PROPOSAL TUGAS AKHIR
GAME 2D PLATFORMING "'Ctrl'+It" MENGGUNAKAN BEHAVIOR TREE BERBASIS UNREAL ENGINE 4

Nama : David Brave Navy Putra
NRP : 218116715
Jurusan/Prodi/Major : Teknik Informatika / S1 /*Computational Intelligence*
Dosen Pembimbing : Hendrawan Armanto, S.Kom, M.Kom
Co-Pembimbing : -

I. Latar Belakang

Perkembangan teknologi informatika atau disiplin ilmu komputer berperan sangat penting bagi kehidupan manusia. Dengan adanya perkembangan teknologi terutama di bidang informatika, manusia dapat dengan mudah mendapatkan, memberi juga menyimpan informasi dan melakukan kegiatan sehari-hari dengan bantuan teknologi yang ada. Teknologi membawa manusia melihat lebih jauh ke dunia luar, membuka wawasan berpikir, serta membangun kreativitas untuk menciptakan hal-hal dan inovasi baru. Karena teknologi dapat melakukan banyak hal yang berada di luar kemampuan manusia seperti menyelesaikan perhitungan matematis berjuta-juta dalam satuan waktu detik dan juga tidak akan terjadi *human error*. Teknologi yang berkembang sendiri dapat berbentuk benda nyata (*hardware*) maupun berbentuk ide, atau sistem (*software*).

Salah satu bentuk perkembangan teknologi informatika yang sedang *booming* pada masa ini adalah game. Game adalah media hiburan yang dapat digunakan oleh setiap kalangan yang untuk menghilangkan rasa jenuh atau bosan. Game juga memiliki manfaat lain seperti pengembangan otak, melatih pemecahan masalah, meningkatkan konsentrasi, melatih kecepatan koordinasi mata dengan tangan, dan lainnya. Terdapat banyak sekali *genre* pada game, beberapa di antaranya adalah *action, shooter, horror, strategy, role playing, sport, vehicle simulations, platforming, construction management, adventure, artificial life* dan *puzzle games*.

Pada awal tahun 2000-an, industri game terutama permainan digital mengalami perkembangan yang sangat pesat. Hal ini ditunjukkan dengan banyaknya perangkat keras untuk bermain game yang terus bermunculan dan terus berkembang, seperti *mobile phone*, tablet, PC (komputer) , laptop, *console* (Playstation, XBOX, dan Nintendo Wii), dan lain-lain. Oleh karena itu, game ini akan dibuat dengan Unreal Engine 4 atau yang lebih dikenal dengan Unreal Engine yang merupakan salah satu *game engine* paling populer dalam pembuatan game dengan target utama pengguna PC atau laptop.

II. Tujuan

Pada subbab ini akan menjelaskan mengenai tujuan-tujuan yang akan dihasilkan dalam pembuatan tugas akhir ini. Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Membuat game 2D menggunakan Unreal Engine 4.
2. Merancang game 2D dengan konsep *platformer* untuk desktop atau laptop berbasis Unreal Engine, yang menarik dan dapat menghibur para pemainnya.
3. Mengaplikasian AI menggunakan *Behavior Tree* agar NPC terasa lebih hidup.

III. Teori Penunjang

Terdapat juga beberapa teori yang menunjang pembuatan aplikasi game ini, seperti:

A. Object Oriented Programming (OOP)

Pemrograman berorientasi objek merupakan metode untuk membuat program menggunakan *class* dan *object*. OOP membuat pengembangan dan pemeliharaan menjadi lebih mudah. OOP menyediakan beberapa konsep seperti *inheritance, data binding, polymorphism, encapsulation*, dan lain-lain.

B. C++

Bahasa pemrograman C++ akan digunakan sebagai dasar untuk mempelajari coding yang digunakan oleh Unreal, Hal ini dikarenakan *coding* Unreal Engine menggunakan bahasa C++ sebagai bahasa *native* atau bahasa dasar dari pemrograman yang telah disediakan.

C. Artificial Intelligence (AI)

Kecerdasan buatan atau *artificial intelligence* (AI) merupakan salah satu bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan atau berpikir seperti dan sebaik manusia. Pada game, AI bertindak dalam mengendalikan karakter komputer agar dapat bermain dengan pemain manusia maupun melawan pemain manusia. Penggunaan AI pada game *platformer* adalah sebagai penentu pergerakan apa yang akan dilakukan NPC atau musuh.

D. AI Behaviour Tree

Agar pergerakan AI terlihat lebih natural dan masuk akal, dibutuhkan metode untuk merancang perilaku musuh atau NPC. Metode yang akan digunakan adalah *Behaviour Tree* (BT). Selain untuk sistem kontrol, BT adalah model yang umum digunakan untuk merancang perilaku AI pada sebuah game yang memiliki kelebihan berupa kompleksitas AI yang dapat dibuat menghasilkan AI yang terlihat lebih natural dibandingkan metode lainnya. Penentuan BT model sesuai dengan jenis perilaku yang sudah ditentukan, dengan kondisi-kondisi tertentu yang dapat memperlancar alur permainan dan pergerakan AI.

E. Genre Game

Game ‘Ctrl’+It merupakan sebuah game dengan genre platformer. Platformer adalah genre game yang mirip dengan side scroll, perbedaan platformer biasanya dibumbuhi dengan berbagai puzzle dan bukan hanya sekedar bermain saja, seperti side scroll yang lebih casual gaya permainaya (acarde) platformer lebih membutuhkan sedikit pemahaman dari side scroll,

platformer biasanya juga digabungkan dengan sidescroll, seperti game super Mario, sonic, dan game sejenisnya, ataupun hanya pada satu layar saja tanpa adanya transisi seperti game donkey kong dan space panic.

F. Photo Editor

Mengingat game yang dibuat adalah 2D game, maka akan diperlukan aplikasi yang dapat membuat, dan mengedit file *sprite* 2D yang akan digunakan di dalam game. Sehingga dibutuhkan sebuah aplikasi *photo editor*, untuk menghasilkan sprite dan *background terrain* yang dapat dipakai. Photo editor yang akan dipakai adalah Adobe Photoshop CC 2018. Photoshop adalah perangkat lunak pengedit citra buatan Adobe Systems yang dikhususkan untuk pengeditan foto/gambar dan pembuatan efek. Photoshop melakukan penyuntingan berbasis pixel, sehingga lebih cocok digunakan dalam pembuatan *sprite*. Beberapa langkah pemerosesan sprite yang akan dilakukan adalah sebagai berikut:

a. Pixelation

Pixelation merupakan proses pengurangan pixel dari sebuah gambar menjadi gambar dengan ukuran dimensi resolusi pixel lebih kecil. Proses ini biasanya dilakukan untuk mengubah sebuah gambar object menjadi pixel art atau sprite yang dapat digunakan dalam proses animasi.

b. Depixelation

Depixelation merupakan proses kebalikan dari pixilation. Di mana gambar dengan ukuran pixel terlalu kecil, akan ditambah jumlah pixelnya agar dimensi resolusi pixel menjadi lebih besar. Proses ini dilakukan mengingat tidak semua asset yang digunakan memiliki ukuran dimesi resolusi yang sama, sehingga jika terdapat pixel art yang terlalu kecil, resolusi pixel art tersebut dapat ditingkatkan.

c. Lineart Edit

Secara umum, pixel art memiliki dua jenis style tampilan, yaitu dengan lineart atau tanpa lineart. Lineart ini berfungsi untuk menegaskan perbedaan sprite satu dengan sprite lainnya. Lineart biasanya akan membuat sprite terlihat lebih artificial atau tidak natural. Pada game ini, asset-asset utama tidak menggunakan lineart agar memberi kesan alami, sehingga pada asset yang memiliki lineart dan dirasa tidak sesuai dengan art style game, lineart akan dihilangkan menggunakan photo editor.

d. Color Correction

Pixel art memiliki ciri khas berupa bentuk dari gambar yang terlihat secara utuh batas-batas warnanya. Ini juga yang menjadi alasan penyebutan ‘pixel’ ‘art’. Terkadang beberapa warna pada sprite pixel art dapat terlihat terlalu mencolok, entah karena warna yang terlalu gelap, terlalu terang maupun kontras warna yang tidak sesuai dengan tema asset lainnya. Karenanya, akan dilakukan color correction untuk asset yang terlalu mencolok tersbut. Color

correction dapat berupa menurunan atau penambahan intensitas warna maupun perubahan warna secara total.

