singer Selection Process. Permasalahan yang terdapat pada penelitian ini adalah dalam menemukan penyanyi yang tepat untuk posisi vokalis dalam sebuah band. Pada umumnya penilaian hanya akan memberikan komentar pada tiap peserta tanpa memberikan perangkingan yang pada akhirnya berakibat pada hasil akhir saat menentukan keputusan, karena permasalahan ini sering terjadi hasil yang tidak akurat dan tidak sesuai dengan ekspektasi yang berakibat pada penseleksian ulang. Berdasarkan permasalahan ini, dalam penelitian nya, digunakan *Multi Criteria Decision Making* (MCDM) menggunakan *Analytic Hierarchy Process (*AHP) dan *Simple Additive Weighting* (SAW). AHP digunakan untuk perhitungan bobot pada tiap kriteria, dan SAW digunakan untuk proses perangkingan. Berdasarkan hasil pengukuran tingkat akurasi dari penggunaan metode tersebut dan pertimbangan dari expert daam bidangnya, menghasilkan hasil yang bagus dengan tingkat akurasi mencapai 84.61%. Ketika dibandingkan dengan penelitian sebelumnya, kombinasi dari metode AHP dan SAW dapat meningkatkan akurasi dari hasil perangkingan alternatif.

Pada Penelitian Eka Sugiyarti et al. (2018) dalam penelitiannya tentang Decision Support System of Scholarship Grantee Selection using Data Mining,

International Journal of Pure and Applied Mathematics. Tujuan dari penelitian ini yaitu menentukan serta menyeleksi para calon kandidat yang akan menerima beasiswa untuk melanjutkan pendidikan. Permasalahan yang dihadapi ialah terdapat beberapa kriterai menjadi dasar penilaian terhadap calon penerima beasiswa tersebut. Dalam penelitian ini digunakan metode data mining dengan algoritma C4.5 yang digunakan untuk merancang pohon keputusan. Dasar penggunaan metode serta algoritma ini yaitu karena sifatnya yang kuat dan cukup terkenal dalam mentransformasi data ke dalam *decision tree* yang direpresentasikan ke dalam sebuah aturan, aturan ini dapat dengan mudah dipahami dengan menggunakan *natural language,* bahkan dapat dinyatakan secara langsung.

Pada penelitian Arif et al. (2021) dalam penelitiannya tentang An Automatic Scenario Control in Serious Game to Visualize Tourism Destinations Recommendation. Tujuan dari penelitian ini yaitu untuk memvisualisasikan aktivitas kunjungan pada setiap destinasi kunjungan. Permasalahan yang dihadapi pada penelitian ini yaitu, terkadang pengguna menginginkan interaksi secara *realtime* saat memainkan *game.* Berdasarkan permasalahan yang dihadapi tersebut digunakanlah *Decision Support System* untuk menangani pemilihan. Metode yang digunakan merupakan *Multi Criteria Decision Making* (MCDM) yang merupakan salah satu dari DSS. Dari banyaknya teknik dalam MCDM. untuk mengurutkan pilihan berdasarkan kemiripan terhadap solusi ideal, digunakan TOPSIS yang merupakan salah satu teknik dengan komputasi yang sederhana Berdasarkan pengujian serta pengimplementasian metode, sistem ASC dapat otomatis memilih

skenario berdasarkan input rekomendasi destinasi wisata serta data dari ekspektasi pengguna.

Tabel 2.1 Perbandingan Penelitian

No	Peneliti	Judul	Persamaan	Perbedaan
1	(Pratiwi et al., 2021)	Memangun Game 3D *Side Scroll* Dan Menerapkan Model *Behaviour Tree* Pada NPC *Enemy* Dalam Game "Maverick"	Perancangan Perilaku NPC, *Behaviour Tree*	Game *Sidescroller*, Game 3D
2	(Hogg et al., 2020)	*Evolving behaviour trees for supervisory control of robot swarms*	Perancangan Perilaku, *Behaviour Tree*	Objek perancangan (*Swarms*)
3	(Alamsyah et al., 2019)	Implementasi Algoritma *Collision Detection* Dan *Finite State Machine* Untuk Karakter Musuh Pada Game Bertipe Metroidvania	Perancangan Perilaku NPC	Metode Perancangan *Collision Detection* Dan *Finite State Machine*, Game 2D
4	(Chen, 2021)	Application of Analytic Hierarchy Process (AHP) and Simple Additive Weighting (SAW) Methods In Singer Selection Process	*Decision Support System, Simple Additive Weighting*	Penggabungan *Analytic Hierarchy Process (AHP) and Simple Additive Weighting* (SAW) Methods, *Research Object*
5	(Eka Sugiyarti et al., 2018)	Decision Support System of Scholarship Grantee Selection using Data Mining, International Journal of Pure and Applied Mathematics	*Decision Support System*	*Data Mining Method*, C4.5 *Algorithm*
6	(Arif et al., 2021)	An Automatic Scenario Control in Serious Game to Visualize Tourism Destinations Recommendation	*Decision Support System, Multi criteria Decision Making*	Objek Penelitian (Destinasi Wisata)

Berdasarkan Tabel 2.1 persamaan dan perbedaan penelitian terdahulu dengan penelitian ini yang berjudul Perancangan Perilaku *Non-Player Character* Pada *Game* "The Ma'Had" Menggunakan Metode *Behaviour Tree* Berbasis *Simple Additive Weighting***.** Penelitian ini bertujuan merancang perilaku *non-player character* yang bersifat adaptif serta dinamis berdasarkan perkembangan dalam *game*. Metode yang digunakan dalam penelitian ini yaitu *behaviour tree* dan berbasis *decision support system* menggunakan model *simple additive weighting*, yang dimana perilaku atau tiap *state* pada NPC akan diinisiasikan dan akan dijalankan sesuai kondisi dalam *game*, kemudian penerapan *simple additive weighting* akan membantu meningkatkan akurasi *output* perilaku serta memberikan keputusan yang sesuai terhadap perilaku NPC di dalam *game*.

2.2 Pengertian Game

Menurut Suryadi (2018) dalam jurnal penelitiannya, *game* atau permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius dengan tujuan *refreshing*. Berdasarkan hal ini dapat dikatakan bahwa permainan sendiri merupakan suatu aturan yang dimainkan atau harus diikuti oleh seorang pemain atau lebih dengan tujuan untuk menyelesaikan permainan tersebut berdasarkan aturan yang telah diterapkan.

Berdasarkan penelitian Suryadi (2018) dinyatakan bahwa teori permainan pertama kali ditemukan oleh sekelompok ahli matematika pada tahun 1944. Teori itu dikemukakan oleh John von Neumann dan Oskar Morgenstern yang berisi: "Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing

dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan."

Permainan atau *game* sendiri memiliki beragam jenis atau gaya permainan itu sendiri yang menjadi karakteristiknya. Menurut Henry (2010:112) (Wulandari, 2015) format sebuah *game* bisa murni sebuah *genre* atau bisa merupakan campuran (*hybrid*) dari beberapa *genre* lain. Beberapa *Genre game* itu sendiri seperti *Action Games, Real Time Strategies* (RTS), *Role Playing Games* (RPG), *Real World Simulation, Construction and Management, Adventure Games, Side Scrolling Games,* dan masih banyak yang lainnya.

2.3 Non-Playable Character

Non-Playable Character (NPC) adalah sebuah karakter di dalam permainan yang biasanya ditemukan dalam bentuk manusia atau *humanoid.* NPC merupakan salah satu unsur penting yang berfungsi menambahkan kesan realistis dalam permainan. *Non-Playable Character* (NPC) adalah karakter yang terdapat di dalam permainan yang tidak berada di bawah kontrol *player* dan dioperasikan oleh mesin atau sistem dalam permainan itu sendiri (Fadila et al., 2023).

NPC terbagi menjadi dua yaitu, bersifat statis yang dimana hanya sebagai objek penambah realitas *game,* dan bersifat dinamis yang mana NPC tersebut memiliki perilaku tersendiri yang telah diinisiasikan oleh *developer* dengan berbagai tujuan dalam permainan. *Non-playable character* yang mampu bereaksi dengan aksi yang dilakukan pemain menjadi salah satu faktor agar permainan tetap dapat menarik bagi pemain (Pratiwi et al., 2021).

2.4 Unity 3D

Unity merupakan salah satu dari berbagai macam jenis *game engine* yang berperan dalam proses pengembangan atau pembuatan sebuah *game.* Dalam jurnal Šmíd (2017) mengatakan bahwa *Unity Technologies company* mengembangkan *Unity game engine* yang mana menjadi salah satu engine yang paling banyak digunakan. Versi pertama yang diluncurkan pada Apple Conference tahun 2005 dan ditargetkan hanya untuk pengembangan OS X. Hingga saat ini *Unity* telah berkembang dan sekarang mendukung 27 platform termasuk VR.

Hal penting yang menjadi landasan *Unity Engine* banyak digunakan ialah karena kemudahan dalam penggunaannya. Terdapat banyak keunggulan dari *Unity Game Engine* yang mana Pengembangan menggunakan *Unity* sangat cepat, khususnya pada platform mobile, Proyek serta pembentukan *game* yang kecil, proses eksport yang simple, komponen dan arsitektur nyam udah untuk dipahami, serta proses *scripting* dengan bahasa pemrograman C# sangat cepat dan efisien. Terdapat beberapa fitur yang tersedia di dalam *game engine* ini yang sangat berperan dalam pengembangan *game*.

2.5 Behaviour Tree

Behaviour Tree Merupakan sebuah pemodelan dalam pengeksekusian dan penyusunan transisi antara *task* satu dan yang lainnya secara matematis (Colledanchise & Ögren, 2017). *Behaviour tree* adalah sebuah model untuk pengeksekusian rencana yang secara grafis di representasikan sebagai sebuah pohon. *Behavoiur tree* memungkinkan untuk dapat mengontrol perilaku karakter pada *game* dan menjelaskan tingkatan keputusan dan aksi (Junaidi et al., 2021).

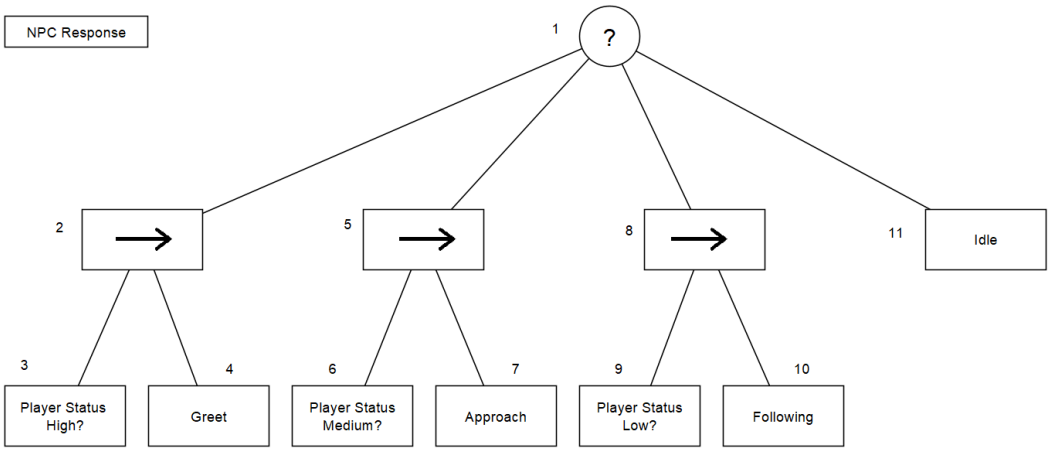
Untuk menerapkan alat teori kontrol pada analisis pohon perilaku, mereka dapat didefinisikan sebagai tiga-tupel.

$$T_i = \{f_i, r_i, \Delta_t\} \tag{2.1}$$

Dimana i ∈ N adalah index dari pohon,

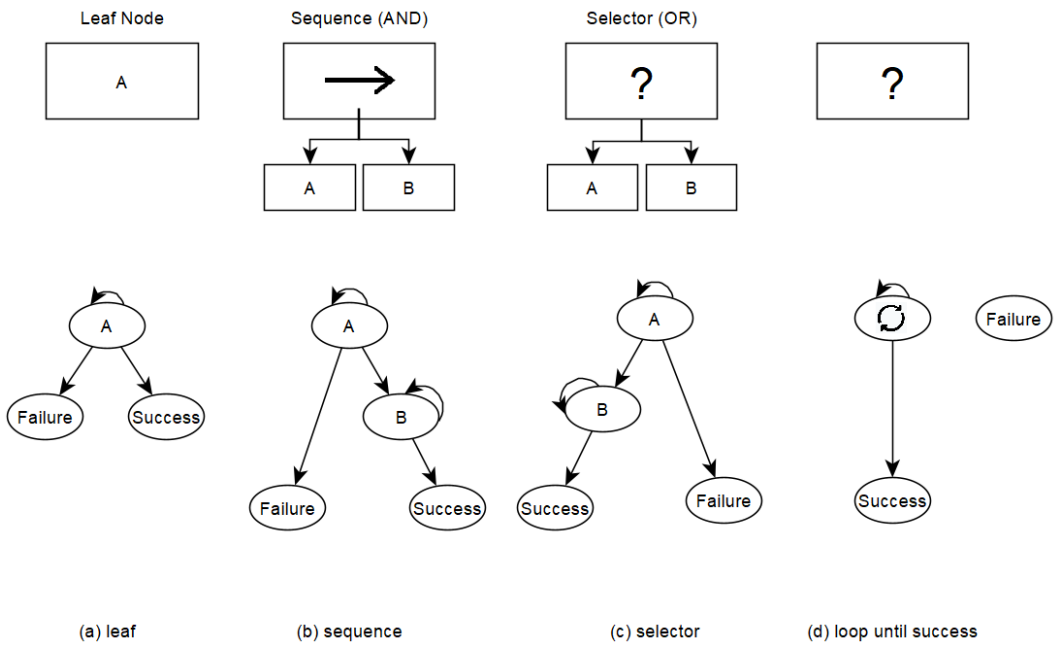
$$f_i = R^n \rightarrow R^n \tag{2.2}$$

Adalah bidang vektor yang mewakili ruas kanan persamaan beda biasa, Δ_t adalah langkah waktu dan $r_i : R^n \rightarrow \{R_i, S_i, F_i\}$ Adalah waktu pengembalian yang bisa sama dengan running R_i, Success S_i, atau Failure F_i.



Gambar 2.1 Contoh *Behaviour Tree* (Mcquillan, 2015).

Pada Gambar 2.1 terdapat contoh *Behaviour Tree*, Selama permainan berjalan, maka *tree* akan terus "*ticked*", yang artinya ia akan terus berjalan dari *root* melalui semua *child node* semakin kebawah, jika *node* memiliki *child* (Sekhavat, 2017).



Gambar 2.2 Komponen Pada *Behaviour Tree* (Mcquillan, 2015)

Berdasarkan Gambar 2.2 *Behaviour tree* memiliki berbagai macam elemen, tetapi elemen dasar yang harus ada dan pasti dimiliki oleh sebuah algoritma *behaviour tree* adalah sebagai berikut (Sekhavat, 2017).

1. *Leaf Node*, merupakan sebuah node yang bertugas menjalankan sebuah aksi. Seperti menyapa, mendekati, mengejar, memutar clip suara, menjalankan animasi dan sebagainya. Jika komponen ini berhasil dijalankan maka akan mengembalikan kembalian *success*. Sedangkan, jika gagal akan mengembalikan kembalian *failure.* Jika komponen masih dalam keadaan berjalan maka akan mengembalikan kembalian *idle.*
2. *Condition Component* (if-else), merupakan sebuah komponen yang berupa nilai *boolean question,* berfungsi mengembalikan nilai true atau false. Kondisi ini tidak mungkin untuk mengembalikan nilai *looping* seperti *idle*. Pada gambar 2.1 digambarkan dalam bentuk pertanyaan. '*player status high?*'

3. *Control Flow* (*Selector* dan *Sequence*), adalah kumpulin komponen *child* yang berfungsi menentukan urutan pelaksanaan sebuah aksi, diantaranya terdapat dua macam *control flow* secara umum.

   a. *Selector* merupakan komponen yang berjalan dengan cara membuat keputusan. Berjalan dari kiri ke kanan. Contoh pada gambar 2.1 berupa lingkaran dengan tanda tanya. Cara kerjanya dapat dilihat pada gambar 2.2.

   b. S*equence* merupakan komponen yang didalamnya terdapat sederet child yang akan dieksekusi mulai dari kiri ke kanan. Contoh pada gambar 2.1 berbentuk anak panah. Cara kerjanya dapat dilihat pada gambar 2.2.

Tabel 2.2 Kelebihan dan Kekurangan *Behaviour Tree* (Mcquillan, 2015).

	Simplicity	**Complexity**	**Readability**	**Debugability**	**Scalability**	**Modularity**	**Reusability**
Strength	Elegant & intuitive	Powerfull	Non-programmer Designer	Simple to Follow	No fixed limit to number of levels	Aids reusability hybridity	Increased productivity
Limitation		Can affect performance					
	Flexibility	**Composability**	**Hierarchy**	**Dependency**	**Nuance**	**Hybridity**	**Efficiency**
Strength	Many applications & hybrid approaches	Sub-trees handle complexity, & hybridity, while maintaining a single shared, intuitive interface	Improvement on FSMs, HSMs for managing complexity		Bottom Up can make more informed, better decision	Eminently suitable for FSMs, Planners, & most others; Can result in best of both worlds	Top Down (Fast) Data-oriented (Cache) Event-driven (Reduced workload)
Limitation	Capable of most AI needs, but not ALWAYS the very best option		Static prioritization can increase dependences	Reliance on fluid externalities can lead to "stupid" behaviour/ loops	Can require knowledge of changing external systems, which means constant update	Need a good understanding of what works best when, to take full advantage of hybridity	Bottom Up (Increased processing power required)

Pada Tabel 2.2 dapat dipahami bahwa *Behaviour Tree* merupakan metode yang elegan dan sangat kuat, serta memudahkan pengguna dalam mengatur *task*

serta alur yang akan dijalankan. Selain itu alur yang sangat mudah dan mudah dipahami oleh *non-programmer* menjadi salah satu kelebihan metode ini.

Dalam menangani *task* yang rumit dan banyak berdasarkan turunan dari *child* tidak terbatas. Metode *Behaviour Tree* tetap sanggup untuk menangani hal ini meskipun berpengaruh terhadap performa dalam *game*. Dalam penggunaannya, metode *Behaviour Tree* dapat digunakan secara berulang dan dapat dihibridasi dengan metode lain sehingga dapat menciptakan *task* lain yang lebi kompleks sesuai dengan kebutuhan pengguna. Metode ini juga dapat diimplementasikan dalam banyak hal dan tidak hanya terbatas hanya untuk penggunaannya dalam *game*.

2.6 Decision Support System

Sistem pendukung keputusan (SPK) atau *Decision Support System* (DSS) adalah bagian dari sistem informasi berbasis komputer termasuk sistem berbasis pengetahuan atau manajemen pengetahuan yang dipakai untuk mendukung pengambilan keputusan dalam suatu organisasi atau perusahaan. (Fauzan et al., 2018). Sistem pendukung keputusan adalah sebuah cara untuk menentukan keputusan berdasarkan analisis kriteria serta bobot yang sebelumnya telah ditetapkan menjadi variabel atau data yang nantinya akan dijadikan parameter yang akan menghasilkan *output* berupa alternatif keputusan

2.7 Multi Criteria Decision Making

Multi Criteria Decision Making (MCDM) merupakan salah satu metode pengambilan keputusan yang paling akurat. MCDM merupakan metode yang digunakan untuk menguji kemampuan pemodelam matematis dengan cara

membantu Menyusun masalah pengambilan keputusan dan menghasilkan preferensi dari alternatif. Metode ini mempertimbangkan kriteria kualitatif dan kuantitatif berbeda yang perlu diperbaiki untuk menemukan solusi terbaik. Salah satu kriteria yang paling umum dalam permasalahan dalam pengambilan keputusan yaitu, *cost* atau harga kualitas prosesnya. Selain itu, terdapat bobot yang berbeda terhadap tiap-tiap kriteria yang dijadikan dasar penting masing-masing kriteria dalam kasus tertentu (Taherdoost & Madanchian, 2023).

2.8 Simple Additive Weighting

Simple Additive Weighting (SAW) yang juga dikenal sebagai dengan metode penjumlahan terbobot. Konsep dasar dari metode SAW adalah untuk menemukan jumlah bobot dari tingkat performa pada tiap alternatif ke atribut (Senthil Kumar & Malathi, 2018). Metode SAW yang merupakan salah satu dari banyak metode yang digunakan untuk menentukan keputusan berdasarkan penjumlahan terbobot dari rating kinerja pada tiap alternatif dari semua atribut yang diinisiasikan. Metode SAW akan melakukan normalisasi matriks keputusan (X) ke suatu skala yang dapat dibandingkan dengan semua rating alternatif yang ada, pada akhir tahap normalisasi akan dihasilkan keputusan berupa alternatif terbaik berdasarkan kriteria dan bobot yang telah ditentukan.

Metode *simple additive weighting* memiliki beberapa tahap inisiasi serta perhitungan hingga tahap perangkingan yang merupakan penentuan alternatif berdasarkan nilai tertinggi. Pada tahap awal terdapat inisiasi kriteria yang merupakan parameter data yang akan diambil dari user. Selanjutnya terdapat tahap inisiasi alternatif, dimana merupakan bentuk *output* yang akan dihasilkan dalam

perangkingan. Setelah itu tahap inisiasi bobot, yang dimana memberikan besaran tingkat kepentingan pada masing-masing kriteria.

Setelah mendapatkan data set, maka tahap selanjutnya yaitu matriks keputusan, dimana tahap pembagian nilai data pada tiap kriteria dengan nilai maksimum pada masing-masing kriterianya. pada tahap normalisasi nilai, jika faktor kriteria benefit digunakan rumus,

$$r_{ij} = \{\frac{x_{ij}}{max_i\{x_{ij}\}}\} \quad (2.3)$$

Jika faktor kriteria cost maka digunakan rumus,

$$r_{ij} = \{\frac{min_i\{x_{ij}\}}{x_{ij}}\} \quad (2.4)$$

Keterangan :
r_{ij} = Rating Kinerja Ternormalisasi
x_{ij} = Matrix Keputusan pada Setiap Alternatif dan Setiap Kriteria

Setelah mendapatkan hasil normalisasi data pada tiap-tiap matrix, dilakukan perangkingan terkait keputusan yang tepat berdasarkan kriteria dan alternatif yang telah didapat, dilakukan penjumlahan tiap alternatif berdasarkan perkalian matriks ternormalisasi (R) dengan bobot (W) yang bersesuaian elemen kolom matriks (W) dengan rumus berikut,

$$V_i = \sum_{j=1}^{n} w_j \, r_{ij} \quad (2.5)$$

Keterangan:

V_i = rangking untuk setiap alternatif
W_j = nilai bobot dari setiap kriteria
R_{ij} = nilai rating kinerja ternormalisasi

BAB III

METODOLOGI PENELITIAN

3.1 Rancangan Game

Pada perancangan *game* terdapat beberapa tahap yang dimulai dengan penjelasan *game*, perancangan desain serta *interface* serta asset-aset yang nantinya akan digunakan di dalam *game*. pada tahap ini, dijelaskan secara detail alur permainan serta deskripsi dari objektif permainan yang harus dilakukan oleh *player* untuk menyelesaikan *game* "THE MA'HAD".

3.1.1 Deskripsi Game

Penelitian ini berupa *life simulation* yang menceritakan serta menggambarkan kehidupan di dalam ma'had. *Game* ini memuat edukasi yang di dalamnya berisi materi terkait kegiatan dan pembelajaran yang diajarkan saat berada di ma'had. *Game* ini menggambarkan suasana ma'had yang divisualisasikan dalam bentuk tiga dimensi yang dibuat semirip mungkin dengan kondisi ma'had Sunan Ampel Al-'Aly. Di dalam *game* ini *player* diharuskan untuk menjalani *test* yang telah tersedia berdasarkan alur yang sudah ditetapkan, tidak hanya *test player* juga diharuskan untuk mengikuti pembelajaran materi yang disajikan melalui beberapa NPC yang mewakili sosok musyrif yang berada di dalam ma'had. Pada tiap NPC, *player* akan mendapatkan materi serta *test* yang berbeda hingga akhirnya *player* akan mendapatkan skor hasil akhir dari proses pembelajaran.

3.1.2 *Storyline*

Game ini menceritakan keadaan seorang mahasantri baru yang baru saja memasuki kehidupan di ma'had Sunan Ampel Al-'Aly. Berdasarkan gambar 3.1, dapat dilihat bahwa pada awal permainan *player* akan berada dalam suatu lingkungan ma'had atau asrama dimana terdapat empat NPC yang merupakan representasi musyrif yang masing-masing memiliki materi ta'lim yang berbeda. *Player* diharuskan untuk menghampiri tiap-tiap musyrif untuk menyelesaikan setiap pertanyaan serta rangkaian tes yang nantinya akan diberikan pada masing-masing musyrif.

Saat pertama kali *player* bertemu dengan NPC, *player* diharuskan menjawab pertanyaan terkait pengalaman terhadap materi yang nantinya akan muncul saat menjalankan tes tersebut. Setelah menjawab beberapa pertanyaan pengalaman, selanjutnya NPC akan memberikan *pre-test* kepada *player* untuk mengukur tingkat pemahaman *player* terkait materi yang nantinya akan dipelajari, tes ini didasari oleh kegiatan yang dilakukan pada tahap awal saat mahasantri baru memasuki kehidupan ma'had. Setelah melakukan *pre-test* yang diberikan oleh masing-masing NPC yang berjumlah empat karakter. Setelah melakukan *pre-test, player* harus kembali menemui keempat NPC tersebut untuk melakukan pembelajaran terkait materi agama islam yang dimana pada ke-empat NPC tersebut memiliki sub bab materi pembelajaran yang berbeda-beda.

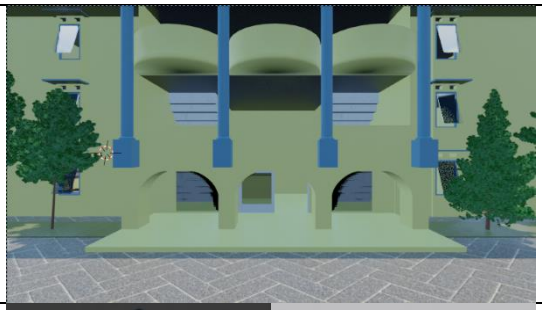
Setelah *player* berhasil menyelesaikan seluruh materi, *player* akan dinyatakan telah berhasil menyelesaikan permainan tersebut. Permainan ini berisi kehidupan serta pembelajaran yang diajarkan dalam lingkungan ma'had Sunan

Ampel Al-Aly, dengan harapan *player* dapat menikmati serta memahami kegiatan pembelajaran yang diberikan.

3.1.3 *Storyboard*

Berdasarkan *storyline* yang telah dijelaskan, Tabel 3.1 merupakan *storyboard* yang nantinya akan digunakan di dalam permainan.

Tabel 3.1 *Storyboard*

No	Gambar	Keterangan
1.		Pada posisi awal *player* akan *spawn* di dalam mahad
2.		*Player* diharuskan untuk menghampiri tiap-tiap NPC untuk menyelesaikan *pre-test,* pembelajaran materi serta *post-test*
3.		Permainan akan selesai apabila semua objektif telah dilakukan

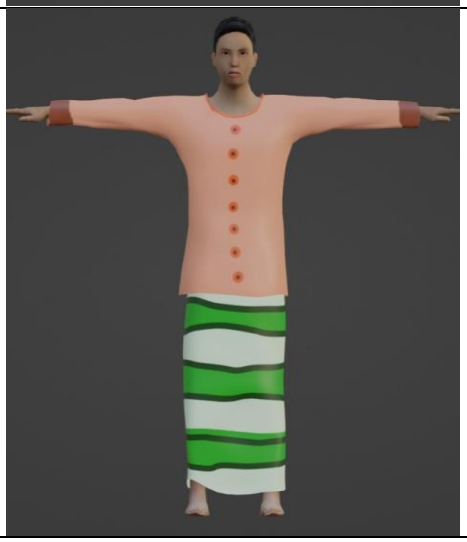
3.1.4 Desain *User Interface*

Pada Tabel 3.2 berisikan desain *interface* yang akan digunakan dalam *game* "THE MA'HAD" beserta dengan penjelasannya.

Tabel 3.2 Desain *User Interface*

No	*Frame*	Keterangan
1.		• *Button Play* untuk memulai permainan • *Button Setting* untuk konfigurasi suara • *Button Instruction* berisi petunjuk menjalankan permainan • *Button Credit* berisi tentang pembuatan permainan • *Button Quit* untuk meninggalkan permainan
2.		Ahmad Penanggung Jawab Ta'lim Al-Qur'an Materi 1
3.		Muklis Penanggung Jawab Ta'lim Al-Qur'an Materi 2

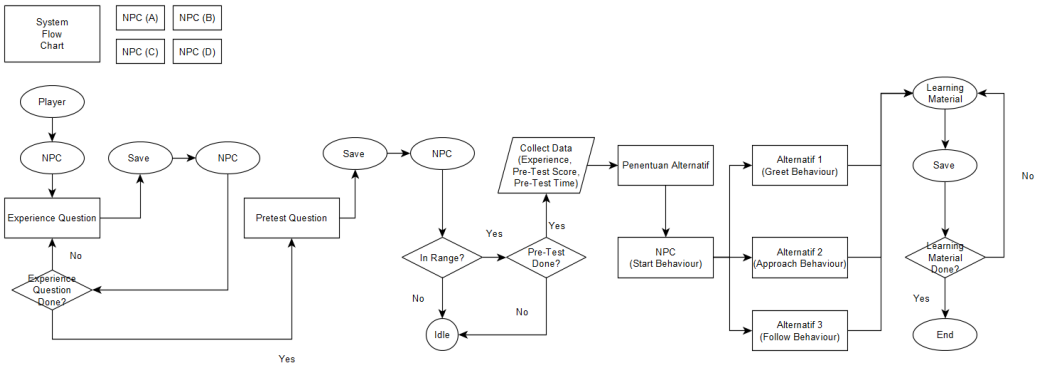
Lanjutan Tabel 3.2

4.		Muhammad Sumbul Penanggung Jawab Ta'lim Afkar Materi 1
5.		Furqon Penanggung Jawab Ta'lim Afkar Materi 2

3.2 Perancangan Metode

Pada perancangan metode berisikan alur penelitian yang akan dilakukan serta inisiasi perilaku yang akan diterapkan terhadap *non-playable character* yang merupakan objek implementasi metode yang akan dilakukan dalam penelitian. Terdapat inisiasi perilaku yang diimplementasikan ke dalam metode *behaviour tree* serta perhitungan *decision support system* yang digunakan untuk menghasilkan alternatif yang merupakan perilaku NPC di dalam *game.*

3.2.1 Non-Playable Character Behaviour

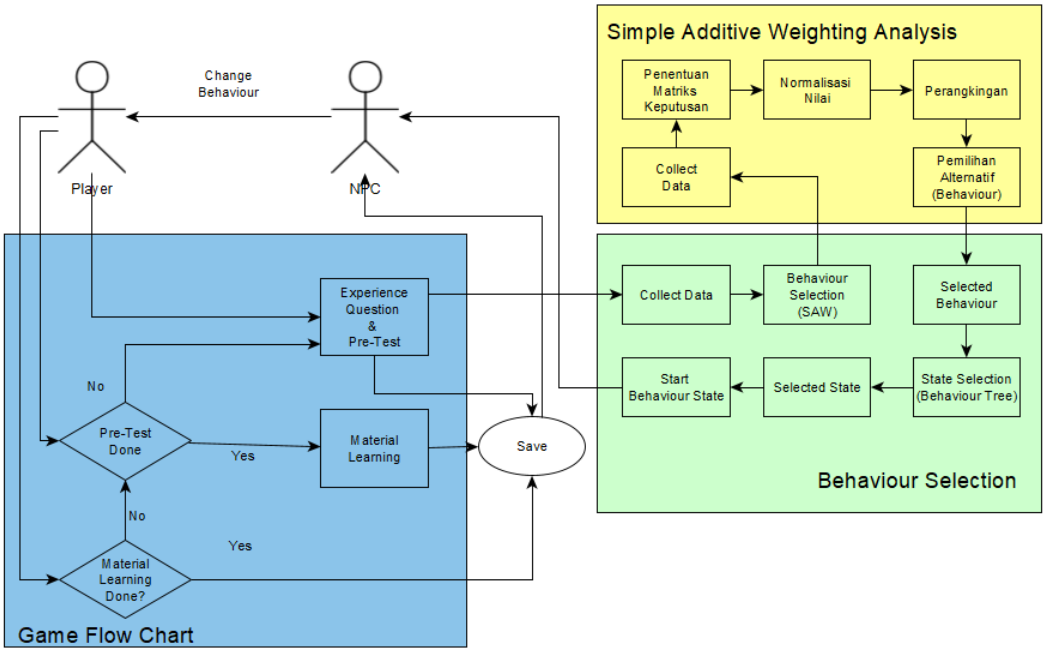


Gambar 3.1 Diagram Alur Permainan

Pada Gambar 3.1 dapat dilihat, permainan diawali dengan kondisi *player* diharuskan menemui NPC untuk menyelesaikan pertanyaan terkait pengalaman terhadap beberapa materi yang akan diujikan dalam *pretest.* Selanjutnya setelah *player* menyelesaikan pertanyaan terkait pengalaman terkait materi, jawaban *player* akan disimpan dan ditampung sebagai salah satu kriteria yang nantinya akan digunakan untuk menghasilkan alternatif *behaviour* NPC. Setelah itu, *player* diharuskan untuk menemui NPC kembali untuk melaksanakan *pretest.* Hasil *pretest* yang berupa nilai dan waktu pengerjaan akan disimpan untuk penentuan alternatif *behaviour* NPC.

Setelah seluruh kriteria yang dibutuhkan telah terpenuhi, maka sistem akan menjalankan perhitungan menggunakan metode *simple additive weighting* hingga menemukan alternatif *behaviour* yang sesuai dengan *value* dari masing-masing kriteria. Apabila *value* dari tiap kriteria menghasilkan nilai tinggi, maka alternatif yang dijalankan merupakan alternatif 1, dimana sistem akan mengeksekusi *greet behaviour* sebagai *output* perilaku NPC. Apabila *value* dari tiap kriteria menghasilkan nilai sedang atau rata-rata, alternatif yang dijalankan merupakan

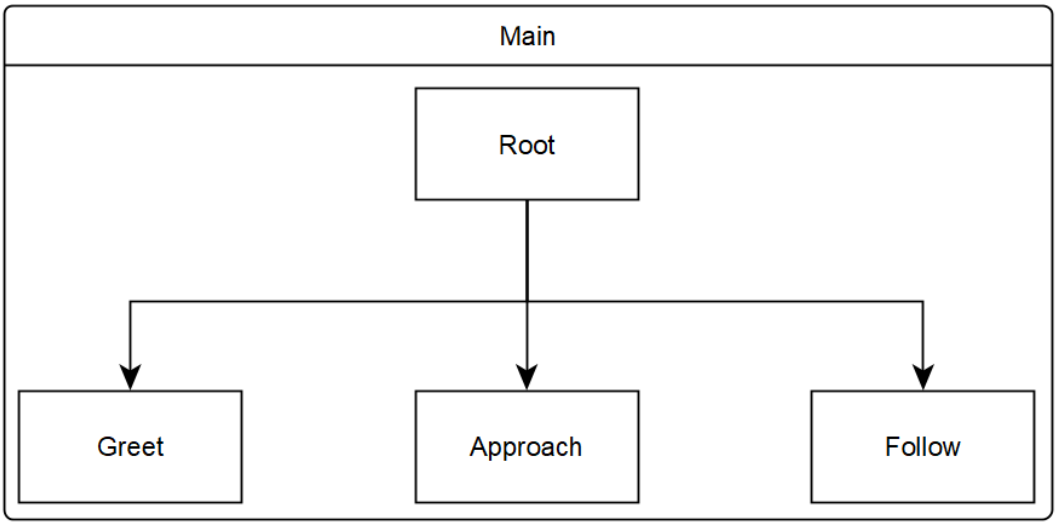
alternatif 2, dimanaa sistem akan mengeksekusi *approach behaviour* sebagai *output* perilaku NPC. Apabila *value* dari tiap kriteria menghasilkan nilai rendah, maka alternatif yang dijalankan merupakan alternatif 3, dimana sistem akan mengeksekusi *follow behaviour* sebagai *output* perilaku NPC. Pada setiap perilaku yang dieksekusi NPC, terdapat perintah untuk menyelesaikan pembelajaran materi terkait masing-masing ta'lim, perbedaan perilaku akan terlihat pada *gesture* NPC yang akan semakin memaksa apabila nilai *player* semakin rendah.



Gambar 3.2 Diagram Blok Penelitian

Berdasarkan Gambar 3.2 Pada NPC akan diberikan beberapa perilaku. Setiap perilaku NPC terhadap *player* akan berbeda berdasarkan hasil nilai *player* pada saat menjalankan *pretest* sebelumnya. Berikut merupakan perilaku NPC yang berdasar pada nilai *player* pada *pretest.* Pada kondisi awal NPC hanya akan berdiam di tempat yang telah ditetapkan pada masing-masing NPC, namun apabila *player*

memasuki wilayah NPC yang sudah ditetapkan menggunakan *collider*, maka perilaku NPC akan berubah sesuai berdasarkan status *player*.



Gambar 3.3 *Behaviour* Utama pada NPC

Berdasarkan pada Gambar 3.3 NPC memiliki tiga perilaku utama yang pada masing-masing perilaku memiliki child masing-masing. Pada NPC diberikan tiga *task* utama atau tiga perilaku dasar yang menjadi *main root* dari masing-masing *child.* Berikut merupakan *task* utama pada NPC.

a. Idle

Task idle merupakan perilaku *default* pada NPC apabila tidak ada *player* yang memasuki wilayah *collider* NPC. *Task* ini merupakan kondisi *looping* yang dimana akan terus berjalan selama kondisi masih terpenuhi.

b. Greet

Task greet merupakan perilaku pada NPC untuk menyapa apabila *player* memasuki wilayah *collider* NPC.

c. Approach

Task approach merupakan perilaku pada NPC untuk menghampiri apabila *player* memasuki wilayah *collider NPC.*

d. Follow

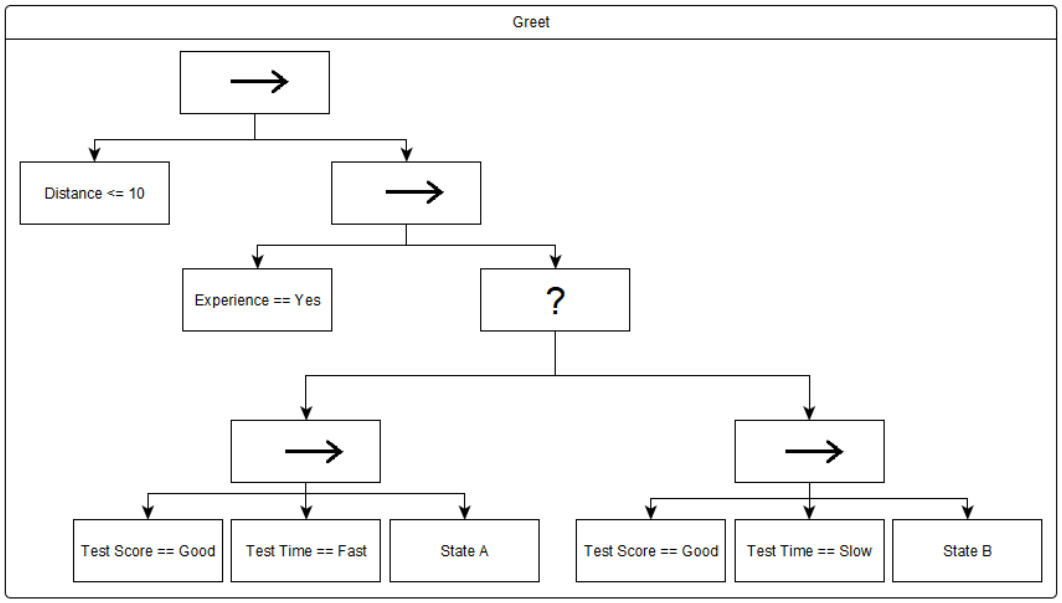
Task follow merupakan perilaku pada NPC untuk mengejar apabila *player* telah memasuki wilayah *collider NPC.*

Berdasarkan *task* yang dimiliki oleh NPC tersebut, *Behaviour* NPC dapat memiliki perilaku seperti berikut. Prioritas utama perilaku NPC yaitu adalah *idle* dimana NPC hanya akan menampilkan animasi berdiam di tempat yang sudah ditentukan.

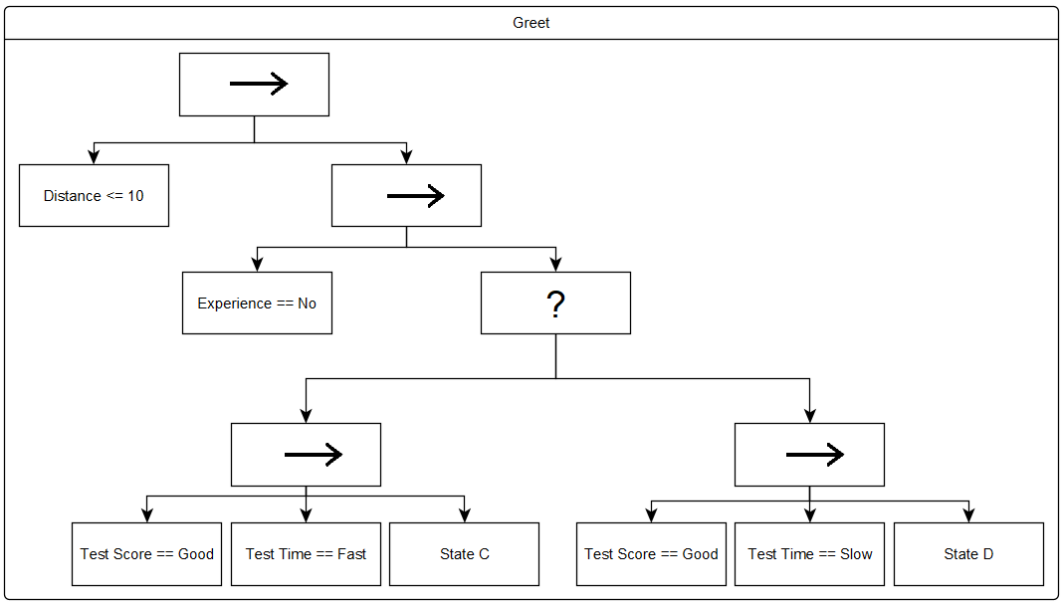
Selanjutnya berdasarkan urutan hasil nilai *pre-test* mulai dari terendah ke tertinggi, maka tingkatan prioritas perilaku NPC akan bermula pada *task follow*, *task* ini akan dijalankan dengan syarat skor nilai *player* masuk ke dalam kategori rendah, saat *player* mulai memasuki wilayah collider NPC, maka NPC akan mengejar *player* hingga *player* berhenti pada jarak interaksi NPC. Selanjutnya apabila skor *player* pada tingkat menengah maka *task* perilaku yang akan dijalankan yaitu *approach* dimana NPC akan menghampiri *player* apabila *player* berada dalam jangkauan, NPC akan berhenti menghampiri *player* apabila *player* berada pada jarak interaksi NPC. *Task* terakhir yaitu *greet* yang dimana apabila *player* memiliki skor tinggi, maka NPC akan menyapa *player* apabila *player* berada dalam jarak yang telah ditetapkan.

Berdasarkan beberapa *task* yang telah diinisiasikan diatas, perhitungan terkait syarat serta kondisi terpenuhinya *task* tersebut akan menggunakan sistem

pendukung keputusan metode *simple additive weighting* agar menghasilkan perilaku yang lebih adaptif terhadap kondisi perkembangan *player* di dalam *game*.