**Teori penunjang dapat berubah atau bertambah dalam tahap pengembangan program untuk memenuhi kebutuhan dan perkembangan teknologi terkini yang ada.*

IV. Teori Pengembangan Game

Berikut merupakan metode, tools, algoritma maupun konsep utama yang akan digunakan dalam pembuatan dan pengembangan game ini:

4.1 Unreal Engine 4

Unreal Engine merupakan sebuah Game engine yaitu software yang ditujukan untuk penciptaan dan pengembangan sebuah video game. Sekarang telah banyak beredar segala macam jenis game engine dan diantara sekian banyak game engine yang beredar ada diantaranya yang sangat terkenal, yaitu Unreal Engine dari Epic Games dan Unity dari Unity Technology.

Unreal Engine dikembangkan oleh Epic Games, pada tahun 1998 untuk permainan first-person shooter. Meskipun pada awalnya dikembangkan untuk jenis first person, tetapi kemudian Unreal Engine telah berhasil digunakan dalam berbagai genre lain, termasuk stealth, MMORPG, dan RPG lainnya. Dengan kode yang ditulis dalam C + +, Unreal Engine memiliki fitur tingkat tinggi dan merupakan alat yang digunakan oleh banyak pengembang game saat ini.

Inilah yang menjadi alasan utama digunakannya Unreal Engine. Karena Unreal Engine menyediakan banyak fitur yang berguna dalam pengembangan game, entah secara program, tampilan, interface maupun mengenai AI. Seperti behavior tree yang sudah didukung secara langsung dan penuh oleh Unreal Engine 4. Mulai dari komponen, action atau task juga dokumentasi mengenai penggunaan, fungsi dan pengaplikasian behavior tree pada AI. Sehingga Unreal Engine 4 dipilih menjadi game engine yang digunakan dalam proses pembuatan game ini. Penjelsan lebih lengkap mengenai fitur Unreal Engine 4 yang digunakan akan dijabarkan secara lebih mendetail setelah pembahasan rilis Unreal Engine 4.

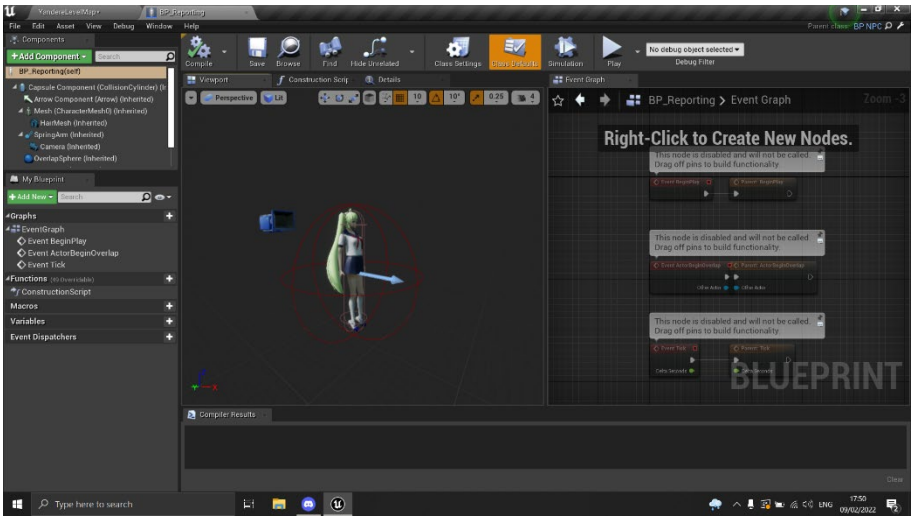
Unreal Engine memiliki beberapa versi rilis yang terkenal dan sering digunakan, seperti Unreal Engine 3 atau yang lebih dikenal dengan Unreal Development Kit, Unreal Engine 4 yang merupakan versi Unreal Engine yang paling populer dan juga merupakan game engine yang akan digunakan pada pembuatan game ini serta engine terbaru yaitu Unreal Engine 5, yang mana baru mencapai tahap early access, namun sudah dapat digunakan sejak Mei 2021.

Unreal Engine 4 menggunakan Blueprint (Visual Scripting) dan dapat disatupadukan dengan Text Script berbahasa C++. Pada Unreal Engine 4 sendiri banyak kelebihan yang dapat diimplementasikan pada

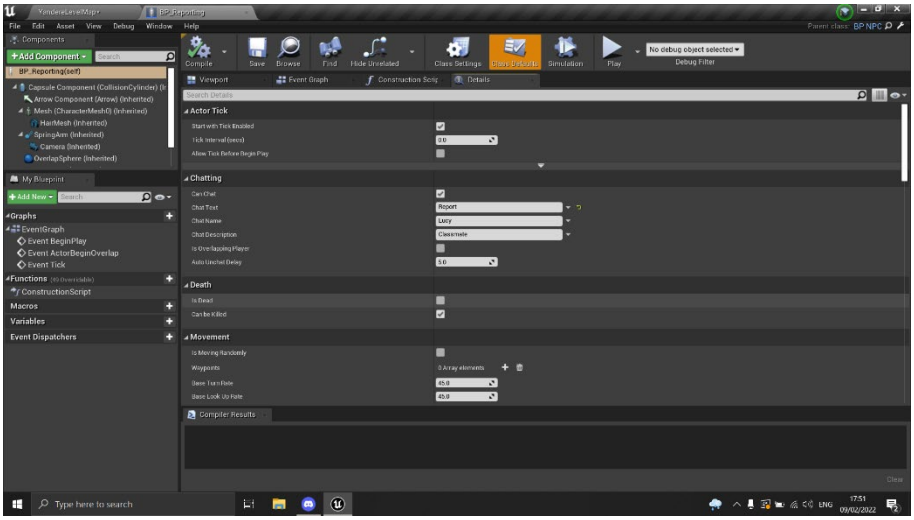
game ini, berikut adalah penjabaran fitur-fitur Unreal Engine 4 yang akan digunakan pada game ‘Ctrl’+It :

a. Unreal Blueprint Visual Scripting

Blueprint Visual Scripting adalah sistem scripting yang menggunakan konsep node-interface untuk menghasilkan sebuah element pada gameplay dari Unreal editor secara langsung. Bahasa scripting ini juga bersifat Object Oriented Classes atau yang lebih dikenal dengan sebutan Object. Object inilah yang dapat diimplementasikan sebagai blueprint pada penggunanya nanti.



Gambar 4.1
Interface Blueprint Editor
(Sumber gambar : Dokumen pribadi)

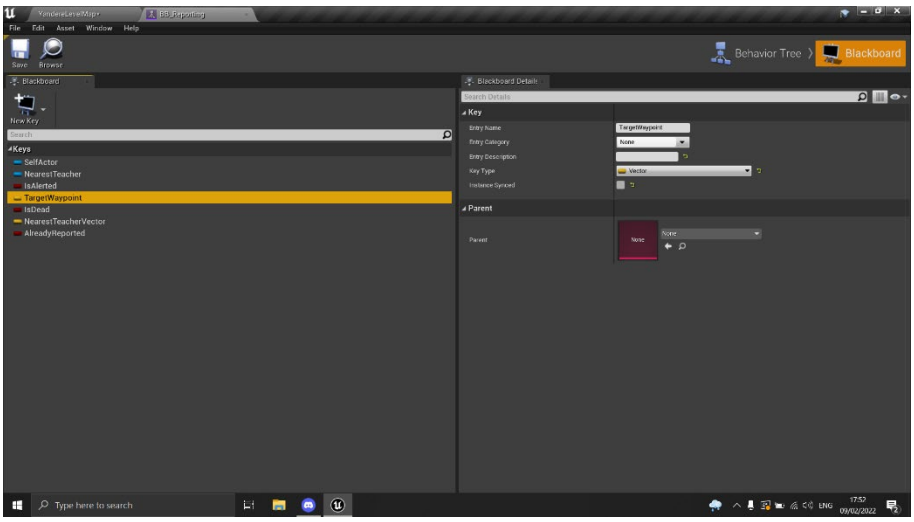


Gambar 4.2
Interface Blueprint Property Editor
(Sumber gambar : Dokumen pribadi)

Blueprint ini merupakan salah satu fitur khas yang hanya dimiliki Unreal Engine. Blueprint memiliki banyak properti juga asset yang didukung dan didokumentasikan dengan baik, sehingga pemrograman dengan Blueprint-lah yang akan menjadi metode utama pembuatan game ‘Ctrl’+It.

b. Unreal Black Board

Blackboard adalah tempat penyimpanan data di mana nilai dari data tersebut dapat dibaca dan diubah untuk membentuk decision making pada AI. Sebuah blackbord dapat dipakai oleh sebuah actor atau AI pawn maupun digunakan untuk beberapa actor sekaligus dalam sebuah grup. Tujuan utama blackboard adalah membentuk central data yang bisa dipakai dengan mudah, di mana sebuah blackboard dapat digunakan bersama behavior tree untuk membentuk AI pada Unreal Engine 4.