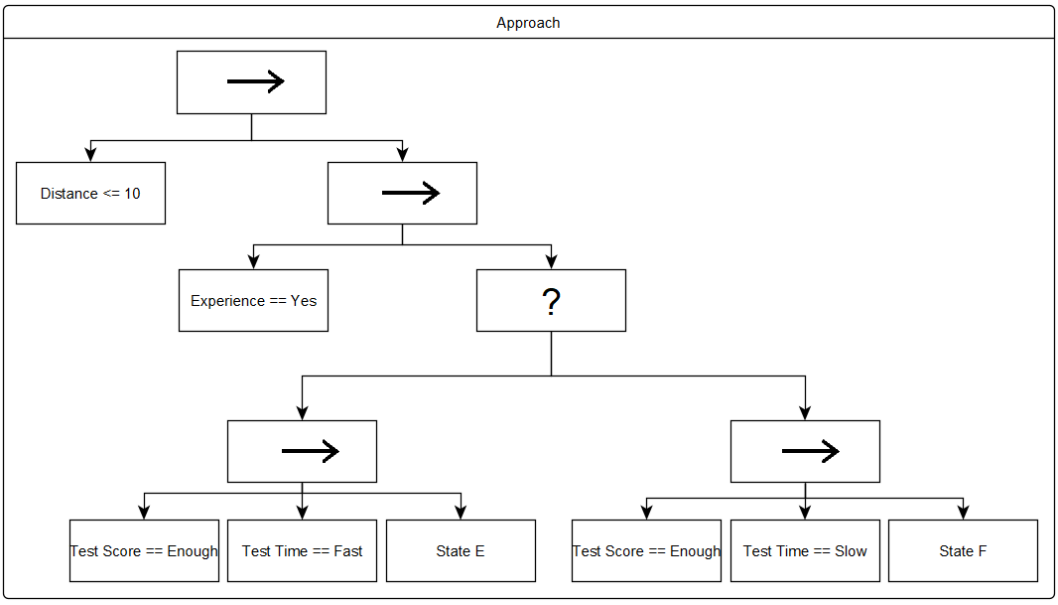
Gambar 3.4 *Behaviour Greet* dengan Kondisi Berpengalaman



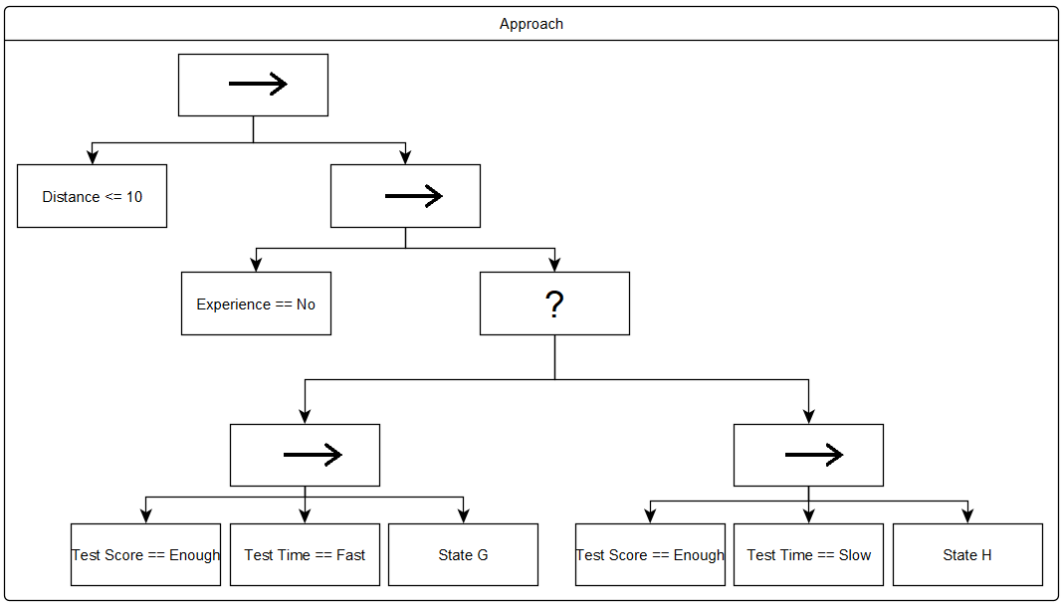
Gambar 3.5 *Behaviour Greet* dengan Kondisi Tidak Berpengalaman

Pada Gambar 3.4 dan Fambar 3.5 digambarkan bahwa child dari *behaviour greet*, node akan di *ticked* mulai dari arah kiri ke kanan. Pertama-tama NPC akan

mengecek kondisi jarak, apabila *player* telah memasuki jarak NPC maka akan berlanjut pada node *experience*, disini NPC akan membaca data pada *player* berdasarkan pada *player* apakah pernah atau tidak menjalani kehidupan pondok pesantren dan sebagainya. Selanjutnya akan digunakan node *selector* yang mana akan dilakukan *ticked* secara bersamaan dengan berdasarkan data *player* terkait skor dan waktu pengerjaan soal, maka NPC akan menentukan *state* yang lebih cocok. Berdasarkan kriteria yang telah ditentukan, pemilihan perilaku *greet* berdasar atas hasil nilai yang tinggi dalam *pretest* sebelumnya. Pada *behaviour greet* sendiri akan terbagi menjadi empat *state* yang berbeda bergantung pada hasil perkembangan *player* pada *pretest.*

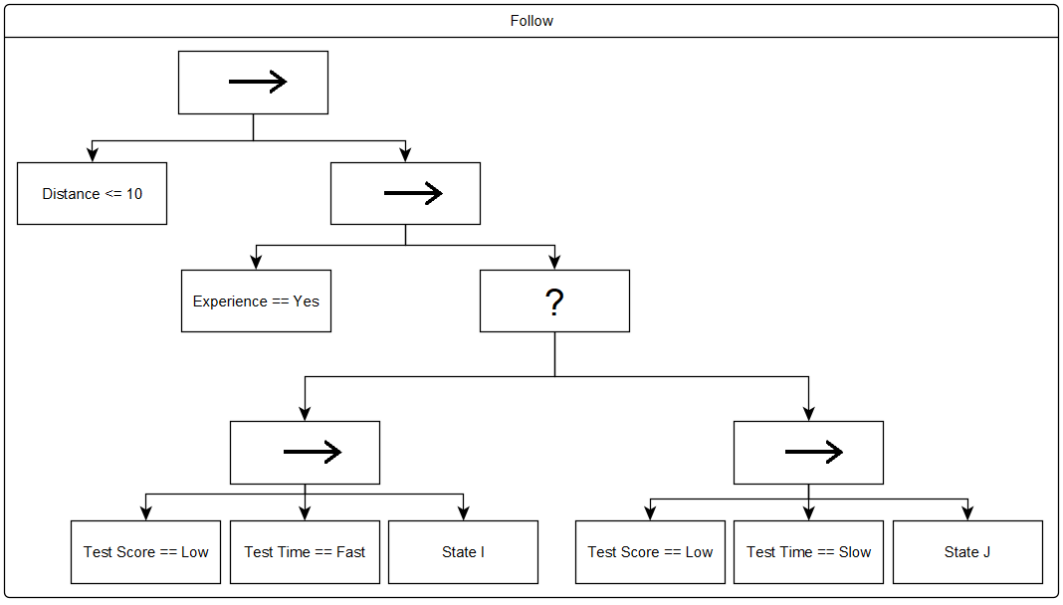


Gambar 3.6 *Behaviour Approach* dengan Kondisi Berpengalaman

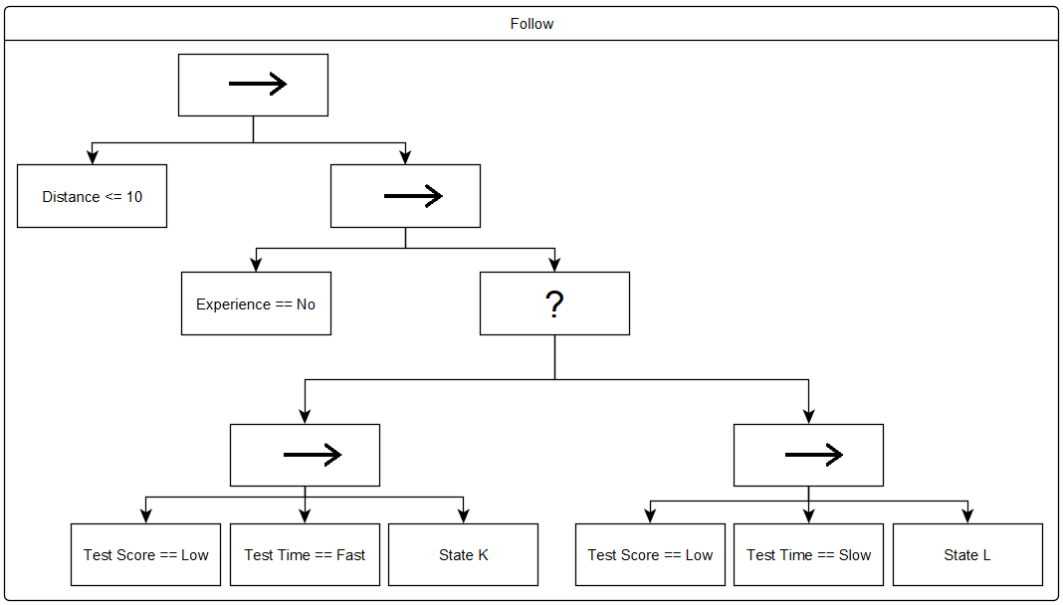


Gambar 3.7 *Behaviour Approach* dengan Kondisi Tidak Berpengalaman

Sama seperti alur pada *behaviour greet*, Pada Gambar 3.6 dan Gambar 3.7 *behaviour approach* akan melakukan *ticked* mulai dari kiri ke kanan. Pada *behaviour approach*, state ini akan berjalan apabila hasil nilai *pretest player* dikatakan cukup atau ditingkat rata-rata.



Gambar 3.8 *Behaviour Follow* dengan Kondisi Berpengalaman



Gambar 3.9 *Behaviour Follow* dengan Kondisi Tidak Berpengalaman

Pada Gambar 3.8 dan Gambar 3.9 *state* ini akan berjalan apabila *player* mendapatkan hasil nilai pada *pretest* dalam kategori kurang atau rendah, maka NPC akan menjalankan *state* pada *behaviour approach.*

3.2.2 Analisis Perhitungan dengan SAW

Terkait NPC, Terdapat beberapa kriteria yang menjadi parameter terhadap perubahan *behaviour* pada NPC

Tabel 3.3 Kriteria Penentuan Perilaku

No.	Kriteria	Nama Kriteria	Keterangan
1.	C_1	Nilai	Benefit
2.	C_2	Waktu	Benefit
3.	C_3	Pengalaman	Benefit

Berdasarkan Tabel 3.3 terdapat empat kriteria yang terdiri dari jarak, pengalaman, nilai, dan waktu yang dimana masing-masing kriteria bernilai benefit.

Tabel 3.4 Alternatif Perilaku

No.	Kode	Nama Kriteria
1.	A_1	*Greet*
2.	A_2	*Approach*
3.	A_3	*Follow*

Berdasarkan Tabel 3.4, terdapat tiga alternatif yang terdiri dari *greet*, *approach*, dan *follow*, yang dimana masing-masing alternatif merepresentasikan perilaku yang akan ditampilkan atau dijalankan sebagai output dalam *game*.

Tabel 3.5 Tingkat Kepentingan (Bobot)

Kode Kriteria	Nama Kriteria	Tingkat Kepentingan
K1	Nilai	0,5
K2	Waktu	0,3
K3	Pengalaman	0,2

Berdasarkan Tabel 3.5, Penentuan tingkat kepentingan atau bobot (W) untuk masing-masing kriteria yang telah ditetapkan sebagai berikut.

Tabel 3.6 Data Crips

Kriteria	Crips	Keterangan
Nilai	0 > n > 4	*Greet*
	5 > n > 8	*Approach*
	9 > n > 12	*Follow*
Waktu	n < 20	*Greet*
	20 < n < 40	*Approach*
	40 < n < 60	*Follow*
Pengalaman	Pernah belajar (n = 1)	*Greet*
	Kurang paham (n = 2)	*Approach*
	Tidak Pernah (n = 3)	*Follow*

Pada Tabel 3.6 untuk data crips yang digunakan sebagai parameter penilaian masing-masing dibagi menjadi tiga, yang dimana tiap nilai memiliki batasan yang digunakan sebagai pemilihan skala penilaian terhadap value, data crips merepresentasikan parameter pada tiap alternatif yang merujuk pada perilaku NPC.

Tabel 3.7 Skala Penilaian

Keterangan	ROC	Tingkat Kepentingan
Sesuai	(1+1/2+1/3)/3	0,611
Cukup Sesuai	(1/2+1/3)/3	0,277
Tidak Sesuai	(1/3)/3	0,111

Pada Tabel 3.7 untuk skala penilaian yang digunakan terbagi menjadi tiga, yaitu sesuai, cukup sesuai, dan tidak sesuai. Data yang didapat dari *player* akan digunakan sebagai input data. Berdasarkan tujuan dari penelitian yang dimana menghasilkan perilaku NPC yang interaktif maka tingkat prioritas alternatif yang sesuai dengan tujuan penelitian dimulai dari *follow*, *approach*, dan dilanjut dengan *greet* pada tingkat prioritas terakhir. Penggunaan tingkat prioritas ini untuk mengisi skala penilaian pada keterangan cukup sesuai.

Berdasarkan data yang didapatkan dari *player* melalui serangkaian tes yang telah disediakan, apabila data sesuai dengan kriteria yang telah diinisiasikan akan +bernilai 0,611, pada keterangan cukup sesuai akan bernilai 0,277 berdasarkan urutan tingkat prioritas , dan bernilai 0,111 pada keterangan tidak sesuai dengan kondisi apabila data dari *player* tidak memenuhi kriteria pada alternatif tersebut dan berada tingkat prioritas terendah.

Tabel 3.8 Matriks Keputusan

Alternatif	Kriteria		
	C_1	C_2	C_3
A_1	0,111	0,111	0,111
A_2	0,277	0,611	0,277
A_3	0,611	0,277	0,611

Pada perhitungan normalisasi nilai, jika faktor kriteria benefit digunakan rumus (2.3), Jika faktor kriteria cost maka digunakan rumus (2.4).

r_{11} = 0,611 / max {0,611;0,111;0,277} = 1,

r_{21} = 0,611 / max {0,111;0,611;0,277} = 1,

r_{31} = 0,611 / max {0,111;0,277;0,611} = 1,

r_{12} = 0,611 / max {0,611;0,111;0,277} = 1,

r_{22} = 0,611 / max {0,111;0,611;0,277} = 1,

r_{32} = 0,611 / max {0,111;0,277;0,611} = 1,

r_{13} = 0,611 / max {0,611;0,111;0,277} = 1,

r_{23} = 0,611 / max {0,111;0,611;0,277} = 1,

r_{33} = 0,611 / max {0,111;0,277;0,611} = 1,

Berdasarkan analisa perhitungan, penentuan value rating kinerja ternormalisasi {"sesuai"; 1} terletak pada kriteria yang sesuai berdasarkan data yang diambil dari *player* setelah menyelesaikan *pre-test.* Sebagai contoh, apabila didapatkan data dari *player* saat *pretest* dengan nilai = 10, waktu = 35, dan pengalaman = 3. Maka hasil perhitungan normalisasi akan seperti berikut:

Tabel 3.9 Hasil Normalisasi Data

Alternatif	Kriteria		
	C_1	C_2	C_1
A_1	0,181669394	0,181669394	0,181669394
A_2	0,453355155	1	0,453355155
A_3	1	0,453355155	1

Pada Tabel 3.9 dapat dilihat, untuk melakukan perangkingan terkait keputusan yang tepat berdasakan kriteria dan alternatif yang telah didapat, dilakukan penjumlahan tiap alternatif berdasarkan perkalian matriks ternormalisasi dengan bobot yang bersesuaian elemen kolom matriks, dengan menggunakan rumus (2.5).

Tabel 3.10 Perangkingan Nilai

Alternatif	Kriteria			Value	Rangking
	C_1	C_2	C_1		
A_1	0,090834697	0,054500818	0,036333879	0,181669394	3
A_2	0,226677578	0,3	0,090671031	0,617348609	2
A_3	0,5	0,136006547	0,2	0,836006547	1

V_1 = 0,181669394 V_2 = 0,617348609 V_3 = 0,836006547

Berdasarkan Tabel 3.10 hasil perangkingan yang telah dianalisa sesuai dengan data yang didapat, maka akan terdapat sebuah data yang memiliki value paling besar diantara yang lainnya, value ini menjadi dasar indikator pemilihan alternatif yang akan dipilih serta dieksekusi NPC dalam bentuk perubahan perilaku NPC terhadap *player* di dalam permainan.

3.3 Rencana Pengujian

Rencana pengujian dalam penelitian ini adalah dengan menggunakan validasi dari perilaku yang ditampilkan sebagai output oleh NPC dalam *game* ini. NPC yang sudah diberikann perilaku menggunakan metode, akan divalidasi dan disamakan dengan menggunakan *Black Box Testing* dengan NPC yang lain agar menghasilkan *Output* yang sama dan akurat.

Selanjutnya yaitu mengukur akurasi dari sistem NPC dengan nilai yang dimiliki *player.* NPC akan dikatakan berhasil apabila berhasil mencapai target yang diinginkan peneliti yaitu menjalankan seluruh *state* yang ada sesuai dengan keadaan yang diberikan oleh *player.*

BAB IV

IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan membahas hasil dari pembuatan *game* "THE MA'HAD". Pada bab ini pula akan dibahas validasi rancangan yang telah didesain pada bab sebelumnya, sehingga secara transparan dapat diketahui proses serta perkembangan pada *game* ini.

4.1 Implementasi Desain

Pada sub-bab ini akan dijelaskan pengimplementasian yang telah dilakukan dalam *game*, yang terdiri dari implementasi pada grafis visual, alur kerja sistem serta penerapan metode dalam permainan.

4.1.1 Hardware and Software

Uji Coba dilakukan pada PC dengan spesifikasi perangkat keras dan perangkat lunak sebagai berikut.

2. *Processor* : Intel Core i5-10300H CPU @ 2.5GHz (8CPUs), ~2.5GHz
3. RAM : 8192 MB
4. *Storage* : 500 GB SSD
5. *Operating System* : Windows 11 Home Single Language 64-bit
6. *Graphic Card* : NVIDIA GeForce GTX 1650
7. *Game Engine* : Unity 3D
8. 3D *Assets* : Blender (FBX, OBJ)
9. *Script Editor* : Microsoft Visual Studio 2019
10. 2D *Concept* : Figma, Canva, Photoshop

4.1.2 Desain Interface

Berikut tampilan *User Interface* serta *Gameplay* pada *game* ini. di dalam permainan *player* akan disambut oleh beberapa NPC yang pada masing-masing NPC bertanggung jawab atas ta'lim yang berbeda. Terdapat empat NPC yang merepresentasikan dua kelas ta'lim afkar dan dua kelas ta'lim Al-Qur'an.

Gambar 4.1 Interface *Game* THE MA'HAD

Dapat terlihat pada Gambar 4.1 dua NPC yang masing-masing mengangani ta'lim yang berbeda. Pada NPC memiliki perilaku yang akan berjalan ketika *player* telah menyelesaikan serangkaian test yang harus dilakukan.

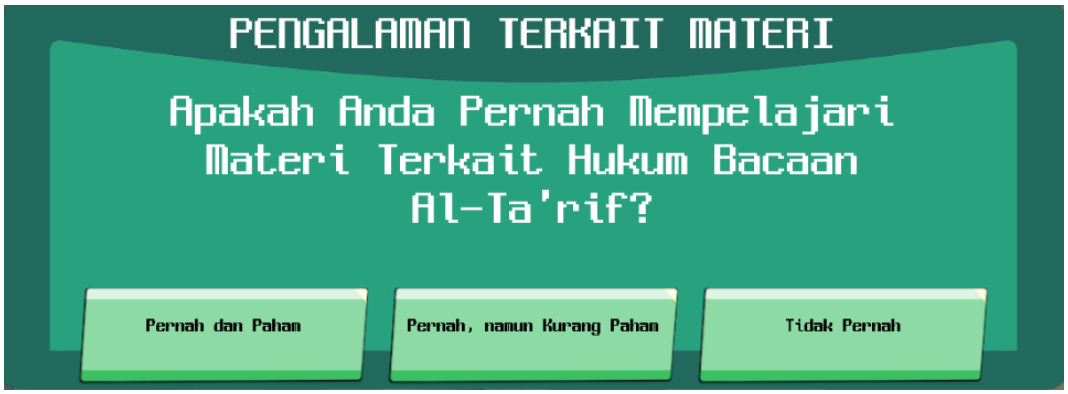
4.2 Implementasi Player Movement

Game The Ma'Had merupakan permainan yang mengusung unsur 3D yang di dalamnya memiliki *environment* dalam bentuk 3 dimensi. *User* yang memainkan *game* ini akan menggunakan karakter *player*, dimana *player* dapat melakukan pergerakan seperti berjalan dan berlari yang berpusat pada sumbu X dan sumbu Y.

Input pergerakan pemain diinisiasikan menggunakan *controller* yang terdapat pada *keyboard* dengan tombol “W, A, S, D” untuk menentukan arah pergerakan pemain dan menggunakan tombol “*LeftShitf”* yang dipadukan dengan dengan “W, A, S, D” untuk melakukan aksi berlari pada *player*.

4.3 Pengambilan Data

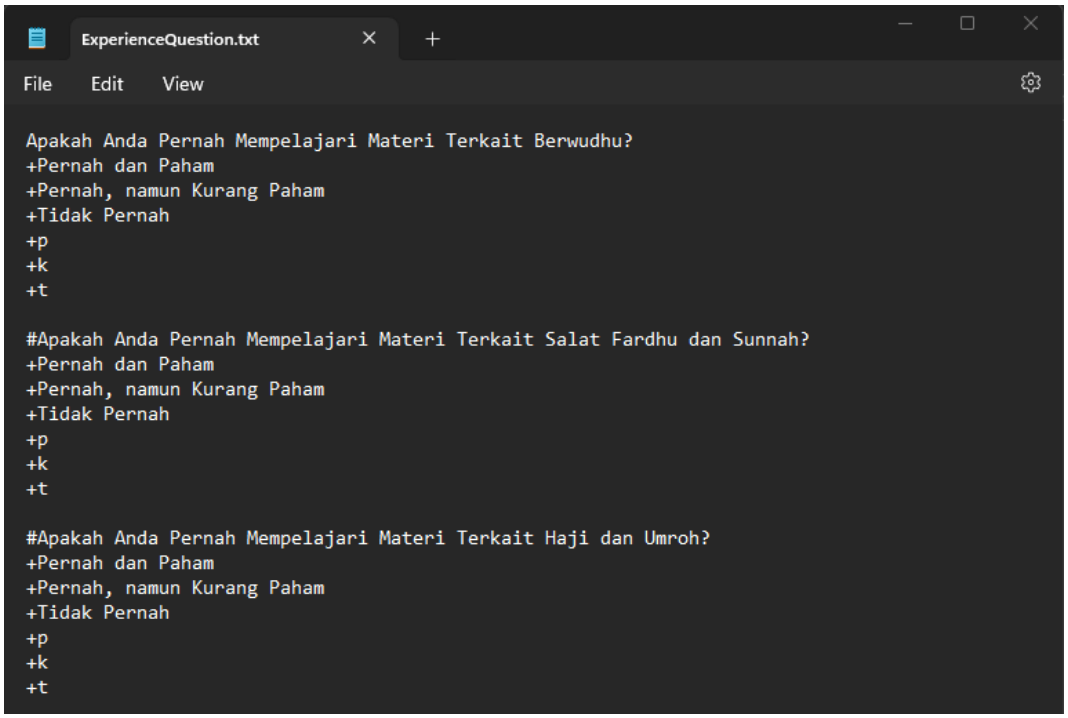
Proses pengambilan data akan dilakukan melalui dua cara, yang pertama yaitu melalui sebuah dialog yang dilakukan oleh NPC terhadap *player*. Dalam dialog ini NPC akan memberikan pertanyaan terkait pengalaman *player* terkait pengalaman *player* dalam pemahaman materi yang akan dikeluarkan saat melakukan *pretest.* Tahap kedua, setelah melakukan dialog, *player* akan menjalankan serangkaian *pretest* untuk mengetahui tingkat pemahaman *player* terhadap materi.



Gambar 4.2 Pengambilan Data Pengalaman

Gambar 4.2 merupakan tampilan pengambilan data terkait pengalaman yang dilakukan memalui interaksi antara *player* dengan NPC. Pertanyaan tersebut berisi tiga kategori pengalaman yang harus dijawab oleh *player* untuk mengukur tingkat pemahaman yang dimiliki *player* terkait test yang nantinya akan

diselesaikan oleh *player*. Pertanyaan dimunculkan dalam dialog yang dilakukan antara NPC dengan *player* disimpan ke dalam sebuah file berekstensi .txt yang dapat dirubah atau diedit apabila ingin mengganti pertanyaan yang akan ditanyakan kepada *player*.



Gambar 4.3 Penyimpanan Pertanyaan Pengalaman

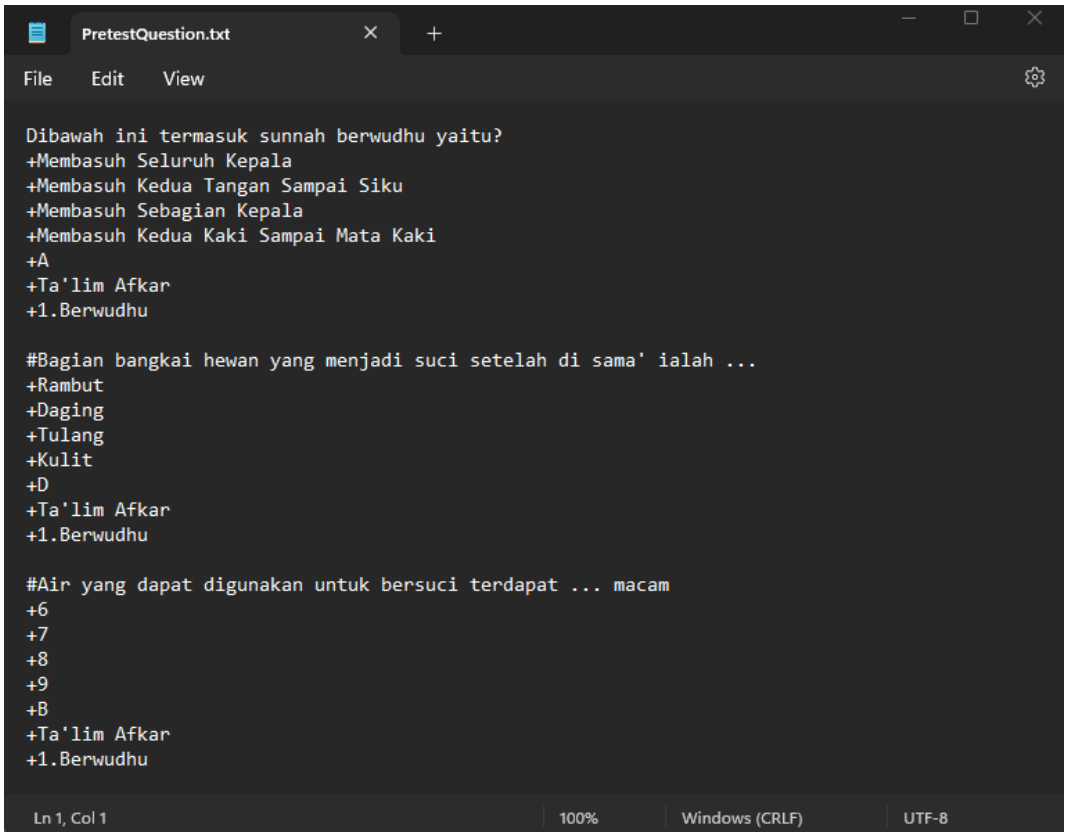
Pada Gambar 4.3 dapat dilihat, pertanyaan terkait pengalaman yang tersimpan dalam file berektensi .txt tersebut akan dipanggil dalam *source code* yang berperan sebagai pengolah output dari *experience question* yang terdapat dalam *game*. Pertanyaan tersebut dipanggil dalam bentuk list array yang mana pada tiap index pertanyaan diberikan batas dengan simbol (#) untuk pembentukan index baru, dan simbol (+) digunakan untuk memberikan pembatas pada tiap pilihan jawaban dalam masing-masing index pertanyaan. Index pertanyaan akan dipanggil dalam sebagai output pertanyaan begitu pula dengan pilihan jawaban pada tiap-tiap index

yang dimana jawaban tersebut yang nantinya akan dijadikan data terkait pengalaman *player*.

Gambar 4.4 Pengambilan Data *Pretest*

Pada Gambar 4.4 dapat terlihat bentuk *pretest* yang harus diselesaikan oleh *player*, test tersebut merupakan bentuk pengambilan data terkait pemahaman *player* terkait pembelajaran yang terdapat di Ma'had. Data yang akan diambil berupa jumlah soal yang terjawab salah serta lama waktu pengerjaan test yang dilakukan *player* hingga *pretest* tersebut terjawab seluruhnya atau hingga waktu pengerjaan habis dengan sendirinya. Sistem penyimpanan pertanyaan pada *pretest* sama halnya dengan sistem penyimpanan dengan pertanyaan pengalaman dimana telah

disiapkan file dengan ekstensi .txt yang akan menampung berbagai pertanyaan terkait materi *pretest*.



Gambar 4.5 Penyimpanan Pertanyaan *Pretest*

Gambar 4.5 merupakan daftar pertanyaan yang akan dimunculkan sebagai *pretest* yang akan dijawab oleh *player*, meskipun memiliki sistem penyimpanan serta pengolahan yang hampir sama dengan pertanyaan pengalaman, sistem pengambilan data pada *pretest* memiliki sedikit perbedaan, dimana pada *pretest* jawaban *player* akan diambil oleh sistem lalu disesuaikan dengan kunci jawaban yang terdapat pada file penyimpanan soal *pretest.* Apabila *player* berhasil menjawab pertanyaan dengan benar maka sistem akan menampung jumlah jawaban benar dan begitu pula sebaliknya apabila jawaban *player* tidak sesuai dengan kunci

jawaban yang telah ditentukan maka jawaban salah yang akan ditampung oleh sistem. Pada akhir *pretest* sistem akan mendata seluruh jawaban benar dan seluruh jawaban salah sehingga menghasilkan nilai yang merupakan kalkulasi dari jawaban benar yang dijawab oleh *player.*

Setelah mendapatkan data terkait nilai, waktu *pretest,* dan pengalaman terkait materi. Maka data tersebut akan disimpan yang nantinya akan berpengaruh terhadap *behaviour* NPC terhadap *player.*

4.4 Implementasi Metode

Pada implementasi metode dilakukan penerapan *decision support system* dengan model *simple additive weighting* dan metode *behaviour tree* dalam bentuk *source code* yang akan diimplementasikan ke dalam *game* “THE MA’HAD”

4.4.1 Simple Additive Weighting

Pada tahap pertama, setelah mendapatkan data melalui serangkaian *pretest,* data tersebut akan diolah menggunakan *decision support system* metode *simple additive weighting.* Pengolahan nilai dilakukan untuk mendapatkan alternatif yang berupa *behaviour* yang akan dieksekusi oleh NPC.

1. Implementasi Kriteria

Implementasi kriteria yang akan digunakan dalam *Decision Support System* metode *Simple Additive Weighting* akan terbagi menjadi tiga kriteria yaitu nilai, waktu, dan pengalaman.

A. Nilai

Dalam Penentuan Kriteria, nilai dibagi menjadi tiga kategori yaitu tinggi, sedang, dan rendah.

```
if (nilai > 8)
        {
            MKeputusan11 = 0,111;
            MKeputusan21 = 0,277;
            MKeputusan31 = 0,611;
        }
        else if (nilai > 4 && nilai <= 8)
        {
            MKeputusan11 = 0,111;
            MKeputusan21 = 0,611;
            MKeputusan31 = 0,277;
        }
        else if (nilai >= 0 && nilai <= 4)
        {
            MKeputusan11 = 0,611;
            MKeputusan21 = 0,111;
            MKeputusan31 = 0,277;
        }
```

Kategori tinggi dalam kriteria nilai memiliki *value* lebih dari 8, yang dimana apabila *value* memenuhi kondisi tersebut maka variabel alternatif11 akan memiliki inisiasi awal 0,75. Pada kategori sedang, kriteria nilai memiliki *value* lebih dari 4 dan kurang dari sama dengan 8, apabila *value* nilai memenuhi kondisi tesebut maka variabel alternatif21 akan memiliki inisiasi awal 0,75. Kategori rendah dalam kriteria nilai memiliki *value* lebih dari sama dengan 0 dan kurang dari sama dengan 4, yang dimana apabila *value* nilai memenuhi kondisi terebut maka variabel alternatif31 akan memiliki inisiasi awal 0,75.

B. Waktu

Dalam Penentuan Kriteria, waktu dibagi menjadi tiga kategori yaitu cepat, sedang, dan lambat (dalam hitungan detik).

```
if (waktu <= 20)
        {
            MKeputusan12 = 0,611;
            MKeputusan22 = 0,111;
            MKeputusan32 = 0,277;
        }
        else if (waktu > 20 && waktu <= 40)
        {
            MKeputusan12 = 0,111;
            MKeputusan22 = 0,611;
            MKeputusan32 = 0,277;
        }
        else if (waktu > 40 && waktu <= 60)
        {
            MKeputusan12 = 0,111;
            MKeputusan22 = 0,277;
            MKeputusan32 = 0,611;
        }
```

Kategori cepat dalam kriteria waktu memiliki *value* kurang dari sama dengan 20, yang dimana apabila *value* memenuhi kondisi tersebut maka variabel alternatif12 akan memiliki inisiasi awal 0,75. Pada kategori sedang, kriteria waktu memiliki *value* lebih dari 20 dan kurang dari sama dengan 40, apabila *value* waktu memenuhi kondisi tesebut maka variabel alternatif22 akan memiliki inisiasi awal 0,75. Kategori lambat dalam kriteria waktu memiliki *value* lebih dari 40 dan kurang dari sama dengan 60, yang dimana apabila *value* waktu memenuhi kondisi terebut maka variabel alternatif32 akan memiliki inisiasi awal 0,75.

C. Pengalaman

Dalam Penentuan Kriteria, pengalaman dibagi menjadi tiga kategori yaitu paham, kurang paham, dan tidak paham

```
if (pengalaman == 1)
        {
            MKeputusan13 = 0,611;
            MKeputusan23 = 0,111;
```

```
            MKeputusan33 = 0,277;
        }
        else if (pengalaman == 2)
        {
            MKeputusan13 = 0,111;
            MKeputusan23 = 0,611;
            MKeputusan33 = 0,277;
        }
        else if (pengalaman == 3)
        {
            MKeputusan13 = 0,111;
            MKeputusan23 = 0,277;
            MKeputusan33 = 0,611;
        }
```

Kategori paham dalam kriteria pengalaman memiliki *value* sama dengan 1, yang dimana apabila *value* memenuhi kondisi tersebut maka variabel alternatif13 akan memiliki inisiasi awal 0,75. Pada kategori kurang paham, kriteria pengalaman memiliki *value* sama dengan 2, apabila *value* pengalaman memenuhi kondisi tesebut maka variabel alternatif23 akan memiliki inisiasi awal 0,75. Kategori tidak paham dalam kriteria pengalaman memiliki *value* sama dengan 3, yang dimana apabila *value* pengalaman memenuhi kondisi terebut maka variabel alternatif33 akan memiliki inisiasi awal 0,75.

2. Penentuan Bobot

Inisiasi bobot dalam C# menggunakan tipe data float, lalu data awal bobot dari setiap kriteria di inisiasi sesuai dengan bobot perhitungan masing-masing kriteria yang telah ditetapkan

```
float bobotNilai = 0.5f;
float bobotWaktu = 0.3f;
float bobotPengalaman = 0.2f;
```

Inisiasi bobot dilakukan untuk memberikan *value* awal dikarenakan *value* bobot yang bersifat tidak berubah seiring berjalannya permainan berbeda dengan *value* pada nilai, waktu, serta pengalaman yang akan didapatkan setelah *player* melakukan permainan atau saat permainan sedang berlangsung.

3. Perhitungan Normalisasi

Pada perhitungan normalisasi, setiap variabel normalisasi akan digunakan untuk menampung hasil perhitungan dari tiap-tiap alternatif dibagi dengan value 0,611 yang merupakan value tertinggi dari masing-masing alternatif menggunakan rumus (2.3).

Berdasarkan sifat rumus simple additive weighting hasil perhitungan normalisasi didapatkan dengan membagi setiap nilai pada matrix keputusan yang terdapat dalam setiap alternatif dan kriteria dengan nilai maksimumnya.

```
NormalisasiNilai11 = MKeputusan11 / 0,611;
NormalisasiNilai21 = MKeputusan21 / 0,611;
NormalisasiNilai31 = MKeputusan31 / 0,611;
NormalisasiNilai12 = MKeputusan12 / 0,611;
NormalisasiNilai22 = MKeputusan22 / 0,611;
NormalisasiNilai32 = MKeputusan32 / 0,611;
NormalisasiNilai13 = MKeputusan13 / 0,611;
NormalisasiNilai23 = MKeputusan23 / 0,611;
NormalisasiNilai33 = MKeputusan33 / 0,611;
```

Dikarenakan *value* maksimum yang akan didapat oleh *player* berupa *value* 0,611, yang dimana *value* tersebut didapat dari penetapan kesesuaian antara nilai, waktu dan pengalaman pada tiap matrix keputusan. Maka inisiasi perhitungan normalisasi yang akan diimplementasikan ke dalam permainan, akan menggunakan *source code* seperti yang tertera dan akan diproses saat permainan berlangsung ketika NPC telah mendapatkan seluruh data kriteria dari *player*.

4. Perhitungan Perangkingan

Pada perhitungan perangkingan, setiap variabel perangkingan akan digunakan untuk menampung hasil perhitungan dari normalisasi nilai dikali dengan masing-masing bobot pada setiap kriteria.

```
        perangkingan11 = NormalisasiNilai11 *
bobotNilai;
        perangkingan21 = NormalisasiNilai21 *
bobotNilai;
        perangkingan31 = NormalisasiNilai31 *
bobotNilai;
        perangkingan12 = NormalisasiNilai12 *
bobotWaktu;
        perangkingan22 = NormalisasiNilai22 *
bobotWaktu;
        perangkingan32 = NormalisasiNilai32 *
bobotWaktu;
        perangkingan13 = NormalisasiNilai13 *
bobotPengalaman;
        perangkingan23 = NormalisasiNilai23 *
bobotPengalaman;
        perangkingan33 = NormalisasiNilai33 *
bobotPengalaman;
```

Perangkingan dalam *decision support system model simple additive weighting* menggunakan rumus (2.5), dimana nilai hasil normalisasi akan dikali dengan *value* bobot pada setiap matriks nya. Perbedaan *value* pada masing-masing bobot (nilai, waktu, pengalaman) memiliki pengaruh yang cukup signifikan terhadap *output* alternatif yang berupa *bahaviour* NPC.

5. Penentuan Alternatif

Penentuan alternatif akan meggunakan Alt2, Alt3, dan Alt3 untuk menampung hasil dari penjumlahan antara masing-masing perangkingan berdasarkan rumus *simple additive weighting* yang diterapkan.

```
        Alt1 = perangkingan11 + perangkingan12 +
perangkingan13;
        Alt2 = perangkingan21 + perangkingan22 +
perangkingan23;
        Alt3 = perangkingan31 + perangkingan32 +
perangkingan33;

AltBehaviour = Mathf.Max(Alt1, Alt2, Alt3);
```

Penentuan alternatif diinisiasikan berdasarkan rumus *simple additive weighting* yang dimana alternatif 1 akan didapatkan melalui penjumlahan nilai hasil pada kriteria nilai, waktu dan pengalaman pada kolom alternatif 1, dan seterusnya hingga alternatif terakhir, dengan hal ini akan didapatkan nilai pada masing-masing alternatif yang ada. Selanjutnya akan ditentukan nilai alternatif tertinggi yang telah di ranking kemudian akan digunakan sebagai *output* yang dimana dalam hal ini akan menjadi *behaviour* NPC.

4.4.2 Behaviour Tree

Metode *Behavour Tree* memiliki beberapa unsur krusial yang menjadi pondasi untuk membentuk metode *Behaviour Tree.* Beberapa unsur ini harus ada dalam sebuah perancangan sistem yang menggunakan metode *behaviour tree,* masing-masing unsur memiliki peran penting dalam menjalankan perintah serta pengeksekusian sistem. Unsur-unsur tersebut diantaranya ialah *tree, node, selector,* dan *sequence.*

4.4.2.1 Tree

Tree adalah struktur dalam *behaviour tree* yang digunakan untuk mengorganisasi logika serta mengeksekusi setiap perilaku atau aksi yang nantinya

akan diinisiasikan, tree akan mengeksekusi setiap node mulai dari perilaku dengan tingkat prioritas tertinggi.

```
namespace BehaviourTree
{
    public abstract class Tree : MonoBehaviour
    {
        private Node _root = null;

        protected void Start()
        {
            _root = SetupTree();
        }
        private void Update()
        {
            if (_root != null)
                _root.Evaluate();
        }
        protected abstract Node SetupTree();
    }
}
```

4.4.2.2 Node

Unsur penting selanjutnya yang terdapat dalam *behaviour tree* ialah *node* atau *root node*, dimana *node* merupakan akar teratas dalam metode *behaviur tree.* Node menampung semua logika mulai dari penyimpanan, tingkat prioritas maupun kondisi setiap perilaku yang akan diinisiasikan. Node juga berfungsi sebagai pengarah eksekusi perilaku terhadap node di bawahnya yaitu *selector node* dan *sequence node.*

```
namespace BehaviourTree
{
    public enum NodeState
    {
        RUNNING,
        SUCCESS,
        FAILURE
    }
    public class Node
```

```
    {
        protected NodeState state;
        public Node parent;
        protected List<Node> children = new
List<Node>();

        private Dictionary<string, object>
_dataContext = new Dictionary<string, object>();

        public Node()
        {
            parent = null;
        }
        public Node(List<Node> children)
        {
            foreach (Node child in children)
                _Attach(child);
        }
        private void _Attach(Node node)
        {
            node.parent = this;
            children.Add(node);
        }
        public virtual NodeState Evaluate() =>
NodeState.FAILURE;

        public void SetData(string key, object value)
        {
            _dataContext[key] = value;
        }
        public object GetData(string key)
        {
            object value = null;
            if (_dataContext.TryGetValue(key, out
value))
                return value;

            Node node = parent;
            while (node != null)
            {
                value = node.GetData(key);
                if (value != null)
                    return value;
                node = node.parent;
            }
            return null;
        }
```

```
        public bool ClearData(string key)
        {
            if (_dataContext.ContainsKey(key))
            {
                _dataContext.Remove(key);
                return true;
            }
            Node node = parent;
            while (node != null)
            {
                bool cleared = node.ClearData(key);
                if (cleared)
                    return true;
                node = node.parent;
            }
            return false;
        }
    }
}
```

4.4.2.3 Selector

Selector node adalah *node* yang berada di bawah *node* atau *root node*. *Selector* memiliki beberapa *child node* yang nantinya akan dilakukan evaluasi sehingga mengembalikan *true/success*, apabila salah satu *child node* pada *selector* mengembalikan *true/success* maka *selector node* juga akan mengembalikan *true/success* yang berarti *child node* tersebut telah sukses memenuhi syarat untuk pengeksekusian perilaku. Namun apabila *child node* mengembalikan *false/failure* maka *selector* akan terus melakukan pencarian mulai dari *child node* dengan tingkat prioritas tertinggi hingga terbawah hingga menemukan *child node* yang mengembalikan *true/success.*

```
namespace BehaviourTree
{
    public class Selector : Node
    {
        public Selector() : base() { }
```

```
        public Selector(List<Node> children) :
base(children) { }
        public override NodeState Evaluate()
        {
            foreach (Node node in children)
            {
                switch (node.Evaluate())
                {
                    case NodeState.FAILURE:
                        continue;
                    case NodeState.SUCCESS:
                        state = NodeState.SUCCESS;
                        return state;
                    case NodeState.RUNNING:
                        state = NodeState.RUNNING;
                        return state;
                    default:
                        continue;
                }
            }
            state = NodeState.FAILURE;
            return state;
        }
    }
}
```

4.4.2.4 Sequence

Sequence node merupakan *node* yang memiliki fungsi hampir sama dengan *selector node*, namun dalam hal ini *sequence* memiliki kondisi nilai pengembalian yang berbeda dengan *selector*. *Sequence node* akan mengembalikan *true/success* apabila seluruh *child node* mengembalikan *true/success*, apabila terdapat salah satu *node child* mengembalikan *false/failure* maka *sequence* tidak akan mengembalikan *true/success.*

```
namespace BehaviourTree
{
    public class Sequence : Node
    {
        public Sequence() : base() { }
```

```
        public Sequence(List<Node> children) :
base(children) { }
        public override NodeState Evaluate()
        {
            bool anyChildIsRunning = false;
            foreach (Node node in children)
            {
                switch (node.Evaluate())
                {
                    case NodeState.FAILURE:
                        state = NodeState.FAILURE;
                        return state;
                    case NodeState.SUCCESS:
                        continue;
                    case NodeState.RUNNING:
                        anyChildIsRunning = true;
                        continue;
                    default:
                        state = NodeState.SUCCESS;
                        return state;
                }
            }
            state = anyChildIsRunning ?
NodeState.RUNNING : NodeState.SUCCESS;
            return state;
        }
    }
}
```

4.4.2.5 IdleTask

Class Idle task merupakan salah satu *child node* yang didalamnya berisikan perilaku sebuah NPC menjalankan aksi *idle. IdleTask* akan terus berjalan hingga dan mengeluarkan *output* animasi dengan pengembalian *true*, dan *child node* ini akan mengembalikan nilai *running* hingga terdapat *child node* lain yang memenuhi syarat pengembalian nilai *success* atau *running.*

4.4.2.6 CheckPlayerInRange

Class CheckPlayerInRange merupakan salah satu *child node* yang berisi pengecekan jarak antara *player* dengan NPC. Dalam kondisi ini apabila *collider player* dengan NPC saling bersentuhan dengan jarak yang telah ditetapkan maka akan dilakukan pengambilan data pada *player*. Namun apabila NPC belum memiliki data *player* yang tersimpan, objek *player* akan disimpan dalam variable dengan kata kunci "target", ke dalam *dictionary* yang terdapat pada *root node.* Proses inisiasi serta penyimpanan data ini dilakukan agar ke depannya apabila *collider* NPC bersentuhan dengan *collider player*, sistem dapat memanggil variabel "target" untuk mengevaluasi objek tersebut. *Child node* akan mengembalikan *success* apabila berhasil menambahkan data "target", dan akan mengembalikan *failure* apabila gagal mendeteksi objek yang berada dalam jangkauan NPC.