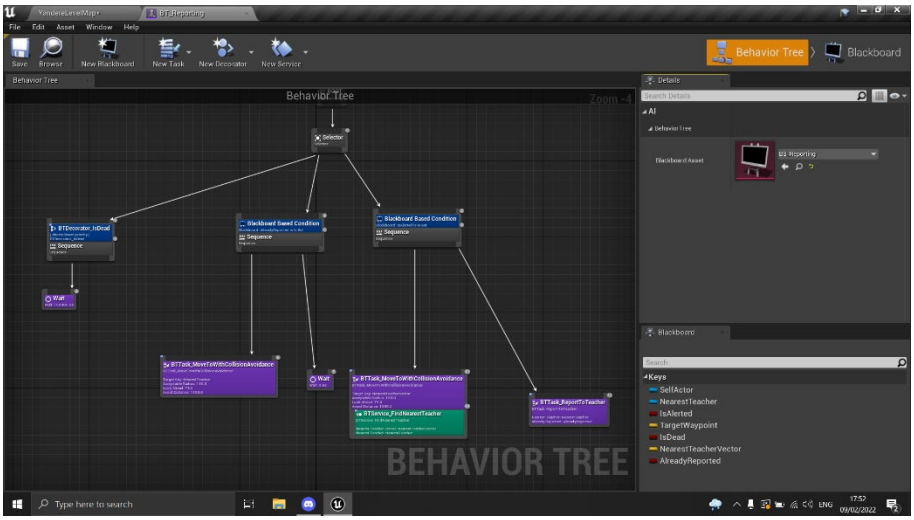
Gambar 4.3
Interface Unreal Blackboard
(Sumber gambar : Dokumen pribadi)

Pada blackboard, dapat ditambahkan variable data. Data pada blackboard bisa berupa tipe data konventional seperti Boolean maupun tipe data khusus seperti class object, vector dan lain sebagainya. Data tersebut juga dapat berupa reference kepada object lain. Variable pada blackboard dapat diakses secara langsung oleh behavior tree yang mana value dari variable tersebut dapat digunakan untuk menentukan action apa yang akan dilakukan AI.

c. Unreal Behavior Tree

Unreal Engine 4 memberikan fitur dan dukungan penuh terhadap pembuatan AI menggunakan behavior tree. Sebuah actor pada

Unreal Engine 4 dapat memiliki sebuah behavior tree. Behavior tree ini akan menjadi otak dari actor tersebut. Seluruh action yang akan dilakukan actor akan ditentukan sesuai task dari behavior tree. Unreal Engine 4 memberikan kemudahan dengan sequence blackboard based condition, di mana sebuah task atau action akan dilakukan berdasarkan value pada sebuah variable di dalam blackboard yang terhubung dengan behavior tree.

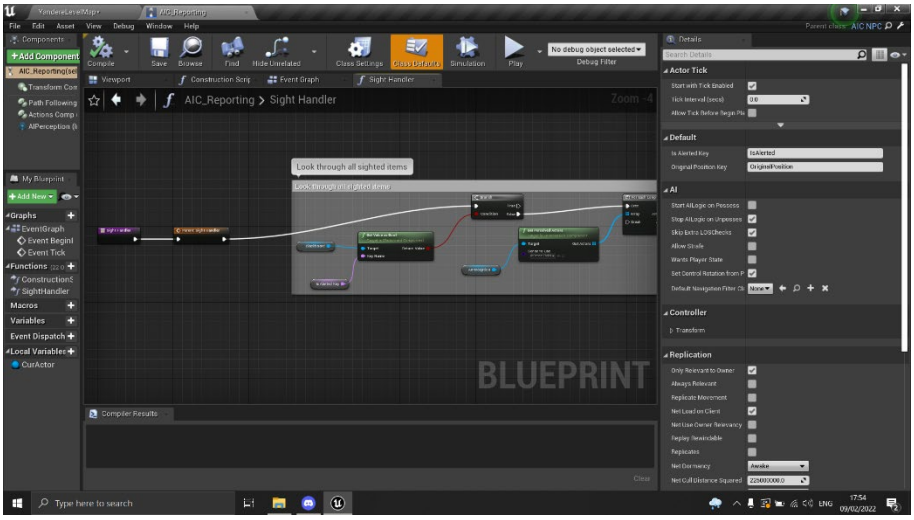


Gambar 4.4
Contoh Unreal Behavior Tree
(Sumber gambar : Dokumen pribadi)

Didalam Behavior Tree Editor, pengguna dapat mengatur script Artificial Intelligence melalui sistem berdasarkan visual node (serupa dengan Blueprint) untuk Actors pada level yang akan dibuat. Setiap node kemudian dapat diatur dan disesuaikan pada details tab di samping kanan layar.

d. Unreal Controller

Controller merupakan sebuah actor yang tidak memiliki badan fisik yang dapat diakses secara langsung (mesh, texture dan lain sebagainya). Controller dapat mengendalikan pawn atau actor yang dikontrol (possessed). Controller secara umum dapat dibedakan menjadi dua, player controller dan AI controller. Payer controller merupakan controller yang menggerakan actor sesuai dengan input user, sedangkan AI controller akan menngerakan actor sesuai dengan AI decision making yang ada. Beberapa contoh AI decision making adalah state machine dan behavior tree.

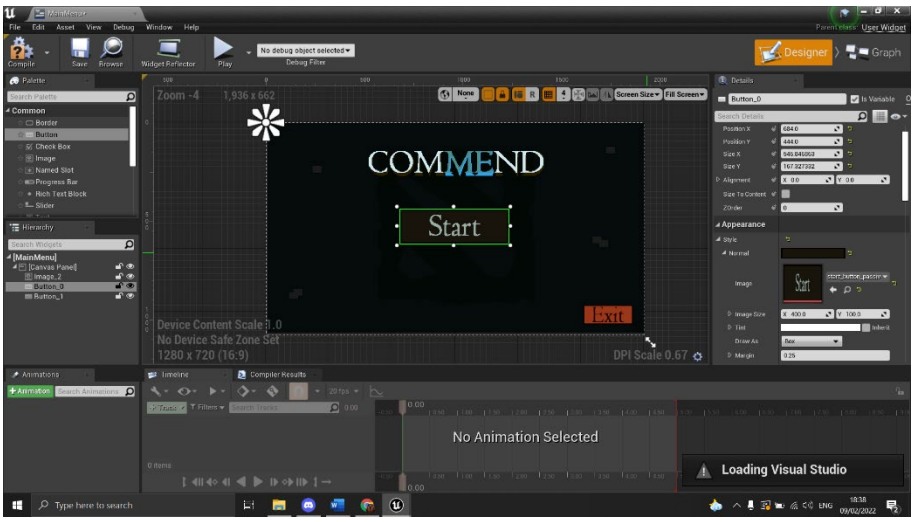


Gambar 4.5
Contoh Unreal Controller
(Sumber gambar : Dokumen pribadi)

Controller memiliki dua fungsi utama, yaitu possessed function di mana controller akan mengendalikan actor pawn, dan unpossessed function, di mana pengendalian akan pawn dilepaskan. Secara umum, terdapat hubungan one-to-one anatara controller dan actor pawn, namun beberapa pawn dapat memiliki controller yang sama tergantung penggunaannya.

e. Unreal UI Designer

UI Desginer merupakan visual tool yang berguna untuk membuat UI element seperti HUD, menu dan element-element lain yang berhubungan dengan graphical interface. Pada intinya, UI Designer adalah widget yang merupakan fungsi-fungsi buatan yang bertujuan untuk membentuk element interface seperti button, slider, progress bar, checkbox dan lain sebagainya. Widget-widget ini dibentuk sedemiakan rupa dalam sebuah blueprint widget khusus yang dapat diakses dari dua tab untuk proses penambahannya. Tab-tab tersebut adalah designer tab dan graph tab. Designer tab berfungsi untuk mengedit tampilan visual dan layout dari interface beserta fungsi-fungsi dasar pada widget. Sedangkan graph tab memberikan lebih banyak keleluasaaan dalam fungsi yang dapat ditambahkan pada widget tersebut.