4.4.2.7 CheckNullData

Class CheckNullData merupakan *child node* yang berisi pengecekan kondisi apabila *player* belum memiliki nilai terkait pengalaman, *pretest*, ataupun keduanya, dalam hal ini sistem akan mengembalikan nilai *false* pada semua *behaviour* yang akan dijalankan, dikarenakan *player* belum memiliki kriteria yang cukup untuk memberikan data kepada NPC untuk menjalankan perubahan perilaku pada NPC. Namun apabila *player* telah memiliki nilai maka kondisi ini akan mengembalikan *failure* dan akan dilanjutkan pada pengecekan kondisi berikutnya.

4.4.2.8 NullDataTask

Class NullDataTask berisi perintah yang akan dieksekusi apabila kondisi *CheckNullData* terpenuhi. *NullDataTask* akan mengeluarkan *output* yang berupa perintah dalam bentuk dialog NPC kepada *player* untuk menyelesaikan tes terlebih dahulu.

4.4.2.9 CheckBehaviourGreet

Class CheckBehaviourGreet merupakan salah satu *child node* yang berisi pengecekan kondisi apakah NPC memenuhi syarat untuk melakukan aksi *greet.* Apabila NPC memenuhi syarat maka *child node* ini akan mengembalikan *running* atau *success,* lalu akan dilanjutkan evaluasi terhadap *child node* berikutnya. Pertama, *child node* ini akan mengecek objek yang berada dalam jangkauan NPC, apabila tidak terdapat objek dengan kata kunci "target" (*player*), maka akan dikembalikan dengan *failure.* Apabila *child node* berhasil mendeteksi objek dengan kata kunci "target" (*player),* akan dilakukan pengambilan hasil pengolahan data dalam *void* perhitunganDSS,

Hasil perhitungan DSS akan menjadi kunci untuk mengevaluasi perilaku *greet.* Pada perilaku *greet* akan digunakan Alt1 sebagai AltBehaviour. Berdasarkan perhitungan DSS yang didapatkan dari hasil pengolahan data *player*, jika perhitungan DSS menghasilkan nilai yang mengindikasikan alternatif 1 (*greet*) maka kondisi ini akan terpenuhi dan akan mengembalikan *running*, namun apabila berdasarkan hasil perhitungan DSS mengindikasikan nilai yang tidak sesuai dengan alternatif 1 (*greet*) maka akan dilakukan pengembalian berupa *failure.*

4.4.2.10 GreetTask

Class GreetTask merupakan *child node* yang berada pada *sequence node* yang sama dengan *CheckBehaviourGreet.* Apabila *CheckBehaviourGreet* berfungsi untuk melakukan evaluasi kondisi, maka *GreetTask* merupakan perintah perilaku yang akan dieksekusi oleh *tree*.

Sama seperti pada *CheckBehavioiurGreet, child node* ini berisi pengecekan objek yang berada pada jangkauan NPC, yang akan mengembalikan *running* jika terdapat kata kunci "target" dan akan mengembalikan *failure* apabila tidak mendeteksi kata kunci tersebut. Pada *child node* ini juga terdapat pengecekan kondisi apabila *player* berada dalam jarak NPC seperti yang telah ditetapkan (> 0.01f dan < 5f) maka NPC akan menjalankan animasi berbicara dengan pengembalian *true*, namun apabila jarak antara *player* dengan NPC (> 5f) maka animasi berbicara pada NPC akan dikembalikan dengan nilai *false*. Masing-masing kondisi tersebut akan tetap mengembalikan *running* pada *child node* yang artinya apabila *CheckBehaviourGreet* dan *GreetTask* mengembalikan *success* atau *running*, maka *child node* yang berada pada *sequence* yang sama ini akan dieksekusi oleh tree.

4.4.2.11 CheckBehaviourApproach

Class CheckBehaviourApproach merupakan salah satu *child node* yang berisi pengecekan kondisi apakah NPC memenuhi syarat untuk melakukan aksi *Approach.* Apabila NPC memenuhi syarat maka *child node* ini akan mengembalikan *running* atau *success*, dan akan dilanjutkan evaluasi terhadap *child node* berikutnya.

Child node ini memiliki proses yang hampir sama dengan *CheckBehaviourGreet.* Perbedaannya terletak pada penentuan hasil perhitungan DSS yang akan dieksekusi menjadi perilaku *Approach.* Pada perilaku *Approach* akan digunakan Alt2 sebagai AltBehaviour. Berdasarkan perhitungan DSS yang didapatkan dari hasil pengolahan data *player*, jika perhitungan DSS menghasilkan nilai yang mengindikasikan alternatif 2 (*Approach*) maka kondisi ini akan terpenuhi dan akan mengembalikan *running,* namun apabila berdasarkan hasil perhitungan DSS mengindikasikan nilai yang tidak sesuai dengan alternatif 2 (*Approach*) maka akan dilakukan pengembalian berupa *failure.*

4.4.2.12 ApproachTask

Class ApproachTask merupakan *child node* yang berada pada *sequence node* yang sama dengan *CheckBehaviourApproach.* Apabila *CheckBehaviouApproach* berfungsi untuk melakukan evaluasi kondisi, maka *ApproachTask* merupakan perintah perilaku yang akan dieksekusi oleh *tree*. Sama seperti pada *CheckBehavioiurApproach, child node* ini berisi pengecekan objek yang berada pada jangkauan NPC, yang akan mengembalikan *running* jika terdapat kata kunci “target” dan akan mengembalikan *failure* apabila tidak mendeteksi kata kunci tersebut.

Pada *child node* ini juga terdapat pengecekan kondisi apabila *player* berada dalam jarak NPC seperti yang telah ditetapkan (< 3f) maka animasi berbicara pada NPC akan dikembalikan dengan *true,* dan pada animasi berjalan akan dikembalikan dengan *false.* Namun apabila jarak antara *player* dengan NPC (> 3f dan < 10f) maka animasi berbicara pada NPC akan dikembalikan dengan *false,* dan animasi berjalan

akan dikembalikan dengan *true* hingga *player* berada jarak yang cukup seperti yang telah diinisiasikan agar animasi berbicara pada NPC mengembalikan *true.* Setiap kondisi tersebut akan mengembalikan nilai *running* yang dimana akan dijalankan apabila setiap *child node* dalam *sequence* yang sama mengembalikan nilai *success* atau *running.*

4.4.2.13 CheckBehaviourFollow

Class CheckBehaviourFollow merupakan salah satu *child node* yang berisi pengecekan kondisi apakah NPC memenuhi syarat untuk melakukan aksi *Follow.* Apabila NPC memenuhi syarat maka *child node* ini akan mengembalikan *running* atau *success,* dan akan dilanjutkan evaluasi terhadap *child node* berikutnya.

Child node ini memiliki proses evaluasi data yang dimana hasil perhitungan DSS akan dieksekusi menjadi perilaku *follow*. Pada perilaku *follow* akan digunakan Alt3 sebagai AltBehaviour. Berdasarkan perhitungan DSS yang didapatkan dari hasil pengolahan data *player*, jika perhitungan DSS menghasilkan nilai yang mengindikasikan alternatif 3 (*Follow*) maka kondisi ini akan terpenuhi dan akan mengembalikan *running*, namun apabila berdasarkan hasil perhitungan DSS mengindikasikan nilai yang tidak sesuai dengan alternatif 2 (*Follow*) maka akan dilakukan pengembalian berupa *failure.*

4.4.2.14 FollowTask

C*lass FollowTask* merupakan *child node* yang berada pada *sequence node* yang sama dengan *CheckBehaviourFollow*. Apabila *CheckBehaviouFollow* berfungsi untuk melakukan evaluasi kondisi, maka *FollowTask* merupakan perintah

perilaku yang akan dieksekusi oleh *tree*. Sama seperti pada *CheckBehavioiurFollow, child node* ini berisi pengecekan objek yang berada pada jangkauan NPC, yang akan mengembalikan *running* jika terdapat kata kunci "target" dan akan mengembalikan *failure* apabila tidak mendeteksi kata kunci tersebut.

Pada *child node* ini juga terdapat pengecekan kondisi apabila *player* berada dalam jarak NPC seperti yang telah ditetapkan (< 3f) maka animasi berbicara pada NPC akan dikembalikan dengan *true,* dan pada animasi berlari akan dikembalikan dengan *false.* Namun apabila jarak antara *player* dengan NPC (> 3f dan < 10f) maka animasi berbicara pada NPC akan dikembalikan dengan *false,* dan animasi berlari akan dikembalikan dengan *true* hingga *player* berada jarak yang cukup seperti yang telah diinisiasikan agar animasi berbicara pada NPC mengembalikan *true.* Setiap kondisi tersebut akan mengembalikan nilai *running* yang dimana akan dijalankan apabila setiap *child node* dalam *sequence* yang sama mengembalikan nilai *success* atau *running.*

4.4.2.15 ActionBT

Class ActionBT merupakan *class* yang *inherit* atau berhubungan langsung dengan *class tree. Class* ActionBT berisikan urutan perilaku yang akan di evaluasi berdasarkan tingkatan prioritas tertinggi hingga terendah. Pada *class* ini juga berisi inisiasi variabel yang telah ditetapkan pada tiap-tiap *child node.*

```
using System.Collections.Generic;
using BehaviourTree;
public class ActionBT : Tree
{
    public UnityEngine.GameObject _player;
```

```
    public UnityEngine.GameObject _npc;
    public PlayerData _playerData;
    public static float playerRange = 10f;
    public static float npcWalkingSpeed = 3f; //
Adjust this value to set the NPC's speed
    public static float npcRunningSpeed = 7f;

    protected override Node SetupTree()
    {
       Node root = new Selector(new List<Node>
        {
            new Sequence(new List<Node>
            {
                new CheckPlayerInRange(_player, _npc,
transform),
                new DefaultTask(_player, _npc,
transform),
            }),
            new Sequence(new List<Node>
            {
                new CheckBehaviourFollow(_playerData,
_player, _npc, transform),
                new FollowTask(_playerData, _player,
_npc, transform),
            }),
            new Sequence(new List<Node>
            {
                new
CheckBehaviourApproach(_playerData, _player, _npc,
transform),
                new ApproachTask(_playerData,
_player, _npc, transform),
            }),
            new Sequence(new List<Node>
            {
                new CheckBehaviourGreet(_playerData,
_player, _npc, transform),
                new GreetTask(_playerData, _player,
_npc, transform),
            }),
            new IdleTask(_player, _npc, transform),
        });
        return root;
    }
}
```

Pada *class* ActionBT dapat dilihat bahwa tingkat prioritas tertinggi di mulai dengan *sequence* tingkat pertama yaitu evaluasi jarak antara *player* dengan NPC. Pada *sequence* tingkat kedua, berisi evaluasi kondisi *alternatif behaviour 1 (greet)*. Pada *sequence* tingkat ketiga, berisi evaluasi kondisi *alternatif behaviour 2 (approach)* .Pada *sequence* tingkat keempat, berisi evaluasi kondisi *alternatif behaviour 3 (follow).* Tingkat evaluasi terakhir merupakan *idleTask* yaitu perilaku yang dijadikan *task* yang memiliki pengembalian *running* dan akan terus berjalan apabila kondisi pada tingkat prioritas tertinggi hingga terendah tidak mengembalikan nilai *running* atau *success.*

4.4.3 Hubungan *Behaviour Tree* Dengan *Simple Additive Weighting*

Hubungan antara *behaviour tree* dengan *decision support system model simple additive weighting* terletak pada kombinasi penentuan tingkat prioritas dan pengecekan kondisi *behaviour* yang akan dieksekusi oleh NPC. Dalam hal ini digunakan *class CheckBehaviourGreet, CheckBehaviourApproach, dan CheckBehaviourFollow* yang merupakan bagian dari *leaf node* pada *behaviour tree* yang di dalamnya diimplementasikan pula perhitungan *decision support system.* Tiap-tiap *class* tersebut merupakan *condition node* yang berfungsi untuk mengecek pengkondisian, dan akan mengembalikan *success* atau *failure* yang bergantung kepada kondisi yang terpenuhi.

```
public override NodeState Evaluate()
    {
        hasTest = _playerData.hasTestData;
        hasExp = _playerData.hasExpData;
        object t = GetData("target");
        if (t == null)
        {
```

```
            state = NodeState.FAILURE;
            return state;
        }

        Transform target = (Transform)t;
        if (hasTest == false && hasExp == false)
        {
            if  (Vector3.Distance(_transform.position,
_player.transform.position) < 5f)
            {
                _animator.SetBool("Talking", false);
                _animator.SetBool("Arguing", false);
                _animator.SetBool("Yelling", false);
                _animator.SetBool("Walking", false);
                _animator.SetBool("Running", false);
            }
            state = NodeState.RUNNING;
            return state;
        }
        state = NodeState.FAILURE;
        return state;
    }
```

Source code tersebut merupakan pengecekan kondisi apabila *player* belum memiliki data pengalaman dan pretest. Maka dari itu tidak ada alternatif yang didapatkan, karena kriteria yang dibutuhkan dalam perhitungan untuk mendapatkan alternatif masih belum ada. Karena belum memiliki alternatif *behaviour*, maka seluruh perilaku pada NPC akan diatur dengan mengembalikan *false.*

```
public override NodeState Evaluate()
    {
        hasTest = _playerData.hasTestData;
        hasExp = _playerData.hasExpData;
        object t = GetData("target");
        if (t == null)
        {
            state = NodeState.FAILURE;
            return state;
        }
```

```
        Transform target = (Transform)t;

        if (hasTest == true && hasExp == true &&
AltBehaviour == Alt1)
        {
            if (Vector3.Distance(_transform.position,
_player.transform.position) < 5f)
            {
                _animator.SetBool("Arguing", false);
                _animator.SetBool("Yelling", false);
                _animator.SetBool("Walking", false);
                _animator.SetBool("Running", false);
                PerhitunganDSS();
            }
            state = NodeState.RUNNING;
            return state;
        }
        state = NodeState.FAILURE;
        return state;

    }
```

Source code tersebut merupakan *void nodestate evaluate()* yang dalam hal ini merupakan *void* pengeksekusi yang diinisiasi dalam sistem *behaviour tree.* Di dalam *void* ini dimulai dengan pengambilan data dengan kata kurci target yang dimana dalam hal ini mengacu pada objek NPC. Setelah berhasil mendapatkan data dari target, maka dilanjut dengan pengecekan alternatif yang akan menjadi *behaviour* NPC. AltBehaviour didapat dari perhitungan DSS yang sebelumnya telah dilakukan.

Pada *void nodestate evaluate* tersebut, nilai AltBehaviour akan dipanggil dan digunakan sebagai parameter pengkondisian terhadap nilai Alternatif 1 (Alt1), apabila nilai AltBehaviour sama dengan nilai Alternatif 1 (Alt1), maka akan dilanjutkan dengan pengecekan jarak NPC dengan *player* yang dimana apabila

jarak antara kedua objek ini memenuhi kondisi yang telah ditetapkan, maka NPC akan mengeksekusi *state behaviour* yang telah diinisiasikan yaitu *Greet.*

```
public override NodeState Evaluate()
    {
        hasTest = _playerData.hasTestData;
        hasExp = _playerData.hasExpData;

        object t = GetData("target");
        if (t == null)
        {
            state = NodeState.FAILURE;
            return state;
        }
        Transform target = (Transform)t;

        if (hasTest == true && hasExp == true &&
AltBehaviour == Alt2)
        {
            if (Vector3.Distance(_transform.position,
_player.transform.position) < 3f)
            {
                _animator.SetBool("Talking", false);
                _animator.SetBool("Yelling", false);
                _animator.SetBool("Running", false);
                PerhitunganDSS();
            }
            state = NodeState.RUNNING;
            return state;
        }
        state = NodeState.FAILURE;
        return state;
    }
```

Pada *void nodestate evaluate* diatas, nilai AltBehaviour akan dipanggil dan digunakan sebagai parameter pengkondisian terhadap nilai Alternatif 2 (Alt2), apabila nilai AltBehaviour sama dengan nilai Alternatif 2 (Alt2), maka akan dilanjutkan dengan pengecekan jarak NPC dengan *player* yang dimana apabila

jarak antara kedua objek ini memenuhi kondisi yang telah ditetapkan, maka NPC akan mengeksekusi *state behaviour* yang telah diinisiasikan yaitu *Approach.*

```
public override NodeState Evaluate()
    {
        hasTest = _playerData.hasTestData;
        hasExp = _playerData.hasExpData;

        object t = GetData("target");
        if (t == null)
        {
            state = NodeState.FAILURE;
            return state;
        }

        Transform target = (Transform)t;
        if (hasTest == true && hasExp == true &&
AltBehaviour == Alt3)
        {
            if (Vector3.Distance(_transform.position,
_player.transform.position) < 3f)
            {
                _animator.SetBool("Talking", false);
                _animator.SetBool("Arguing", false);
                _animator.SetBool("Walking", false);
                PerhitunganDSS();
            }
            state = NodeState.RUNNING;
            return state;
        }
        state = NodeState.FAILURE;
        return state;
    }
```

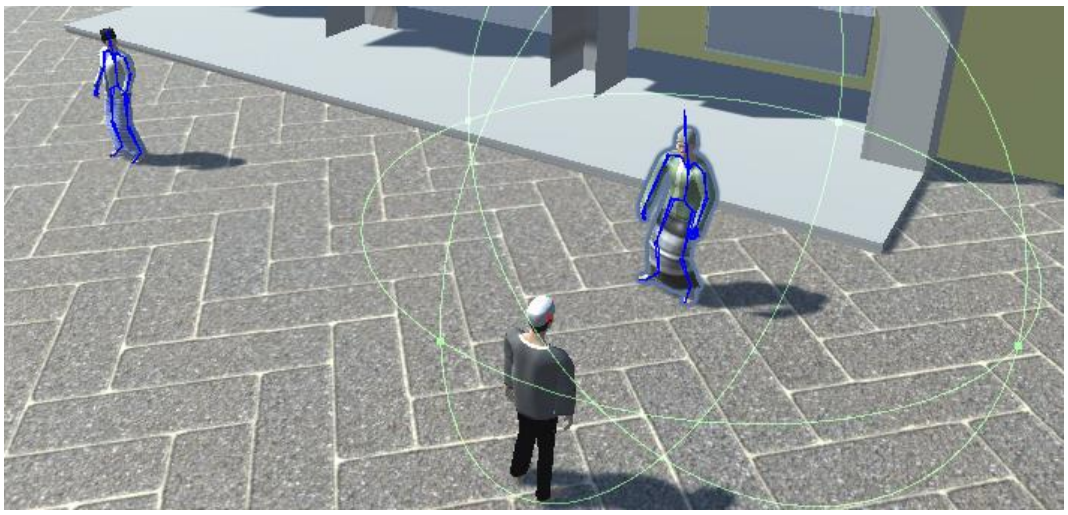
Pada *void nodestate evaluate*, nilai AltBehaviour akan dipanggil dan digunakan sebagai parameter pengkondisian terhadap nilai Alternatif 3 (Alt3), apabila nilai AltBehaviour sama dengan nilai Alternatif 3 (Alt3), maka akan dilanjutkan dengan pengecekan jarak NPC dengan *player* yang dimana apabila

jarak antara kedua objek ini memenuhi kondisi yang telah ditetapkan, maka NPC akan mengeksekusi *state behaviour* yang telah diinisiasikan yaitu *Follow*.

Source code diatas merupakan contoh bentuk kombinasi antara *behaviour tree* dengan *decision support system. Source code* diatas, merupakan *class CheckBehaviourGreet, CheckBehaviourApproach, dan CheckBehaviourFollow* yang didalamnya terdapat inisiasi serta perhitungan DSS. seluruh *source code* diletakkan secara tersusun mulai dari pertama hingga terakhir. Berdasarkan sifat *class* yang dimiliki *behavuiour tree* yaitu *protected,* maka proses perhitungan DSS dilakukan pada tiap-tiap *selector class* dalam *behaviour tree.*

4.5 Pengujian *Non-Playable Character*

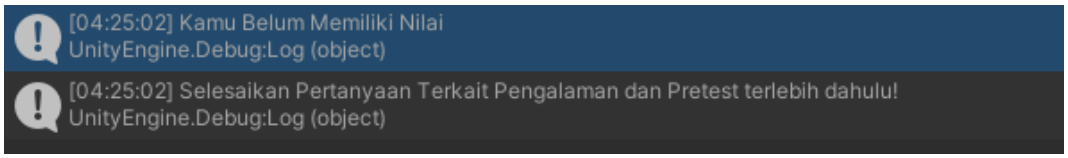
Pengujian *behaviour* NPC dalam permainan ini dilakukan dengan memberikan data set yang berbeda pada setiap percobaan yang memunculkan alternatif yang mengacu pada *behaviour* tersebut, hal ini dilakukan untuk mengukur akurasi serta validasi kesesuaian antara alternatif yang memiliki *value* tertinggi dengan *output behaviour* yang harus dieksekusi.



Gambar 4.6 *Collider* yang teradapat pada NPC

Berdasarkan Gambar 4.6 dapat terlihat, pada NPC telah terpasang *collider* dengan besar yang sudah ditetapkan. NPC akan menjalankan perintah sistem seperti pengambilan data pada *player*, perhitungan kriteria, hingga proses eksekusi alternatif *behaviour* apabila *player* memasuki jangkauan atau *collider* NPC

4.5.1 Pengujian Tanpa Data



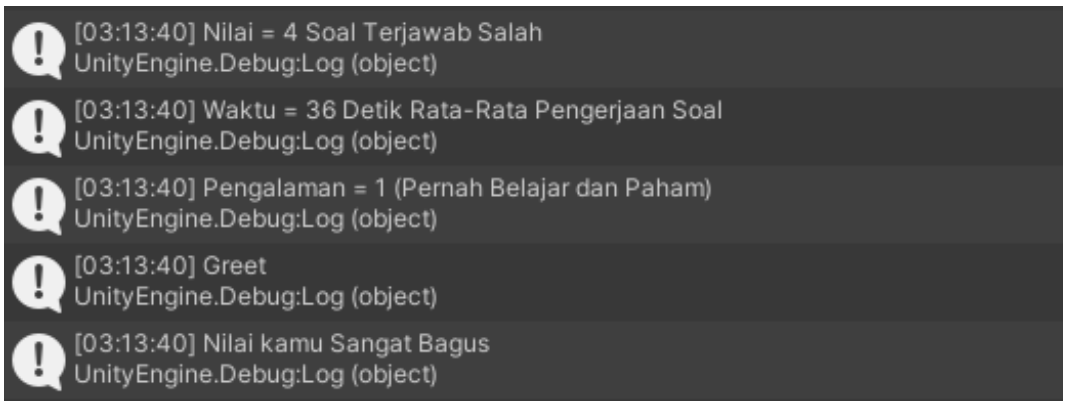
Gambar 4.7 *Output* pada *Console* Pengujian Tanpa Data

Gambar 4.7 merupakan *output* yang ditampilkan dari *console Unity*, *output* tersebut ditampilkan berdasarkan kondisi dimana *player* belum melaksanakan serangkaian test sehingga belum memiliki nilai yang menjadi kriteria dalam perubahan *behaviour* NPC.

Gambar 4.8 *Output Behaviour* Tanpa Data

Pada Gambar 4.8 *player* akan memasuki *collider* NPC dengan cara mendekat kearah NPC pada saat *player* belum melakukan tes apapun, sehingga belum ada data kriteria yang tersimpan pada *player*. NPC akan mengakses script *playerData* pada *player* apabila tidak terdapat *value* pada kriteria nilai, waktu dan pengalaman, maka NPC akan bersifat statis dan hanya akan mengeluarkan *output* berupa perintah untuk melaksanakan test.

4.5.2 Pengujian *Behaviour Greet*



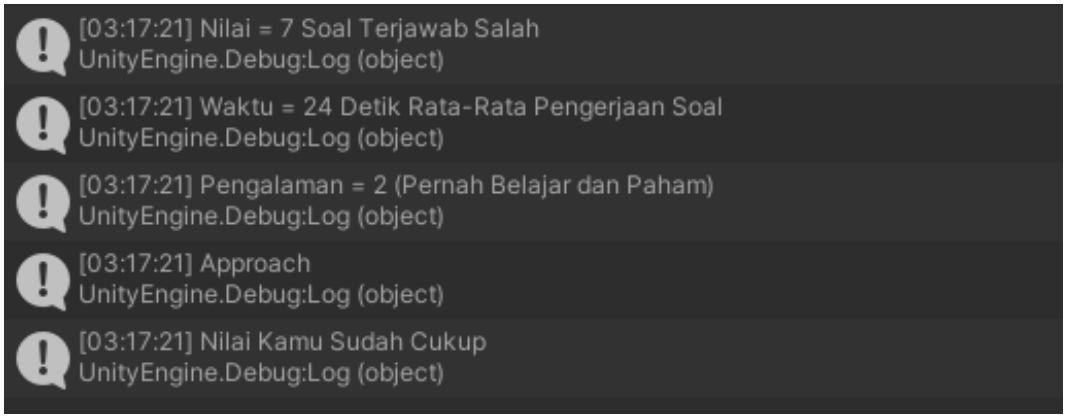
Gambar 4.9 *Output* Pada *Console Behaviour (Greet)*

Gambar 4.9 merupakan *output* yang ditampilkan dari *console Unity*, *output* yang ditampilkan berupa *value* pada-masing-masing kriteria, dan *output behaviour* yang akan dijalankan oleh NPC yaitu *behaviour greet*.

Gambar 4.10 *Output Behaviour (Greet)*

Pada Gambar 4.10 dapat dilihat, *player* akan memasuki *collider* NPC dengan mendekat kearah NPC. Setelah melaksanakan serangkaian test yang pada tahap ini, *player* telah memiliki *value* pada kriteria nilai, waktu, dan pengalaman dalam kondisi yang sesuai dengan alternatif 1. *Output* yang akan dieksekusi oleh NPC berupa *behaviour greet* dimana NPC akan menjalankan animasi menyapa *player*.

4.5.3 Pengujian *Behaviour Approach*



Gambar 4.11 *Output* pada *Console Behaviour* (*Approach*)

Gambar 4.11 Merupakan *output* yang ditampilkan dari *console Unity*, *output* yang ditampilkan berupa *value* pada-masing-masing kriteria, dan *output behaviour* yang akan dijalankan oleh NPC yaitu *behaviour approach*.

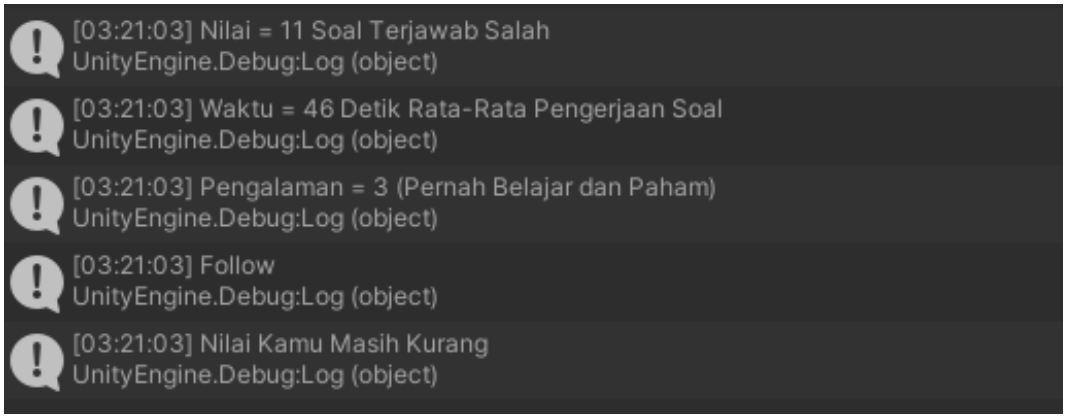
Gambar 4.12 *Arguing Animation* Pada Pengujian *Behaviour (Approach)*

Pada Gambar 4.12 dapat dilihat, *player* memasuki *collider* NPC setelah melaksanakan serangkaian test dan pada pengujian ini *value* pada kriteria nilai, waktu dan pengalaman di set sehingga sesuai dengan alternatif 2. *Output* yang akan dieksekusi oleh NPC berupa *behaviour approach* dimana NPC akan menjalankan animasi berupa memberikan nasihat.

Gambar 4.13 *Walking Animation* Pada Pengujian *Behaviour (Approach)*

Pada Gambar 4.13 dapat dilihat, apabila *player* bergerak menjauhi NPC secara perlahan NPC tersebut akan terus berjalan menghampiri *player* dan akan kembali menasihati *player* hingga sesuai dengan jarak yang telah ditetapkan. Apabila jarak antara *player* dengan NPC mencapat jarak yang telah ditentukan maka NPC akan diam dan kembali menjalankan *arguing animation.*

4.5.4 Pengujian *Behaviour Follow*



Gambar 4.14 *Output* pada *Console Behaviour* (*Approach)*

Gambar 4.14 Merupakan *output* yang ditampilkan dari *console Unity*, *output* yang ditampilkan berupa *value* pada-masing-masing kriteria, dan *output behaviour* yang akan dijalankan oleh NPC yaitu *behaviour approach*.

Gambar 4.15 *Yelling Animation* Pada Penggujian *Behaviour (Follow)*

Gambar 4.15 *player* memasuki *collider* NPC setelah melaksanakan serangkaian test dan pada pengujian ini *value* pada kriteria nilai, waktu dan pengalaman di set sehingga sesuai dengan alternatif 3. *Output* yang akan dieksekusi oleh NPC yaitu *behaviour follow* dimana NPC akan menjalankan animasi memarahi *player* dan memerintah *player* untuk segera melakukan pembelajaran.

Gambar 4.16 *Running Animation* Pada Penggujian *Behaviour (Follow)*

Pada Gambar 4.16 dapat dilihat apabila *player* bergerak menjauhi NPC, maka NPC akan terus berlari mengejar *player* dan akan berhenti pada jarak yang telah ditetapkan dan kembali menjalankan *yelling animation*..

4.6 Pengujian Usability

Pengujian yang akan dilakukan terbagi menjadi ke dalam dua hal yaitu pengujian sistem dan pengujian terhadap pengguna. Pengujian sistem dilakukan untuk menguji tingkat akurasi serta validitas antara pengolahan data dengan kesesuaian *output* yang dieksekusi. Pengujian terhadap pengguna dilakukan untuk menguji kemudahan penggunaan serta pemahaman yang dilakukan oleh pengguna (*user friendly*).

4.6.1 Skenario Pengujian Sistem

Pengujian sistem dilakukan dengan mencoba sistem dengan cara memainkan permainan sehingga mendapatkan 50 data set yang berbeda. Berdasarkan data set tersebut akan dilakukan evaluasi antara kesesuaian alternatif dengan *output* yang dieksekusi serta dapat berjalan dengan baik berdasarkan hasil analisa yang telah dilakukan sebelumnya. Keterangan validasi akan menggambarkan tingkat keakuratan serta kesesuaian antara kriteria dengan *output* yang dieksekusi oleh sistem.

Tabel 4.1 Pengujian *Behaviour* Pada NPC

Pengujian Sistem	Nilai	Waktu	Pengalaman	Alternatif	*Behaviour*	Validasi
1	3	31	1	1	*Greet*	Valid
2	7	29	2	2	*Approach*	Valid
3	2	12	3	1	*Greet*	Valid
4	9	45	2	3	*Follow*	Valid
5	5	18	1	2	*Approach*	Valid
6	11	53	3	3	*Follow*	Valid

Lanjutan Tabel 4.1

7	8	58	3	3	*Follow*	Valid
8	4	37	1	1	*Greet*	Valid
9	6	24	2	2	*Approach*	Valid
10	12	58	1	3	*Follow*	Valid
11	1	43	2	1	*Greet*	Valid
12	10	32	3	3	*Follow*	Valid
13	6	15	3	2	*Approach*	Valid
14	2	48	1	1	*Greet*	Valid
15	11	21	2	3	*Follow*	Valid
16	9	56	3	3	*Follow*	Valid
17	3	10	2	1	*Greet*	Valid
18	5	39	1	2	*Approach*	Valid
19	8	26	1	2	*Approach*	Valid
20	1	60	3	3	*Follow*	Valid
21	10	16	2	3	*Follow*	Valid
22	7	30	3	2	*Approach*	Valid
23	4	13	1	1	*Greet*	Valid
24	12	46	2	3	*Follow*	Valid
25	2	19	2	1	*Greet*	Valid
26	8	54	3	3	*Follow*	Valid
27	5	17	1	2	*Approach*	Valid
28	9	36	3	3	*Follow*	Valid
29	1	23	1	1	*Greet*	Valid
30	7	57	3	3	*Follow*	Valid
31	6	24	2	2	*Approach*	Valid
32	11	31	3	3	*Follow*	Valid
33	3	14	2	1	*Greet*	Valid
34	10	47	1	3	*Follow*	Valid
35	4	20	3	1	*Greet*	Valid
36	12	55	2	3	*Follow*	Valid
37	2	19	1	1	*Greet*	Valid
38	7	38	1	2	*Approach*	Valid
39	5	25	2	2	*Approach*	Valid
40	8	59	3	3	*Follow*	Valid
41	9	33	2	3	*Follow*	Valid
42	6	28	1	2	*Approach*	Valid
43	1	11	3	1	*Greet*	Valid
44	11	44	1	3	*Follow*	Valid
45	3	17	2	1	*Greet*	Valid
46	12	52	3	3	*Follow*	Valid
47	4	36	1	1	*Greet*	Valid
48	2	35	2	2	*Approach*	Valid
49	10	22	3	3	*Follow*	Valid
50	8	50	2	2	*Approach*	Valid

Terlihat pada Tabel 4.1 terdapat hasil pengujian yang dilakukan sebanyak 50 kali pengujian, dalam tabel tersebut dapat dilihat terdapat 50 data set yang memiliki *value* yang berbeda pada masing-masing kriteria. Seperti yang telah dijelaskan dalam pembahasan sebelumnya, alternatif yang akan digunakan terbagi menjadi tiga, yaitu alternatif 1, alternatif 2, dan alternatif 3. Masing-masing alternatif direpresentasikan dengan *behaviour* yang menjadi output terhadap perilaku NPC. Alternatif 1 direpresentasikan dengan *behaviour greet*, alternatif 2 direpresentasikan dengan *behaviour approach*, dan alternatif 3 direpresentasikan dengan *behaviour follow*. Berdasarkan hasil pengujian, sebanyak 50 data yang diuji menghasilkan alternatif yang sesuai berdasarkan perhitungan *decision support system model simple additive weighting* dan output yang dieksekusi oleh sistem juga sesuai dengan sistem penentuan *behaviour* yang menggunakan metode *behaviour tree* dan *simple additive weighting.*

Dapat terlihat dalam Tabel 4.1, sebanyak 50 kali percobaan pengujian menghasilkan *output* serta menampilkan animasi *non-playable character* yang sesuai dengan data set. Dengan hal ini dapat disimpulkan bahwa sistem ini berjalan dengan sempurna tanpa adanya permasalahan dalam perhitungan maupun pengeksekusian output.

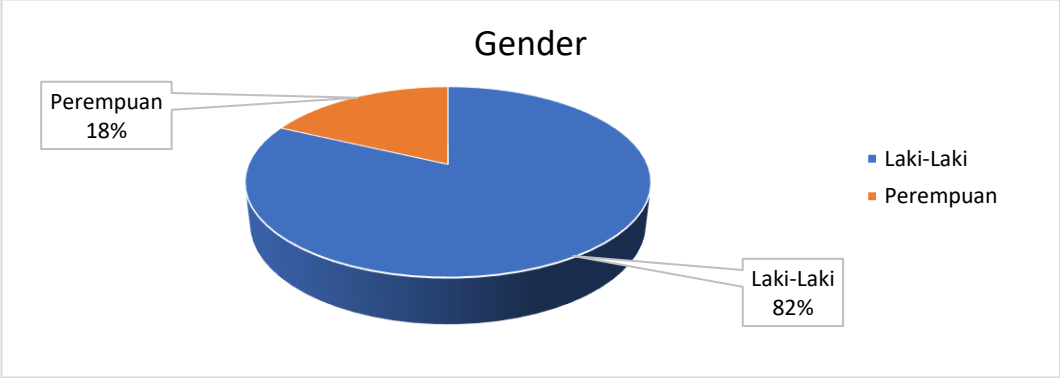
4.6.2 *Usability Testing*

Usability testing merupakan salah satu faktor yang penting dan berpengaruh terhadap pengembangan sebuah sistem. Sistem akan dikatakan bagus apabila mudah digunakan dan dimengerti oleh pengguna, maka dari itu dilakukan pula pengujian sistem terhadap pengguna.

Tabel 4.2 Demografi Responden

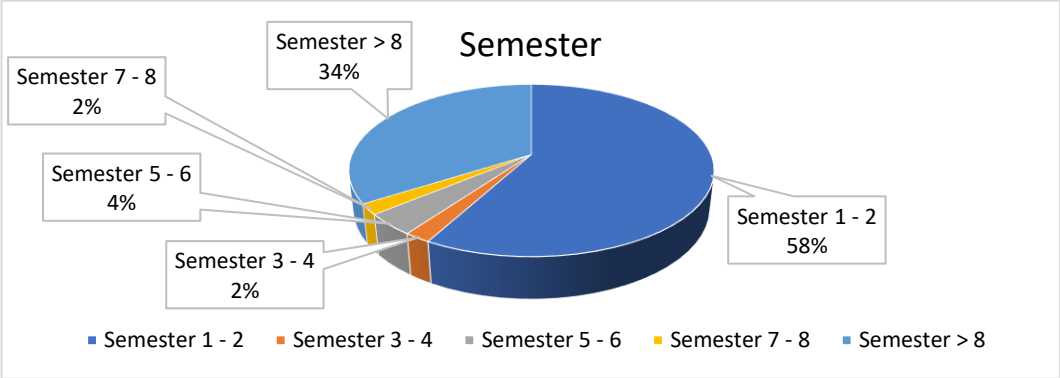
Karakteristik	Keterangan	Jumlah Responden	Persentase
Gender	Laki-laki	41	82%
	Perempuan	9	18%
Semester	1 – 2	29	58%
	3 – 4	1	2%
	5 – 6	2	4%
	7 – 8	1	2%
	> 8	17	34%
Jurusan	Teknik Informatika	23	46%
	Ilmu Al Qur'an dan Tafsir	2	4%
	Hukum Tata Negara	1	2%
	Manajemen	2	4%
	Perpustakaan dan Ilmu Informasi	5	10%
	Pendidikan Ilmu Pengetahuan	1	2%
	Sosial	3	6%
	Hukum keluarga Islam	1	2%
	Akuntansi	2	4%
	Pendidikan Bahasa Arab	1	2%
	Matematika	1	2%
	Sastra Inggris	1	2%
	Biologi	1	2%
	Teknik Arsitektur	2	4%
	Psikologi	2	4%
	Pendidikan Agama Islam	1	2%
	Fisika	1	2%
	Tadris Bahasa Inggris	1	2%

Berdasarkan Tabel 4.2 terdapat pengelompokan demografi responden menjadi beberapa kategori dengan *range* keterangan yang berbeda. Berikut merupakan hasil analisis dalam bentuk grafik pada masing-masing kategori yang telah ditetapkan.



Gambar 4.17 Grafik *Gender* Responden

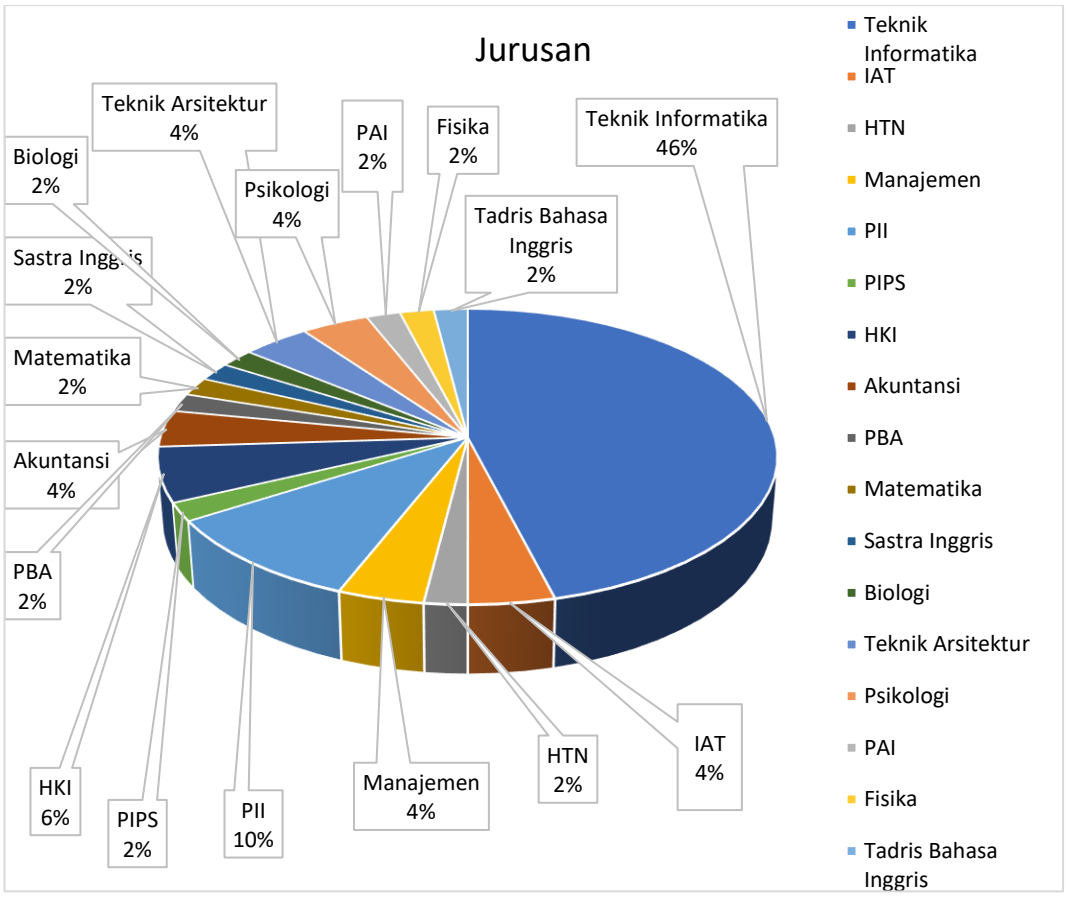
Berdasarkan Gambar 4.17 dapat dilihat dari total seluruh responden, terdapat 43 responden berjenis kelamin laki-laki dengan besar persentase 82% dan 7 responden wanita dengan besar persentase 18%. Demografi dengan membagi ke dalam dua *range* keterangan ini diharapkan dapat memberikkan variasi respon terhadap permainan maupun sistem.