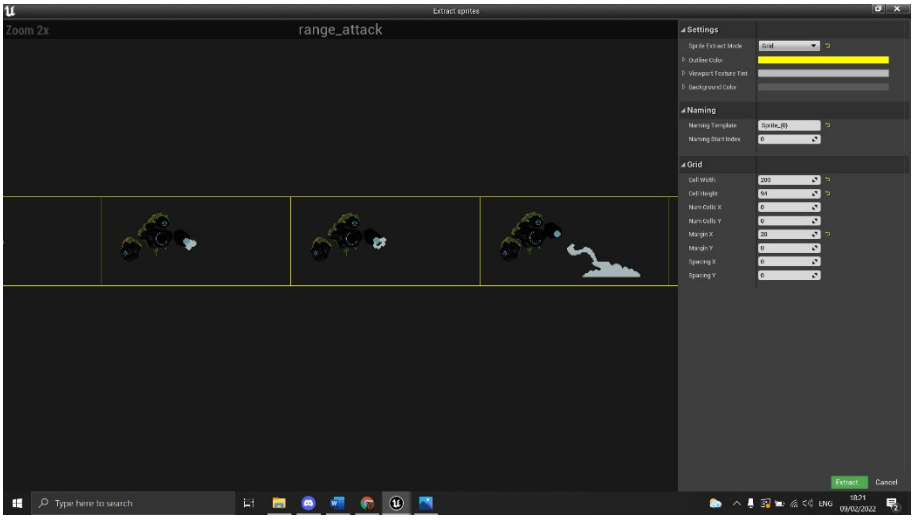


Gambar 4.6
Interface Unreal UI Designer
(Sumber gambar : Dokumen pribadi)

Pada pengaplikasiannya, UI designer akan digunakan untuk membuat start menu screen, HUD dan GUI di dalam game, pause menu screen, option menu screen, confirmation pop up, game over screen, dan game ending screen. Mengingat bahwa game ini berupa single story line yang tidak memiliki cabang atau alternate ending, maka game ending screen yang ada akan dibuat berjumlah satu ending screen.

f. Unreal Paper 2D Sprite

Sebuah sprite dalam Paper 2D adalah sebuah teksture yang dipasangkan ke sebuah mesh bidang datar. Material tersebut dapat direnderer sepenuhnya dalam Unreal Engine 4. Secara sederhana, Paper 2D adalah cara Unreal Engine 4 menggambar 2D images pada layar.



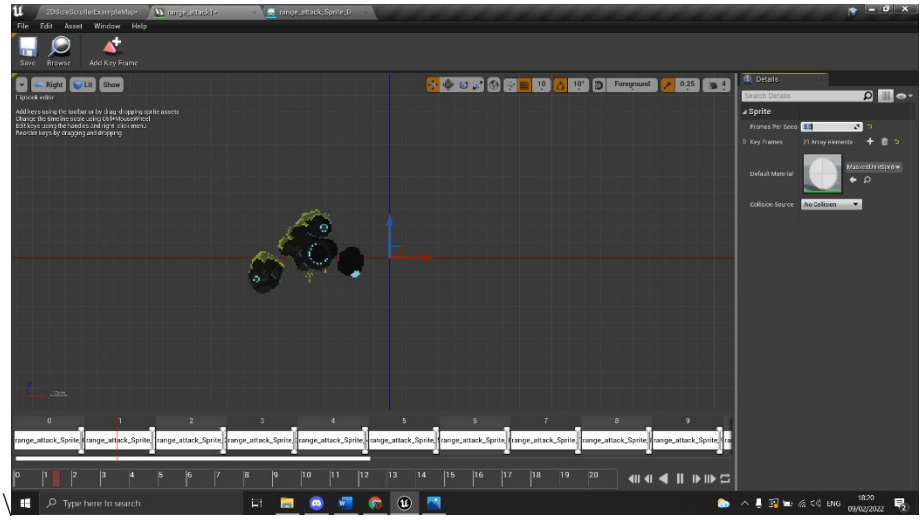
Gambar 4.7
Contoh Paper 2D Sprite Extraction
(Sumber gambar : Dokumen pribadi)

Gambar 4.8
Interface Paper 2D Sprite Editor
(Sumber gambar : Dokumen pribadi)

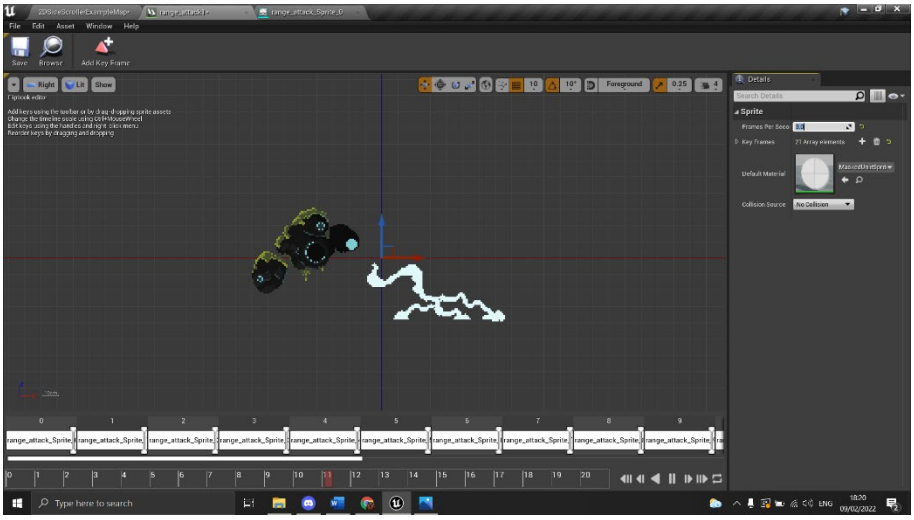
Sprite juga dapat diedit secara langsung pada Unreal Engine 4 dengan sprite editor. Di mana sprite editor sendiri memiliki empat mode utama, yaitu view, edit source region, edit collision display dan edit render geometry. View bertugas memberikan preview sprite yang akan ditampilkan, edit source digunakan untuk menampilkan source teksture sprite secara penuh, edit collision berfungsi untuk mengedit bentuk dan collision pada sprite kemudian render geometry berfungsi untuk menampilkan proses rendering geometry pada sprite.

g. Unreal Paper 2D Flipbook

Beberapa tampilan sprite yang telah dibuat, dapat digabungkan menjadi sebuah animasi yang padu, animasi ini dikenal dengan flipbook. Paper 2D Flipbook secara sederhana merupakan animasi yang dilakukan dengan proses menggambar sprite yang telah disatukan secara berurutan pada layar. Pada Unreal Engine 4, flipbook berisi beberapa keyframe di mana setiap frame memiliki sebuah sprite yang akan ditampilkan dalam selang waktu tertentu. Hal ini lebih dikenal dengan istilah Frame Per Second (FPS). FPS akan menentukan berapa banyak frame yang muncul dalam satu detik dari perhitungan jumlah frame yang kita miliki di sebuah flipbook.



Gambar 4.9
Interface Paper 2D Flipbook 1
(Sumber gambar : Dokumen pribadi)



Gambar 4.10
Interface Paper 2D Flipbook 2
(Sumber gambar : Dokumen pribadi)

Pada bagian ini, kita juga dapat mengatur collsion dari flipbook berdasarkan collsion sprite yang kita miliki. Sehingga sebuah flipbook dapat berubah collisionnya saat dilakukan animasi. Tidak hanya itu, kita juga dapat menambahkan object saat animasi berjalan. Proses ini biasanya dilakukan untuk membuat proyektil pada object atau actor yang memiliki kemampuan menyerang dengan senjata range.

4.2 Behavior Tree

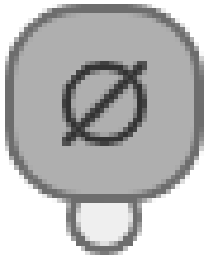
Behaviour tree adalah model matematis yang digunakan untuk mengeksekusi dan menyusun peralihan antara berbagai task yang berbeda pada autonomous agent. Behaviour tree adalah sebuah model untuk pengeksekusian rencana yang secara grafis dipresentasikan sebagai sebuah pohon. Behaviour tree memungkinkan untuk mengontrol karakter pada games dan menjelaskan hirarki keputusan dan aksi. Sehingga dapat dimengerti bahwa Behaviour tree adalah model yang memilki struktur seperti pohon yang dapat digunakan untuk mengontrol behaviour NPC.

BT muncul sebagai tool untuk pemodelan artificial intelligence (AI) pada games sejak digunakannya pada game Halo 2. BT sering digunakan sebagai decision making pada NPC. BT juga popular digunakan sebagai metode yang efektif untuk mengontrol robot. Biasanya, sebuah game engine secara berulang mengeksekusi sebuah game loop, dimana didalamnya terdapat sub-engine untuk physics, artificial intelligence, grafis, dan sebagainya.

Setiap node memiliki aksi dan kembalian (return) tersendiri kepada parent nya, diantaranya:

- Success: node selesai melakukan tugasnya

- Failure: node gagal menyelesaikan tugasnya
- Continue/running: node belum menyelesaikan tugasnya



Gambar 4.11
Penggambaran Root Node
(Sumber gambar : Dokumen pribadi)

Root pada BT adalah node tanpa parent. Sedangkan, node yang tidak memiliki child adalah leaf (daun). Behaviour tree memiliki berbagai macam komponen, tetapi dasar yang harus dan pasti dimiliki oleh BT adalah sebagai berikut:

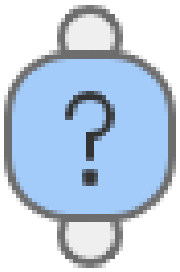
a. Composites

Composite Node adalah node yang bisa memiliki lebih dari satu anak. Node ini akan memproses child(children) baik secara terurut, maupun secara acak, tergantung kondisi yang sedang terjadi saat itu, dan saat prosesnya selesai, node ini akan mengembalikan nilai success ke parent node-nya jika prosesnya berhasil, dan dapat juga mengembalikan failure ke parent node-nya jika prosenya gagal. Selama node ini memproses child(children), node ini akan mengembalikan nilai running ke parent node-nya. Beberapa contoh composite node adalah:

- **Sequence Node**
  Sequences node mengambarkan rangkaian dari suatu task yang perlu dilakukan. Node ini akan mengunjungi semua child(children), satu per satu untuk melakukan operasi-operasi yang ada pada child(children) tersebut. Operasi-operasi yang ada pada child(children) tersebut harus dijalankan seluruhnya. Jika anak terakhir berhasil untuk dijalankan, maka simpul tersebut akan mengembalikan nilai success ke pada orangtuanya. Secara sederhana, node ini dapat diumpamakan sebagai AND operator dalam programming.

Gambar 4.12
Penggambaran Sequence Node
(Sumber gambar : Dokumen pribadi)

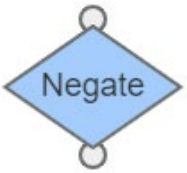
- **Selector Node**
  Selector node berfungsi untuk memilih atau menjalankan salah satu proses pada child dari node tersebut. Selector merupakan kebalikan dari sequences. Jika pada sequence akan menghasilkan failure jika salah satu child menghasilkan failure, selector akan menghasilkan success jika minimal satu dari child menghasilkan success. Program akan mengecek dari child pertama. Jika child pertama mengembalikan nilai failure, maka akan lanjut ke child kedua. Jika child kedua menghasilkan success, maka node ini akan mengirimkan success ke parent node-nya. Sehingga dapat dianggap bahwa selector node menyerupai OR operator dalam programming.



Gambar 4.13
Penggambaran Selector Node
(Sumber gambar : Dokumen pribadi)

b. Decorator
Decorator node ini juga dapat memiliki child, hanya saja child yang dimiliki oleh simpul ini terbatas, yaitu hanya satu. Fungsinya bermacam-macam, bisa untuk mengubah status yang diterima dari child node, bisa untuk menghentikan proses dari child node, ataupun mengulang proses yang terjadi di child node. Itu semua bergantung pada jenis dari decoratornya. Jenis decorator yang digunakan pada

umumya adalah negate, yang berfungsi untuk mengubah hasil (result) yang diberikan oleh child dari node tersebut. Jika child berstatus fail, maka akan mengirimkan success ke parent dari node tersebut, begitu juga sebaliknya, jika child berstatus success, maka node tersebut akan mengirimkan status failure ke parent node-nya.



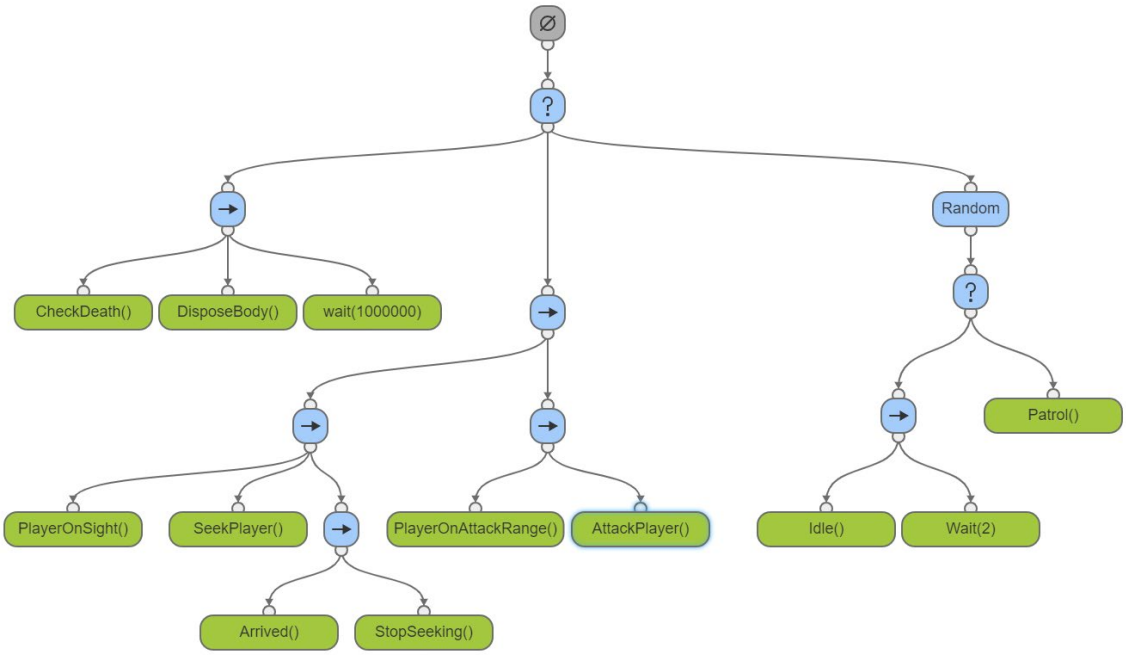
Gambar 4.14
Penggambaran Negator Node
(Sumber gambar : Dokumen pribadi)

c. Leaf

Leaf node adalah node yang tidak memiliki child. Akan tetapi, node ini adalah node yang paling berperan pada behavior tree, karena node jenis ini yang akan menentukan dan menetapkan jalannya game. Contohnya adalah leaf yang digunakan untuk berjalan atau moving. Node inilah yang akan benar-benar menjalankan aksinya untuk berjalan menuju poin tertentu pada peta yang telah didefinisikan sebelumnya. node ini dapat memiliki prosedur yang memiliki parameter, misalnya leaf untuk berjalan akan memiliki parameter berupa targetnya.