Gambar 4.18 Grafik Usia Responden

Berdasarkan Gambar 4.18 dapat dilihat dari total seluruh responden, terdapat mahasiswa semester 1 - 2 dengan persentase 58% dari total 50 mahasiswa, yang berarti terdapat 29 mahasiswa yang berada pada tingkat tersebut. Terdapat 1 mahasiswa semester 3 - 4 dengan persentase 2%. Terdapat 2 mahasiswa semester 5

- 6 dengan persentase 4%. Terdapat 1 mahasiswa semester 7 - 8 dengan persentase 2%. Selebihnya, data responden diisi oleh mahasiswa semester lebih dari 8 dengan persentase 34% dari jumlah total 50 responden, yang berarti terdapat 17 mahasiswa pada tingkat tersebut.



Gambar 4.19 Grafik Jurusan Responden

Berdasarkan Gambar 4.19 dapat dilihat terdapat grafik terkait jurusan dari masing-masing responden. Berdasarkan hasil survey yang dilakukan dapat dilihat mayoritas responden berasal dari jurusan Teknik Informatika dengan besar persentase 46%, terdapat responden yang berasal dari jurusan Pendidikan Agama Islam dengan persentase 2%, Psikologi dengan persentase 4%, Teknik Arsitektur

dengan persentase 4%, Biologi dengan persentase 2%, Sastra Inggris dengan persentase 2%, Matematika dengan persentase 2%, Akuntansi dengan persentase 4%, Pendidikan Bahasa Arab dengan persentase 2%, Hukum Keluarga Islam dengan persentase 6%, Pendidikan Ilmu Pengetahuan Sosial dengan persentase 2%, Perpustakaan dan Ilmu Informasi dengan persentase 10%, Manajemen dengan persentase 4%, Hukum Tata Negara dengan persentase 2%, Ilmu Al-Qur'an dan Tafsir dengan persentase 4%, Tadris Bahasa Inggris dengan persentase 2%, dan Fisika dengan persentase 2%. Variasi jurusan dari masing-masing responden diharapkan mendapatkan tanggapan serta respon yang lebih bervariatif berdasarkan perspektif responden.

Tabel 4.3 Survey *Usability* (Rizky & Pudrianisa, 2019)

No.	Pertanyaan	Kategori
1	Game ini mudah dipahami aturan permainannya	*Learnability*
2	Tampilan visual *game* ini mudah dipahami	
3	Aplikasi mudah untuk dinavigasikan	*Efficiency*
4	Aplikasi ini memungkinkan saya dengan cepat menemukan apa yang saya butuhkan	
5	Menu dan tampilan halaman aplikasi mudah diingat	*Memorability*
6	Saya Memahami Aplikasi ini dengan cepat	
7	Saya tidak menemukan bug (kesalahan) di dalam *game*	*Errors*
8	Saya tidak menemukan kesalahan informasi dari aplikasi yang diberikan	
9	Saya akan merekomendasikan orang lain untuk memainkan *game* ini	*Satisfaction*
10	Saya merasa nyaman dengan model belajar menggunakan *game* ini	

Pada Tabel 4.3 dapat dilihat merupakan pertanyaan yang akan diberikan kepada responden berserta setiap pembagian kategori terkait *usability testing.* Tiap poin katergori dibatasi menjadi masing-masing sebanyak 2 butir pertanyaan

sehingga pada seluruh kategori terdapat total 10 pertanyaan yang akan diberikan kepada responden.

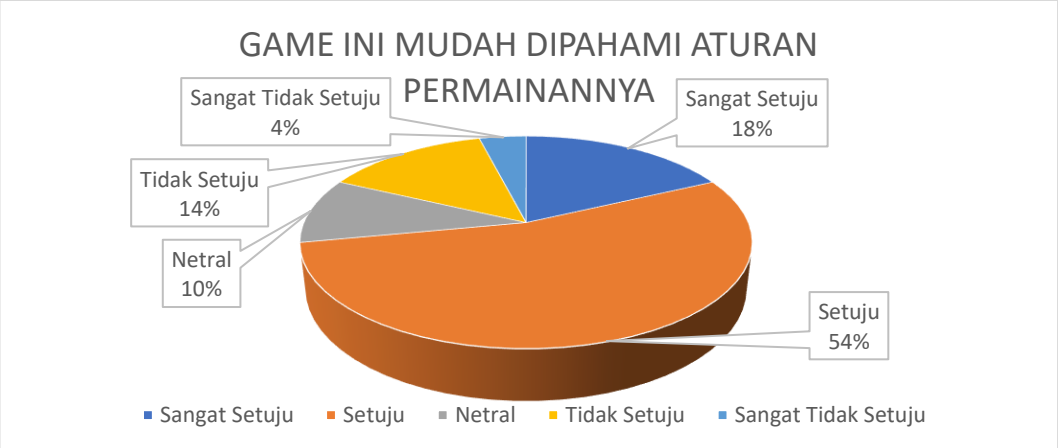
Tabel 4.4 Tanggapan Responden Terkait *Usability Testing*

No.	Pertanyaan	Tanggapan	Responden	Persentase
1	*Game* ini mudah dipahami aturan permainannya	Sangat Setuju	9	18%
		Setuju	27	54%
		Netral	5	10%
		Tidak Setuju	7	14%
		Sangat Tidak Setuju	2	4%
2	Tampilan visual *game* ini mudah dipahami	Sangat Setuju	14	28%
		Setuju	20	40%
		Netral	7	14%
		Tidak Setuju	7	14%
		Sangat Tidak Setuju	2	4%
3	Aplikasi mudah untuk dinavigasikan	Sangat Setuju	14	28%
		Setuju	20	40%
		Netral	11	22%
		Tidak Setuju	3	6%
		Sangat Tidak Setuju	2	4%
4	Aplikasi ini memungkinkan saya dengan cepat menemukan apa yang saya butuhkan	Sangat Setuju	8	16%
		Setuju	21	42%
		Netral	15	30%
		Tidak Setuju	5	10%
		Sangat Tidak Setuju	1	2%
5	Menu dan tampilan halaman aplikasi mudah diingat	Sangat Setuju	14	28%
		Setuju	23	46%
		Netral	9	18%
		Tidak Setuju	3	6%
		Sangat Tidak Setuju	1	2%
6	Saya Memahami Aplikasi ini dengan cepat	Sangat Setuju	14	28%
		Setuju	20	40%
		Netral	9	18%
		Tidak Setuju	6	12%
		Sangat Tidak Setuju	1	2%
7	Saya tidak menemukan bug (kesalahan) di dalam *game*	Sangat Setuju	10	20%
		Setuju	20	40%
		Netral	12	24%
		Tidak Setuju	6	12%
		Sangat Tidak Setuju	2	4%

Lanjutan Tabel 4.4

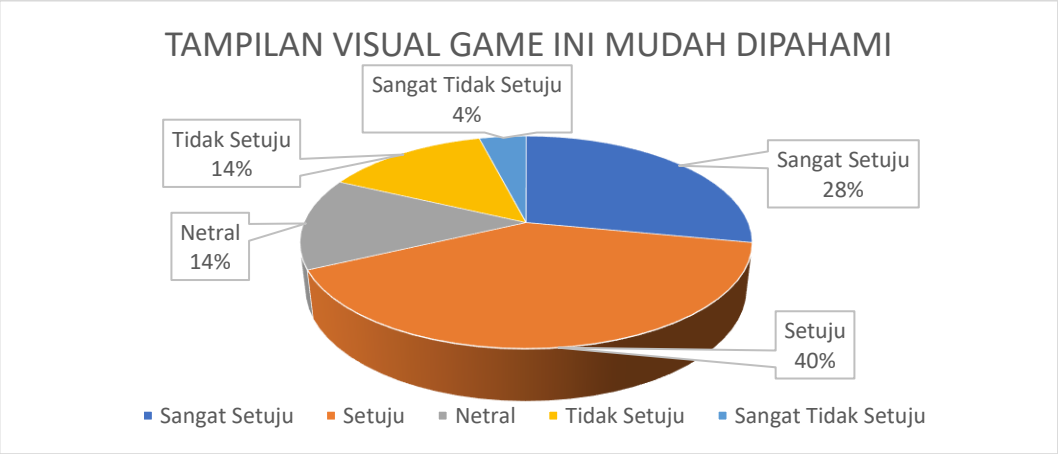
8	Saya tidak menemukan kesalahan informasi dari aplikasi yang diberikan	Sangat Setuju Setuju Netral Tidak Setuju Sangat Tidak Setuju	14 22 9 4 1	28% 44% 18% 8% 2%
9	Saya akan merekomendasikan orang lain untuk memainkan *game* ini	Sangat Setuju Setuju Netral Tidak Setuju Sangat Tidak Setuju	11 21 9 7 2	22% 42% 18% 14% 4%
10	Saya merasa nyaman dengan model belajar menggunakan *game* ini	Sangat Setuju Setuju Netral Tidak Setuju Sangat Tidak Setuju	19 20 5 4 2	38% 40% 10% 8% 4%

Pada Tabel 4.4 dapat dilihat, terdapat hasil jawaban dari para responden terhadap *system usability*. Terdapat 50 responden yang telah memberikan pendapatnya terkait tampilan hingga performa sistem, yang dimana sebelumnya pertanyaan tersebut telah ditetapkan. Selain pada tabel 4.5, terdapat penyebaran data tanggapan responden yang digambarkan ke dalam bentuk grafik. Grafik tersebut dibagi menjadi lima bagian sesuai ketegori pertanyaan survey *usability*.



Gambar 4.20 Grafik Pernyataan Pertama *Learnability*

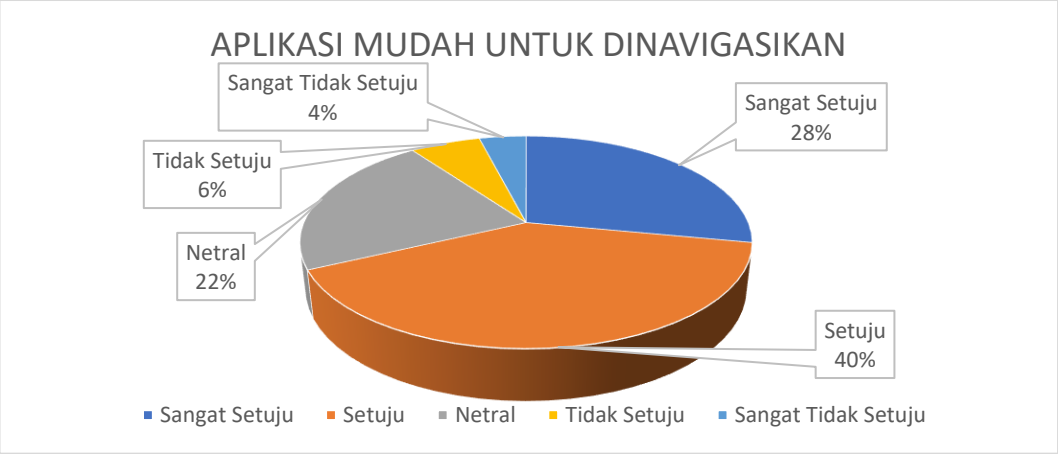
Berdasarkan Gambar 4.20 dapat terlihat terdapat 9 responden dengan persentase 18% berpendapat sangat setuju terhadap penyataan kemudahan pemahaman terhadap aturan permainan, sebanyak 27 responden berpendapat setuju dengan persentase 54%, sebanyak 5 responden masih berpendapat netral dengan persentase 10%, terdapat 7 responden dengan persentase 14% merasa tidak setuju dengan pernyataan ini, dan terdapat 2 responden dengan persentase 4% berpendapat sangat tidak setuju terhadap pernyataan *game* ini mudah dipahami aturan permainannya.



Gambar 4.21 Grafik Pernyataan Kedua *Learnability*

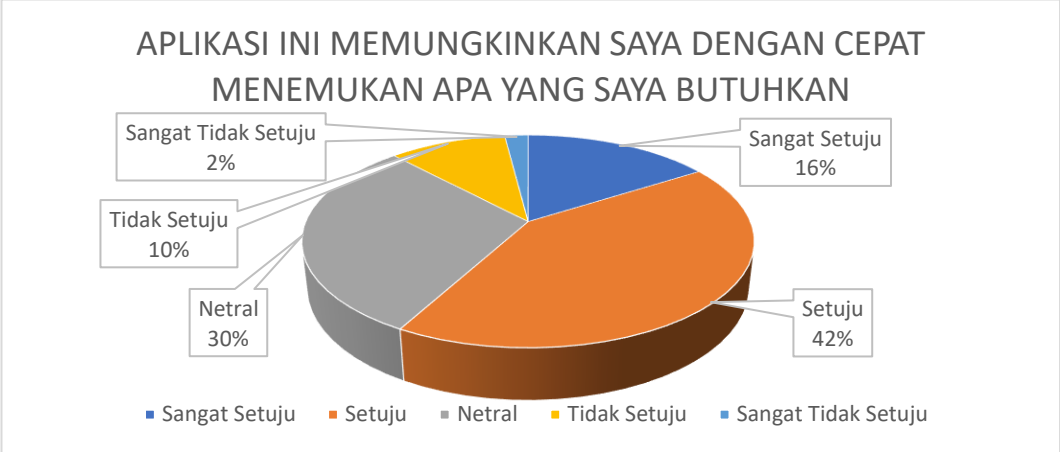
Pada Gambar 4.21 terdapat grafik terkait kemudahan dalam memahami tampilan visual dalam *game*, terdapat 14 dari total responden dengan persentase 28% berpendapat mereka sangat setuju terhadap pernyataan tersebut. Terdapat 20 responden dengan jawaban setuju, 7 responden dengan persentase 14% menjawab netral, 7 responden menjawab tidak setuju dengan persentase 14%, dan sebanyak 2 dari total 50 responden berpendapat sangat tidak setuju dengan peryataan tersebut. pernyataan terkait kemudahan dalam memahami tampilan visual dalam *game*

didominasi dengan tanggapan positif, namun terdapat pula pendapat yang tidak tersebut dengan pendapat tersebut. berdasarkan variasi latar belakang responden hal ini menjadi kemungkinan yang terhitung normal mengingat perbedaan pemahaman serta pandangan terhadap *game* ini.



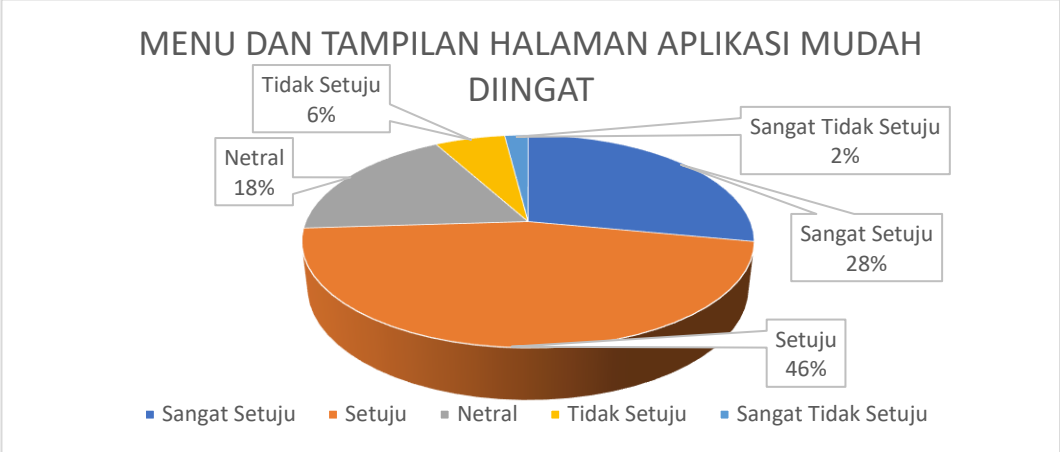
Gambar 4.22 Grafik Pernyataan Pertama *Efficiency*

Pada Gambar 4.22 terkait pernyataan kemudahan navigasi dalam aplikasi, terdapat 14 responden berpendapat sangat setuju dengan pernyataan tersebut dengan persentase 28%. Selain itu terdapat 20 dari total 50 responden berpendapat setuju dengan pernyataan tersebut. sebanyak 11 responden dengan persentase 22% masih berpendapat netral, terdapat 3 responden dengan persentase 6% berpendapat tidak setuju bahwa aplikasi mudah untuk dinavigasikan, dan terdapat 2 responden menjawab sangat tidak setuju terhadap pernyataan ini, dengan persentase 4%.



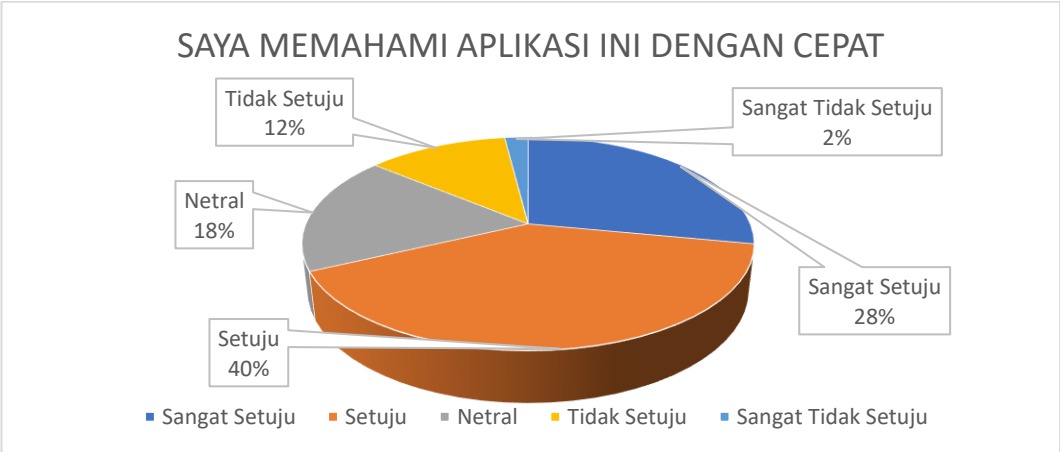
Gambar 4.23 Grafik Pernyataan Kedua *Efficiency*

Pada Gambar 4.23 terlihat grafik terkait pernyataan aplikasi ini memungkinkan saya dengan cepat menemukan apa yang saya butuhkan, terdapat 8 responden berpendapat sangat setuju dengan pernyataan ini dengan persentase 16%. Sebanyak 21 dari total 50 responden berpendapat setuju terhadap pernyataan ini, dengan persentase 42%. Sebanyak 15 responden menjawab netral, dengan persentase 30%. Terdapat 5 responden berpendapat tidak setuju, dengan persentase 10%, dan terdapat 1 responden berpendapat sangat tidak setuju terhadap pernyataan ini, dengan persentase 2%. Dapat dilihat dalam grafik pada gambar 4.21, jawaban didominasi dengan tanggapan positif terhadap pernyataan pada kategori *efficiency*, dapat disimpulkan *efficiency system* berjalan sebagaimana mestinya.



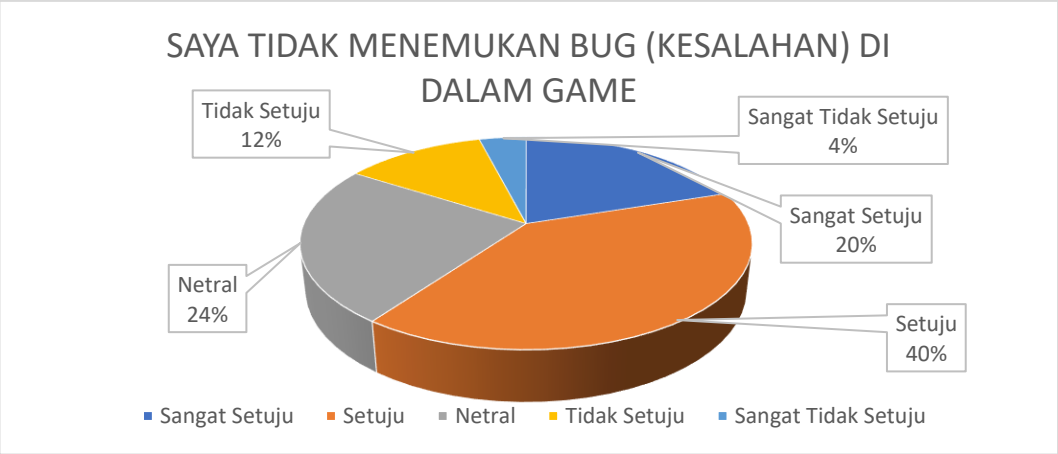
Gambar 4.24 Grafik Pernyataan Pertama *Memorability*

Pada Gambar 4.24 pada grafik terkait pernyataan kemudahan dalam mengingat tampilan menu dan halaman aplikasi, terdapat 14 responden berpendapat sangat setuju, dengan persentase 28%. Sebanyak 23 responden berpendapat setuju, dengan persentase 46%. Terdapat 9 responden dengan persentase 18% memilih menjawab netral. Terdapat 3 responden berpendapat tidak setuju terhadap pernyataan kemudahan dalam mengingat tampilan pada aplikasi, dan terdapat 1 responden menjawab sangat tidak setuju dengan pernyataan tersebut, dengan persentase 2%.



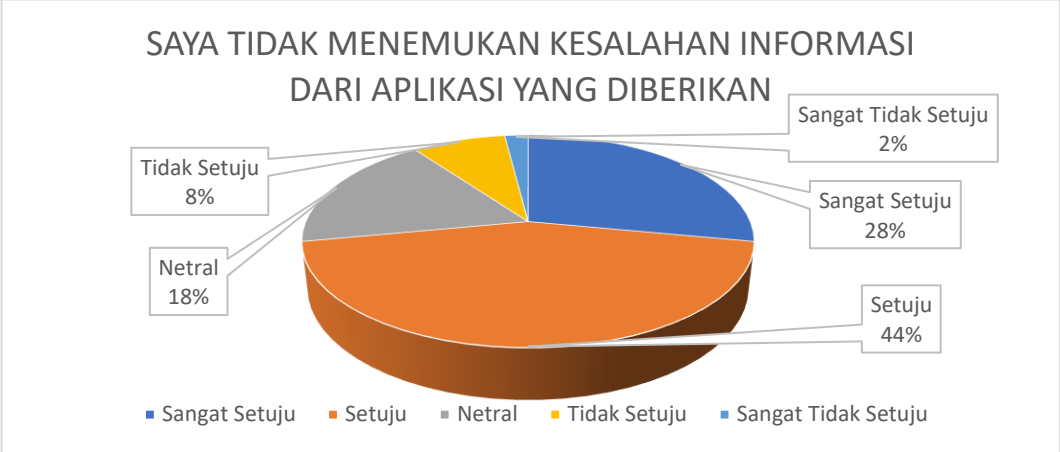
Gambar 4.25 Grafik Pernyataan Kedua *Memorability*

Pada Gambar 4.25 dapat terlihat grafik pernyataan responden dapat memahami aplikasi dengan cepat, sebanyak 14 dari total 50 responden berpendapat sangat setuju dengan pernyataan tersebut. sebanyak 20 responden berpendapat setuju, dan terdapat 9 responden dengan persentase 18% memilih menjawab netral. Terdapat 6 responden berpendapat tidak setuju dengan pernyataan yang diberikan, dengan besar persentase 12%. Terdapat pula 1 responden dengan besar persentase 2%, berpendapat sangat tidak setuju dengan pernyataan tersebut.