Gambar 4.15
Penggambaran Leaf Node
(Sumber gambar : Dokumen pribadi)

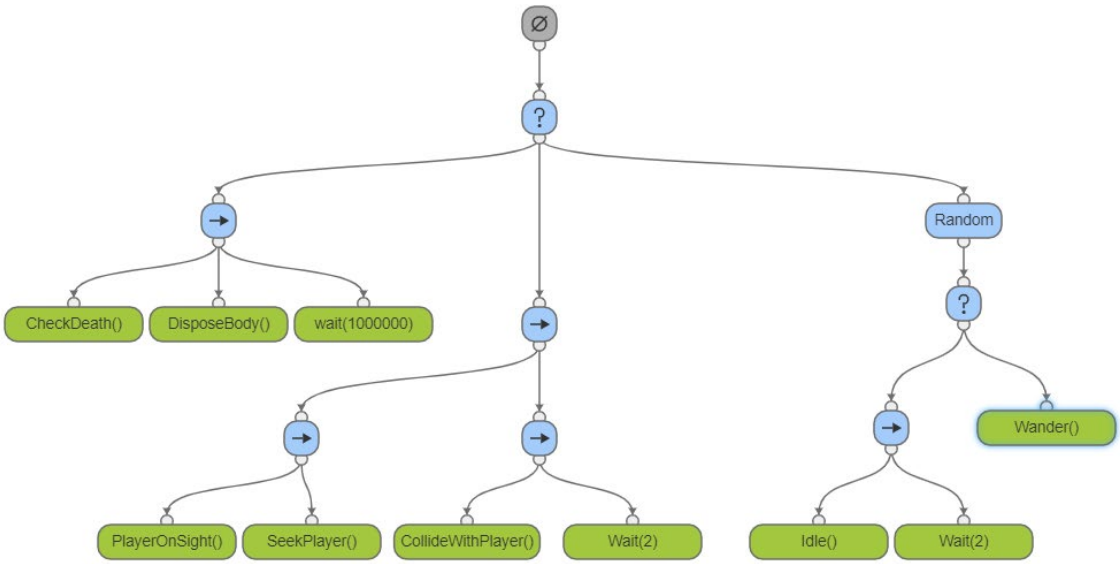
AI pada game ‘Ctrl’+It ini akan memiliki bahvior tree yang berbeda tergantung pada tipe NPC dari AI. Karena terdapat dua tipe NPC, secara umum, akan terdapat dua behavior tree utama pada pengaplikasiannya. Berikut adalah penggambaran kedua behavior tree tersebut secara umum.



Gambar 4.16
Base Design Behavior Tree Aggressive NPC
(Sumber gambar : Dokumen pribadi)

Aggressive NPC dapat menyerang player jika NPC tersebut melihat player pada line of sight-nya. Sehingga akan diberi pengecekan apakah NPC melihat player, jika NPC melihat player, maka akan dilakukan action seek player. Lalu saat AI mencapai player, maka proses seek player akan dihentikan. Setelah itu NPC akan mencoba menyerang player, jika player berada dalam jarak serangnya.

Saat NPC tidak melihat player, maka NPC akan melakukan patrol pada path tertentu di dalam map. Terdapat random yang akan menentukan apakah NPC akan berjalan ke titik patroli berikutnya, ataukah NPC akan diam di tempat. Jika NPC berhasil diserang oleh player, maka body NPC akan dihilangkan (dispose) dan behavior tree akan dibuat menunggu dalam waktu yang lama agar tidak melakukan pengecekan kembali.



Gambar 4.17
Base Design Behavior Tree Friendly NPC
(Sumber gambar : Dokumen pribadi)

Friendly NPC dapat berupa hewan (kucing, atau anjing) yang akan mengikuti player. Sehingga akan diberi pengecekan apakah NPC melihat player, jika NPC melihat player, maka akan dilakukan action seek player. Lalu saat AI menyentuh player, maka NPC tersebut akan menunggu sesaat sebelum melaukan pengecekan action lainnya. Pada saat ini, NPC dapat menggunakan animasi idle, maupun animasi wait.

Jika tidak ditemukan player di sekitar NPC, maka NPC akan melakukan wander atau idle pada map. Penentuan kedua hal tersebut akan dirandom. Sehingga NPC dapat melakukan wander, lalu idle, berdasarkan hasil random. Saat idle, NPC akan menunggu sejenak sebelum melakukan random untuk pengecekan kondisi tersebut lagi. Jika NPC mati, maka body NPC akan dihilangkan (dispose) dan behavior tree akan dibuat menunggu dalam waktu yang lama agar tidak melakukan pengecekan kembali.

4.3 AI Algorithm

AI dari game ini akan melakukan task atau action sesuai dengan behavior tree yang dibuat. Beberapa algortima movement AI yang akan digunakan adalah seek dan wander. Di bawah ini, akan dijelaskan sedikit psudocode dari proses kedua moving algoritma tersebut.

Algoritma 4.1 Data Structure For Kinematic Steering Output

```
01:      CLASS KinematicSteeringOutput()
02:         Let Velocity
03:         Let Rotation
```

```
04:       END CLASS
```

Sebelum menuju ke algortima, terdapat sebuah struktur data yang biasa digunakan untuk actor pada Unreal Engine 4, yaitu KinematicSteeringOutput. Struktur data KinematicSteeringOutput ini digunakan untuk menyimpan nilai variable kecepatan dan sudut rotasi dari object yang akan digunakan. Kecepatan (velocity) biasanya berbentuk vector, 2D atau 3D tergantung bidang object yang akan digunakan. Kemudian nilai rotasi (rotation) dapat berupa angka sudut, maupun nilai radians rotasi object yang akan digunakan.

Algoritma 4.2 Algoritma AI Movement Wander

```
01:       CLASS Wander()
02:          LET character ← NPC.Character
03:          LET max_speed ← NPC.MaxSpeed
04:          LET max_rotation ← NPC.MaxRotation
05:          FUNCTION GetSteering()
06:             LET steering ←
                new KinematicSteeringOutput()
07:             LET steering.Velocity ← max_speed *
                character.Orientation.ToVector()
08:             LET steering.Rotation ← Random() *
                max_rotation
09:             RETURN steering
10:          END FUNCTION
11:       END CLASS
```

Algoritma wander digunakan untuk membuat AI seolah berjalan berkeliling map. Algoritma ini dapat disimpan ke dalam sebuah class, di mana class ini meiliki tiga variable data utama, dan satu fungsi steering. Ketiga variable data tersebut adalah character, yang merupakan character yang akan melakukan wander, max_speed, yang merupakan variable limit dari kecepatan character tersebut, kemudian ada max_rotation yang merupakan limit dari derajat perputaran character tersebut. Variable max_speed dan max_rotation dibuat agar AI terlihat lebih natural dalam bergerak.

Fungsi steering pada class wander akan menginisialisasi sebuah variable steering baru yang akan diisi oleh struktur data KinematicSteeringOutput. Di mana velocity dari steering tersebut merupakan hasil perkalian dari kecepatan maksimal karakter dengan konversi vector dari orientasi karakter tersebut. Untuk membuat karakter tersebut berpindah arah hadap, maka rotasi dari steering akan diisi oleh nilai

random dikalikan rotasi maksimal dari karakter. Hal ini akan mebuat nilai dari kecepatan dan sudut putaran streering berubah, lalu hasil steering baru tersebut akan di return sebagai nilai steering saat ini.

Algoritma 4.3 Algoritma AI Movement Seek

```
01:      CLASS Seek()
02:         LET character ← NPC.Character
03:         LET target ← player.Character
04:         LET max_speed ← NPC.MaxSpeed
05:         FUNCTION GetSteering()
06:            LET steering ←
               new KinematicSteeringOutput()
07:            LET steering.Veloctiy ←
               target.Position – character.Position
08:            LET steering.Velocity ←
               steering.Velocity.Normalize()
09:            LET steering.Velocity ←
               steering.Velocity * max_speed
10:            LET steering.Rotation ← 0
11:            RETURN steering
12:         END FUNCTION
13:      END CLASS
```

Algoritma seek digunakan untuk membuat AI mengikuti player saat player dapat dilihat oleh NPC. Algoritma ini dapat disimpan ke dalam sebuah class, di mana class ini meiliki tiga variable data utama, dan satu fungsi steering. Variable pertama adalah variable character, yang merupakan karakter dari NPC tersebut, variable kedua adalah variable target, di mana pada konteks ini, target yang akan diikuti adalah player, sehingga variable target diisi menjadi karakter dari player. Variable terakhir adalah max_speed yang berupa kecepatan maksimal dari karakter NPC.

Fungsi steering pada class wander akan menginisialisasi sebuah variable steering baru yang akan diisi oleh struktur data KinematicSteeringOutput . Di mana velocity dari steering tersebut merupakan hasil pengurangan dari posisi target atau posisi player dengan posisi karakter atau NPC. Lalu hasil pengurangan tersebut akan dinormalisasi, yang kemudian dikalikan oleh kecepatan maksimal karakter. Lalu steering rotation direset menjadi nol dan didaptkan hasil steering yang baru. Steering ini akan dikembalikan dan dijadikan nilai steering saat ini.

V. Ruang Lingkup

Ruang lingkup yang akan dibahas pada proposal tugas akhir ini adalah hal-hal sebagai berikut:

5.1 Story Game

Storyline dari game ini akan mengambil konsep tentang adanya kehidupan pada dimensi lain. Apa jadinya jika hidup seorang pejuang yang ingin menyelamatkan desanya dari sebuah monster ganas. Monster tersebut sudah meneror seluruh orang-orang yang dia sayangi selama berbulan-bulan. Namun pejuang tersebut tidak memiliki kemampuan dan kekuatan yang cukup untuk melawan monster tersebut. Sang pejuang tersebut akhirnya berusaha mencari bantuan untuk mengalahkan monster tersebut.

Namun bagaimana jika bantuan tersebut datang bukan dari orang di sekitarnya? Bukan guru bela diri, atau kesatria perkasa. Namun orang yang berada 1 dimensi di atas pejuang tersebut. Pemain game inilah orang yang bisa membantu pejuang tersebut. Pemain yang ada pada 3 dimensi, diminta membantu pejuang yang hidup di 2 dimensi dengan menggunakan cara berpikir dan *object* yang ada pada dimensinya sendiri.

Object tersebut bisa berupa *key* pada keyboard player yang ditaruh untuk membantu pejuang tersebut berjalan, maupun *input* mouse yang membantu pejuang tersebut mengalahkan musuh yang ada. Ataupun membantu pejuang tersebut mengakali musuh yang ada.

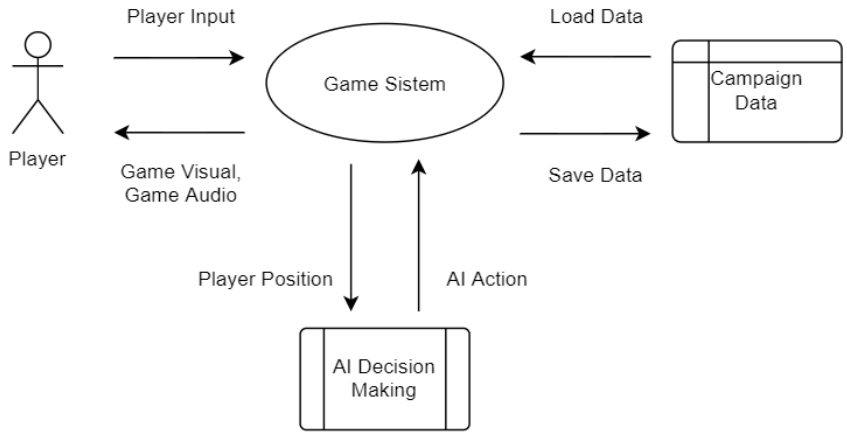
Pada level pertama, akan digambarkan sebuah desa tempat asal pemain. Di desa ini akan terdapat beberapa NPC friendly berupa villager dan juga hewan-hewan peliharaan para villager tersebut. Background map akan lebih dibuat casual dengan object seperti rumah para penduduk desa, pepohonan, lampu-lampu dan object-object pedesaan lainnya. Pada level ini player akan berjalan menelusuri desa sembari diberi arahan tentang movement dari game. Setelah level pertama selesai, maka akan muncul cerita bahwa sebuah monster batu menyerang desa tersebut. Player yang ingin menghentikan serangan monster tersebut berusaha kembali ke desanya, namun terlambat. Monster tersebut telah menghancurkan desanya. Bertekad membalas dendam, player mengejar monster tersebut ke tempat asalnya.

Di level kedua, akan digambarkan sebuah kastil tua markas dari musuh yang menyerang desa player. Pada kastil tersebut akan banyak terdapat dinding batu dari hasil bangunan kastil. Object lain yang terdapat pada level kedua adalah lampu chandelier, pilar batu, obor, lelumutan dan object lain yang mengambarkan kastil tua yang hampir runtuh. Pada level ini, player akan menemui musuh-musuh yang berbeda. Sehingga player harus berusaha melewati map ini serta melawan monster yang menjaga kastil itu. Setelah berhasil melewai seluruh rintangan, player menyadari bahwa monster utama yang menyerang desanya tidak ada di kastil tersebut. Ternyata monster itu berjalan menuju ke sebuah gunung di belakang kastil itu. Merasa tidak gentar, player mengejar monster itu hingga akhirnya dia menemukan collosal golem yang menyerang desanya.

Level terakhir merupakan boss battle melawan collosal golem. Dengan latar belakang pegunungan. Di mana object pada level ini berupa

bebatuan, pepohonan dan awan yang memberikan kesan tempat yang tinggi/ Player dapat menggunakan key untuk menghindari atau menghalangi serangan golem. Golem memiliki hitpoint yang lebih banyak dibandingkan monster lain, sehingga akan dibutuhkan beberapa serangan player untuk mengalahkan golem tersebut. Setelah mengalahkan golem tersebut, player merasa berhasil membalaskan dendam penduduk desanya. Sehingga player berjalan kembali ke desanya. Untuk membangun desa yang diserang golem tersebut.

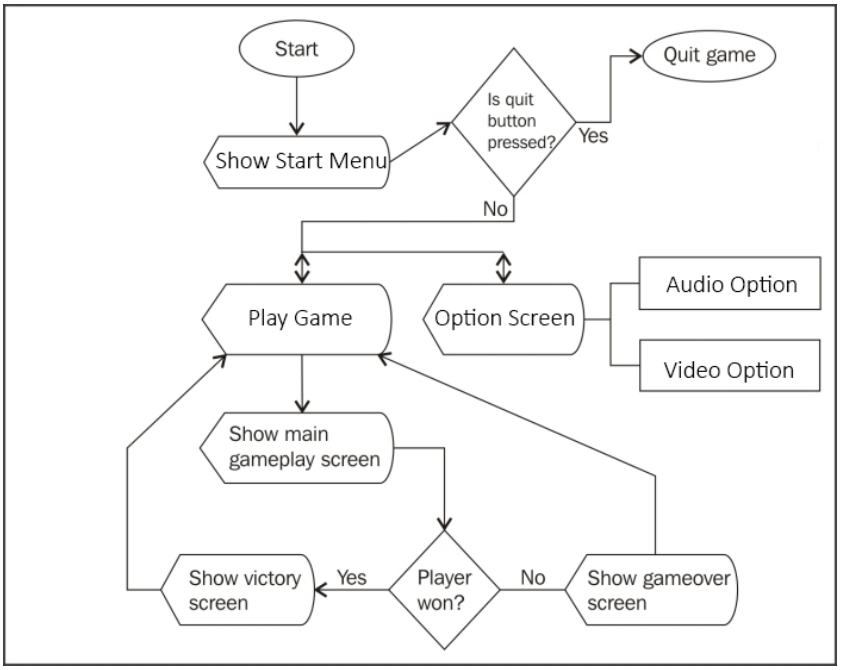
5.2 Arsitektur Game



Gambar 5.1
Arsitektur Game
(Sumber gambar : Dokumen pribadi)

Pada gambar 5.1, player dapat memberikan input kepada sistem melalui keyboard dan mouse. Input tersebut akan diproses oleh game dan akan ada predefined proses berupa decision making dari AI. AI menerima posisi player dan menggunakan data itu untuk menentukan action apa yang akan dilakukan. Seluruh proses itu akan ditampilakan kepada player sebagai output dengan bentuk game visual dan game audio. Seluruh data tersebut dapat disimpan dengan save data pada internal storage dan dapat di load Kembali saat ingin dimainkan lain waktu.