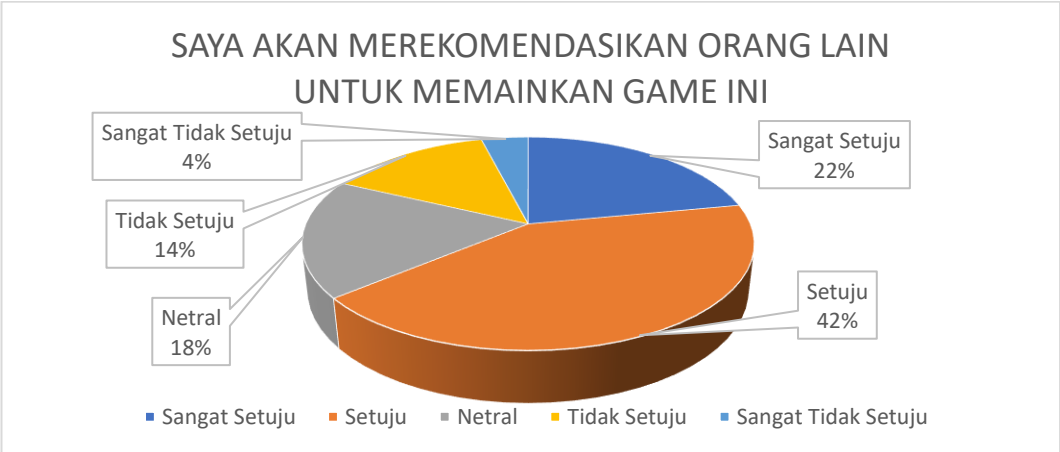
Gambar 4.26 Grafik Pernyataan Pertama *Errors*

Pada Gambar 4.26 terkait pernyataan tidak ditemukannya *bug* di dalam *game*, terdapat 10 responden berpendapat sangat setuju dengan pernyataan tersebut, dengan besar persentase 20%. Sebanyak 20 responden berpendapat setuju, dengan besar persentase 40%, dan sebanyak 12 responden memilih netral terhadap pernyataan ini, dengan besar persentase 24%. Terdapat 6 responden berpendapat tidak setuju dengan pernyataan yang diberikan, dan terdapat 2 responden dengan persentase 4% berpendapat sangat tidak setuju dengan pernyataan ini.



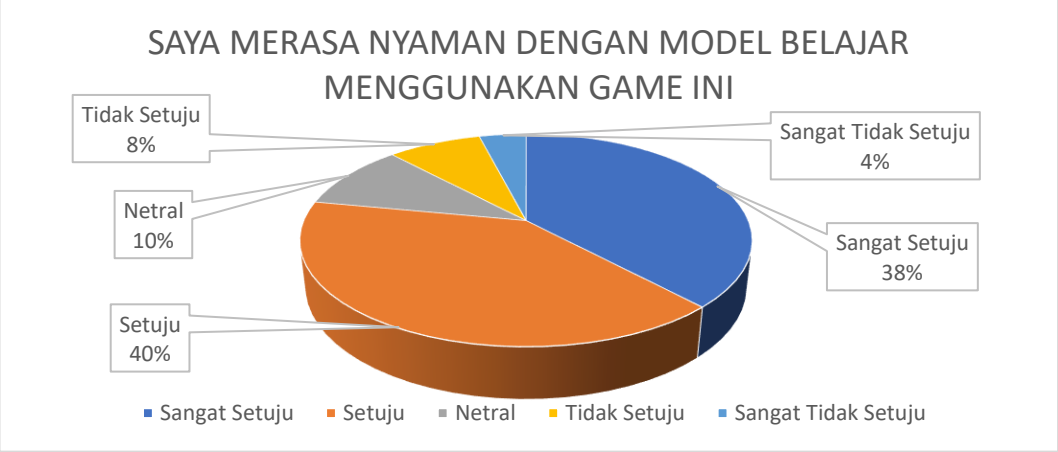
Gambar 4.27 Grafik Pernyataan Kedua *Errors*

Pada Gambar 4.27 terkait pernyataan terkait tidak ditemukannya kesalahan informasi yang terdapat dalam *game*, terdapat 14 responden berpendapat sangat setuju dengan pernyataan ini, dengan persentase 28%. Sebanyak 22 responden berpendapat setuju, dengan persentase 44% dan sebanyak 9 responden memilih menjawab netral pada pernyataan tersebut. terdapat 4 responden berpendapat tidak setuju dengan pernyataan tidak ditemukannya kesalahan informasi yang terdapat dalam *game*, dengan besar persentase 8%, dan terdapat 1 responden menjawab sangat tidak setuju dengan pernyataan tersebut,



Gambar 4.28 Grafik Pernyataan Pertama *Satisfaction*

Pada Gambar 4.28 terkait pernyataan apakah responden akan merekomendasikan *game* ini kepada orang lain, sebanyak 11 responden menjawab sangat setuju, dengan persentase 22%, sebanyak 21 responden dengan persentase 42% berpendapat setuju dalam merekomendasikan orang lain untuk memainkan *game* ini. sebanyak 9 responden memilih menjawab netral, dengan persentase 18%. Terdapat 7 responden merasa tidak setuju dan tidak akan merekomendasikan orang lain untuk memainkan *game* ini, dengan besar persentase 14%. Terakhir terdapat 2 responden yang menjawab sangat tidak setuju terhadap pernyataan ini, dengan besar persentase 4%. Berdasarkan hasil tanggapan para responden sebagian besar jawaban didominasi oleh tanggapan positif, dimana responden merasa *game* ini cukup bagus sehingga mereka berkenan merekomendasikan orang lain untuk memainkan *game* ini.



Gambar 4.29 Grafik Pernyataan Kedua *Satisfaction*

Pada Gambar 4.29 terkait pernyataan perasaan nyaman dengan model belajar dengan menggunakan *game* ini, terdapat 19 dari total 50 responden berpendapat bahwa mereka merasa nyaman dengan model pembelajaran dengan

menggunakan *game* ini, dengan besar persentase nya 38%. Sebanyak 20 responden merasa setuju dengan pernyataan ini, dan sebanyak 5 responden memilih menjawab netral pada pernyataan ini. Terdapat 4 responden dengan besar persentase 8%, merasa tidak setuju dengan pernyataan tersebut, dan terdapat 2 responden berpendapat sangat tidak setuju dengan pernyataan yang diberikan tersebut.

4.7 Integrasi Dengan Agama Islam

Dalam penelitian ini, perancangan *game* "THE MA'HAD" bertujuan untuk memberikan gambaran kepada para mahasiswa baru Universitas Islam Negeri Maulana Malik Ibrahim Malang, untuk lebih memahami sekaligus mengetahui kehidupan serta pembelajaran yang terdapat di dalam ma'had itu sendiri. Selain memberikan gambaran terkait kehidupan ma'had, *game* ini juga memberikan pengetahuan terkait pembelajaran agama islam yang patut diketahui oleh setiap umat muslim dalam bentuk media yang baru yang diharapkan dapat memberikan kesan berbeda dari yang lain. Di dalam islam sendiri telah di jelaskan dalam (*Q.S at-Taubah: 122*), terkait pentingnya menuntut ilmu. Berikut firman Allah SWT. dalam (*Q.S at-Taubah: 122*) :

وَمَا كَانَ ٱلْمُؤْمِنُونَ لِيَنفِرُواْ كَآفَّةً ۚ فَلَوْلَا نَفَرَ مِن كُلِّ فِرْقَةٍ مِّنْهُمْ طَآئِفَةٌ لِّيَتَفَقَّهُواْ فِى ٱلدِّينِ وَلِيُنذِرُواْ قَوْمَهُمْ إِذَا رَجَعُوٓاْ إِلَيْهِمْ لَعَلَّهُمْ يَحْذَرُونَ

"Tidak sepatutnya bagi mukminin itu pergi semuanya (ke medan perang). Mengapa tidak pergi dari tiap-tiap golongan di antara mereka beberapa orang untuk memperdalam pengetahuan mereka tentang agama dan untuk memberi peringatan kepada kaumnya apabila mereka telah kembali kepadanya, supaya mereka itu dapat menjaga dirinya."

Berdasarkan penafsiran dari Ibnu Katsir yang merupakan hafidz, ulama sekaligus pemikir, menafsirkan ayat diatas sebagai berikut. Tatkala kaum Mukminin dicela oleh Allah bila tidak ikut ke medan perang kemudian Nabi ﷺ mengirimkan sariyahnya, akhirnya mereka berangkat ke medan perang semua tanpa ada seorang pun yang tinggal, maka turunlah firman-Nya berikut ini: (Tidak sepatutnya bagi orang-orang yang mukmin itu pergi) ke medan perang (semuanya. Mengapa tidak) (pergi dari tiap-tiap golongan) suatu kabilah (di antara mereka beberapa orang) beberapa golongan saja kemudian sisanya tetap tinggal di tempat (untuk memperdalam pengetahuan mereka) yakni tetap tinggal di tempat (mengenai agama dan untuk memberi peringatan kepada kaumnya apabila mereka telah kembali kepadanya) dari medan perang, yaitu dengan mengajarkan kepada mereka hukum-hukum agama yang telah dipelajarinya (supaya mereka itu dapat menjaga dirinya) dari siksaan Allah, yaitu dengan melaksanakan perintah-Nya dan menjauhi larangan-Nya (*Tafsir Ibnu Katsir 4.2.Pdf*, n.d.). Sehubungan dengan ayat ini Ibnu Abbas r.a. memberikan penakwilannya bahwa ayat ini penerapannya hanya khusus untuk sariyah-sariyah, yakni bilamana pasukan itu dalam bentuk sariyah lantaran Nabi ﷺ tidak ikut. Sedangkan ayat sebelumnya yang juga melarang seseorang tetap tinggal di tempatnya dan tidak ikut berangkat ke medan perang, maka hal ini pengertiannya tertuju kepada bila Nabi ﷺ berangkat ke suatu ghazwah. Selain dalam (*Q.S at-Taubah: 122*), terdapat pula hadits terkait pentingnya belajar dan memberi pengajaran.

إِنَّ أَفْضَلَكُمْ مَنْ تَعَلَّمَ الْقُرْآنَ وَعَلَّمَهُ

"Sesungguhnya orang yang paling utama di antara kalian adalah yang belajar Al-Qur`an dan mengajarkannya."

Al Hafiz Ibnu Katsir dalam kitabnya Fadhail Quran halaman 126-127 berkata: Maksud dari sabda Rasulullah Shalallahu ‘alaihi wasallam “Sebaik-baik kalian adalah orang yang belajar Alquran dan mengajarkan kepada orang lain” adalah, bahwa ini sifat-sifat orang-orang mukmin yang mengikuti dan meneladani para rasul. Mereka telah menyempurnakan diri sendiri dan menyempurnakan orang lain. Hal itu merupakan gabungan antara manfaat yang terbatas untuk diri mereka dan yang menular kepada orang lain.

Berdasarkan penelitian yang dilakukan, terdapat beberapa poin penting yang merupakan bentuk penyebaran ajaran islam serta sarana menambah pengetahuan melalui media baru, khusus nya kepada mahasantri baru yang hendak memasuki ma’had. Melalui *game* diharapkan para mahasantri maupun pengguna yang memainkannya dapat menambah pengetahuan dan memberikan gambaran terkait ma’had, sekaligus memberikan wawasan baru terhadap beberapa materi yang akan dipelajari di ma’had.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil pengujian yang dilakukan terhadap perancangan serta perubahan perilaku NPC dalam *game* "THE MA'HAD" menggunakan metode *behaviour tree* dan berbasis *decision support system* menggunakan metode *simple additive weighting* didapatkan kesimpulan bahwa perilaku NPC dapat diatur sesuai dengan kondisi data *player* yang memainkan *game.* Data yang didapat dari *player* berupa pengalaman, nilai serta waktu dalam menyelesaikan serangkaian tes yang disediakan di dalam *game.* Data ini yang menjadi kriteria dalam penentuan perilaku NPC yang dimana terbagi menjadi tiga *behaviour* yaitu *greet, approach* dan *follow*. Hasil dari pengujian menggunakan 50 data yang berbeda menghasilkan output yang sesuai hipotesis atau berdasarkan perhitungan yang dilakukan secara manual.

Berdasarkan hasil survei yang dilakukan kepada 50 pengguna, didapatkan respon positif dengan persentase lebih dari 80%. Survei dilakukan dengan membagi 10 pertanyaan ke dalam 5 kategori berbeda yaitu, *learnability, Efficiency, memorability, errors, satisfaction*. Berdasarkan hasil survei, pada grafik pertanyaan learnability didapatkan respon dari pengguna yaitu lebih dari 70% responden setuju dengan pernyataan pada kategori learnability. Pada pernyataan kategori efficiency, lebih dari 63% responden setuju dengan pernyataan pada kategori ini. Pada kategori memorability, sebanyak 71% responden berpendapat setuju pada kategori ini. Pada kategori errors, sebanyak 66% responden setuju dengan pernyataan yang terdapat

dalam kategori ini. Pada kategori satisfaction, sebanyak 71% responden setuju dengan pernyataan pada kategori ini.

Berdasarkan hal tersebut, dapat disimpulkan bahwa perancangan perilaku NPC dalam *game* "THE MA'HAD" dapat menggunakan *decision support system* dengan metode *simple additive weighting*. Alternatif yang dihasilkan oleh perhitungan *decision support system* dipadupadankan oleh metode *behaviour tree* untuk menginisiasikan tingkat prioritas dari alternatif yang telah didapat terhadap perilaku NPC yang akan dieksekusi.

5.2. Saran

Perancangan perilaku pada NPC dalam *game* "THE MA'HAD" menggunakan metode *behaviour tree* berbasis *decision support system* masih jauh dari kata sempurna. Masih terdapat banyak kekurangan di dalamnya yang harus ditingkatkan serta diperbaiki agar sistem menjadi lebih baik. Berikut beberapa saran yang dapat penulis sampaikan untuk penelitian kedepannya.

1. Mengembangkan *user interface* dan *user experience* yang terdapat dalam *game* agar lebih menarik dan mudah dipahami oleh pengguna.
2. Memperbanyak variasi karakter serta varisasi perilaku yang terdapat dalam *game* agar menciptakan suasanya yang lebih realistis di dalam *game.*
3. Penerapan *decision support system* dengan model yang berbeda dan mengimplementasikan alternatif yang dihasilkan ke dalam *game.*
4. Memperbanyak kriteria serta alternatif yang dihasilkan yang akan diimplementasikan ke dalam *game.*

DAFTAR PUSTAKA

Alamsyah, F., Diwa, W., & Yunus, A. (2019). Implementasi Algoritma Collision Detection Dan Finite State Machine Untuk Karakter Musuh Pada Game Bertipe Metroidvania. *RAINSTEK : Jurnal Terapan Sains & Teknologi*, *1*(2), 8–13. https://doi.org/10.21067/jtst.v1i2.3062

Anto, A. G., Mustafidah, H., & Suyadi, A. (2019). Sistem Pendukung Keputusan Penilaian Kinerja Karyawan Menggunakan Metode SAW (Simple Additive Weighting) di Universitas Muhammadiyah Purwokerto (Decision. *Juita*, *3*(1), 193–200.

Arif, Y. M., Harini, S., Nugroho, S. M. S., & Hariadi, M. (2021). An Automatic Scenario Control in Serious Game to Visualize Tourism Destinations Recommendation. *IEEE Access*, *9*, 89941–89957. https://doi.org/10.1109/ACCESS.2021.3091425

Chen, Y. (2021). Application of analytic hierarchy process (AHP) and simple additive weighting (SAW) methods in mapping flood-prone areas. *Frontiers in Artificial Intelligence and Applications*, *341*(Mcdm), 435–441. https://doi.org/10.3233/FAIA210274

Colledanchise, M., & Ögren, P. (2017). *Behavior Trees in Robotics and AI: An Introduction*. https://doi.org/10.1201/9780429489105

Eka Sugiyarti, Kamarul Azmi Jasmi, Bushrah Basiron, & Miftachul Huda. (2018). Decision Support System of Scholarship Grantee Selection using Data Mining, International Journal of Pure and Applied Mathematics. *International Journal of Pure and Applied Mathematics*, *119*(15), 2239–2249. https://www.academia.edu/download/57052927/DECISION_SUPPORT_SYSTEM_OF_SCHOLARSHIP_GRANTEE.pdf

Fadila, J. N., Nugroho, F., Artanti, V., & ... (2023). Penerapan HFSM Pada Game 3D "Petualang Qur'an." *Jurnal Ilmiah …*. https://ejournal.upbatam.ac.id/index.php/jif/article/view/6538%0Ahttps://ejournal.upbatam.ac.id/index.php/jif/article/download/6538/3095

Faisal, M., Nurhayati, H., Arif, Y. M., Kurniawan, F., & Nugroho, F. (2016). Immersive bicycle game for health virtual tour of uin maulana malik ibrahim Malang. *Jurnal Teknologi*, *78*(5), 325–328. https://doi.org/10.11113/jt.v78.8330

Fauzan, R., Indrasary, Y., & Muthia, N. (2018). Sistem Pendukung Keputusan Penerimaan Beasiswa Bidik Misi di POLIBAN dengan Metode SAW Berbasis Web. *Jurnal Online Informatika*, *2*(2), 79. https://doi.org/10.15575/join.v2i2.101

Fink, A., Denzinger, J., & Aycock, J. (2007). Extracting NPC behavior from computer games using computer vision and machine learning techniques. *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and*

Games, CIG 2007, *Cig*, 24–31. https://doi.org/10.1109/CIG.2007.368075

Hogg, E., Hauert, S., Harvey, D., & Richards, A. (2020). Evolving behaviour trees for supervisory control of robot swarms. *Artificial Life and Robotics*, *25*(4), 569–577. https://doi.org/10.1007/s10015-020-00650-2

Ibrahim, R. (2020). Implementasi Fisher Yates Shuffle dan Non Player Character pada Permainan Tembak Setan. *Walisongo Journal of Information Technology*, *2*(2), 137. https://doi.org/10.21580/wjit.2020.2.2.7048

Junaidi, A., Yunus, A., Wiguna, A. S., Sukun, B., & Jawa, M. (2021). *Implementasi Behavior Tree Pada Perilaku NPC di Game Sidescroller*. *4*, 92–103.

Mcquillan, K. (2015). *A Survey of Behaviour Trees and their Applications for Game AI*.

Nurhayati, H., Faisal, M., & Romadhoni, M. (2017). Q - Puzzle game for memorizing the recitation of Qur'an surah. *Journal of Theoretical and Applied Information Technology*, *95*(20), 5637–5641.

Pratiwi, H., Pahrudin, P., & Nassa, M. Y. A. T. (2021). Memangun Game 3D Side Scroll Dan Menerapkan Model Behavior Treepada Npc Enemy Dalam Game "Maverick." *Jurnal Informatika Wicida*, *10*(2), 50–61. https://doi.org/10.46984/inf-wcd.1825

Rizky, & Pudrianisa, S. L. G. (2019). Pengujian Usability Pada Tangible Game Sebagai Media Promosi Candi. *Journal INFOS*, *2*(1), 13–19.

Sekhavat, Y. A. (2017). Behavior Trees for Computer Games. *International Journal on Artificial Intelligence Tools*, *26*(2). https://doi.org/10.1142/S0218213017300010

Senthil Kumar, K., & Malathi, D. (2018). Context free grammar identification from positive samples. *International Journal of Engineering and Technology(UAE)*, *7*(3.12 Special Issue 12), 1096–1097. https://doi.org/10.14419/ijet.v7i3.11983

Šmíd, A. (2017). Comparison of Unity and Unreal Engine. *Czech Technical University in Prague Faculty*, *May*, 77.

Stenros, J. (2017). The Game Definition Game: A Review. *Games and Culture*, *12*(6), 499–520. https://doi.org/10.1177/1555412016655679

Sugianto, B., & Utama, G. P. (2021). Implementasi Algoritma Pathfinding Dan Decision Tree Dalam Pembuatan Video Game Bergenre Third Person Shooter. *Skanika*, *4*(2), 7–14. https://doi.org/10.36080/skanika.v4i2.1825

Suryadi, A. (2018). Perancangan Aplikasi Game Edukasi Menggunakan Model Waterfall. *Jurnal Petik*, *3*(1), 8. https://doi.org/10.31980/jpetik.v3i1.352

Tafsir Ibnu Katsir 4.2.pdf. (n.d.).

Taherdoost, H., & Madanchian, M. (2023). Multi-Criteria Decision Making (MCDM) Methods and Concepts. *Encyclopedia*, *3*(1), 77–87. https://doi.org/10.3390/encyclopedia3010006

Trianto, E. M., Junaedi, H., & ... (2017). Agen Cerdas Berbasis Controller Fuzzy Pada Permainan Strategi Pertempuran Dengan Behavior Tree. *Seminar Nasional Ilmu ...*, 1–8. https://ojs.widyakartika.ac.id/index.php/sniter/article/view/34%0Ahttps://ojs.widyakartika.ac.id/index.php/sniter/article/download/34/29

Warpefelt, H. (2016). *The Non-Player Character: Exploring the believability of NPC presentation and behavior* (Issue 16).

Wulandari, A. D. (2015). Game Edukatif Sejarah Komputer Menggunakan Role Playing Game (RPG) MAKER XP Sebagai Media Pembelajaran di SMP Negeri 2 Kalibawang. *Universitas Negeri Yogyakarta*, *Unknown*(Unknown), 1–18. http://www.dt.co.kr/contents.html?article_no=2012071302010531749001