Proses internal game sistem dapat digambarkan sebagai berikut :



Gambar 5.2
Penggambaran Game Sistem
(Sumber gambar : Dokumen pribadi)

Pada saat membuka aplikasi, akan ditampilkan start menu di mana user dapat memilih untuk bermain, mengatur setting atau keluar dari game. Jika user memilih untuk keluar, maka game akan ditutup. Option menu akan membawa user untuk mengatur setting audio dan video pada game. Pada play game, user akan diarahkan ke gameplay screen. Setelah game selesai, akan dilakukan pengecekan apakah user memenangkan campaign atau tidak. Jika user kalah, akan ditampilkan layer game over, jika user menang akan ditampilkan layer kemenangan.

a. Deskripsi Game

Game ini adalah sebuah game *adventure platformer* yang terinspirasi dari game klasik Super Mario Bros. Sehingga visual dan gameplay yang akan dibuat menjadi game 2D agar memberikan kesan dan perasaan yang sama saat bermain Mario Bros. Namun pada game ini, pemain bisa secara langsung membantu karakter utama untuk menyelesaikan misi dan level yang tersedia. Bantuan tidak sekadar cara berjalan namun juga cara mengalihkan atau mengalahkan musuh. Cakupan bantuan yang diberi pemain jauh lebih besar dan banyak dibandingkan game Super Mario Bros, mengingat tingkat kesulitan *puzzle*, level maupun AI dibuat lebih berkembang dan kompleks. Game ini diharapkan bisa memberi rasa senang yang pernah dikenang pemain game pendahulunya, namun dengan *gameplay* dan tampilan yang lebih menarik menggunakan metode juga *tools* yang lebih baik dan modern.

b. Visual

Elemen visual dari game ini berupa 2D *picture* dan 2D *sprite*. *Sprite* tersebut akan dibuat oleh penulis dengan model *pixel art*. Mengingat keterbatasan kemampuan seni penulis, beberapa *asset* akan dilakukan *outsourcing*, kepada pihak lain yang memiliki kemampuan lebih dalam hal ini atau akan diambil dari web *resource* seperti Unreal Engine Marketplace (UE Marketplace) dengan pemberian *credit* sesuai ketentuan jika *asset* digunakan.

c. Non-Player Character (NPC)

NPC pada game ini akan diberi AI yang akan memiliki *Behaviour Tree* untuk membuat NPC dapat terasa lebih hidup. Setiap *Behaviour Tree* bisa memiliki jumlah dan tipe AI *action* atau *task* yang berbeda-beda. NPC yang ada dapat dibagi menjadi dua, yaitu *neutral*, dan *aggressive*.

Untuk NPC yang *neutral*, maka *Behaviour Tree* berisi *action wandering, idle* dan *flee*, dengan *player action* berupa *interact*. Sedangkan untuk NPC *aggressive*, *Behaviour Tree* dapat berupa *patrol, idle, wander, seek, arrive* dan *attack*. *Action player* dapat dibuat menyerang untuk NPC yang bertipe *aggressive*.

Walau tipe NPC sama, *Behaviour Tree* bisa saja berbeda. Seperti NPC musuh yang menyerang dengan pedang harus berjalan menuju pemain terlebih dahulu, baru menyerang saat sampai di depan pemain. Sedangkan musuh yang memiliki serangan proyektil, bisa tidak perlu menuju pemain namun dapat langsung menembak.

d. Skenario

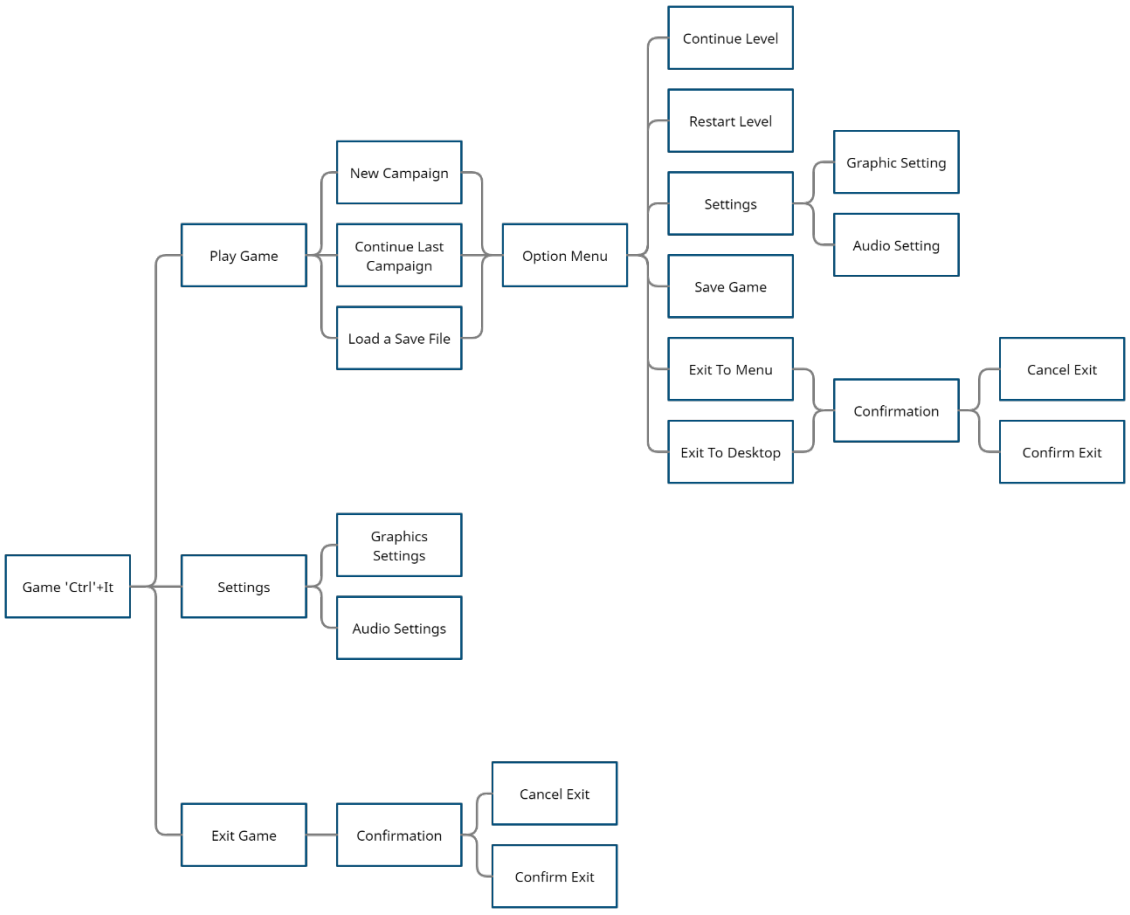
Setelah cerita ditampilkan, maka muncul layar permainan. Pemain akan masuk ke level pertama untuk mengenali mekanik game. Mulai dari *key* yang digunakan, juga fungsi dari *key* yang ada. Pada saat inilah *gameplay* game akan dimulai. Sehingga pemain dapat menggunakan mekanik yang telah disampaikan pada tutorial untuk menyelesaikan *scenario game*. Skenario tidak memiliki ending bercabang. Hanya terdapat dua kondisi game ini selesai, yaitu player kalah dan gagal menyelesaikan story, di mana player akan di respwan pada checkpoint terdekat, atau kemungkinan selanjutnya player berhasil menyelesaikan story dengan mengalahkan bos terakhir.

5.3 Block Diagram

Dalam aplikasi game yang akan dibuat, user pada saat membuka aplikasi akan tersedia pilihan menu mengenai *game, settings* dan *exit*. Jika user memilih game, maka user akan diberi pilihan untuk membuat *campaign* baru, melanjutkan *campaign* terakhir jika ada file dan melakukan *load save* file. Pada pilihan *setting*, user dapat mengatur *interface setting*. *Interface*

terdiri atas 2 hal yaitu *graphical* dan *audio*. User dapat mengatur kedua hal tersebut sesuai kebutuhan. Jika user memilih untuk *exit game*, akan dimunculkan pesan konfirmasi untuk memastikan user benar-benar ingin keluar dan bukan merupakan *misclick* atau kesalahan *input*.

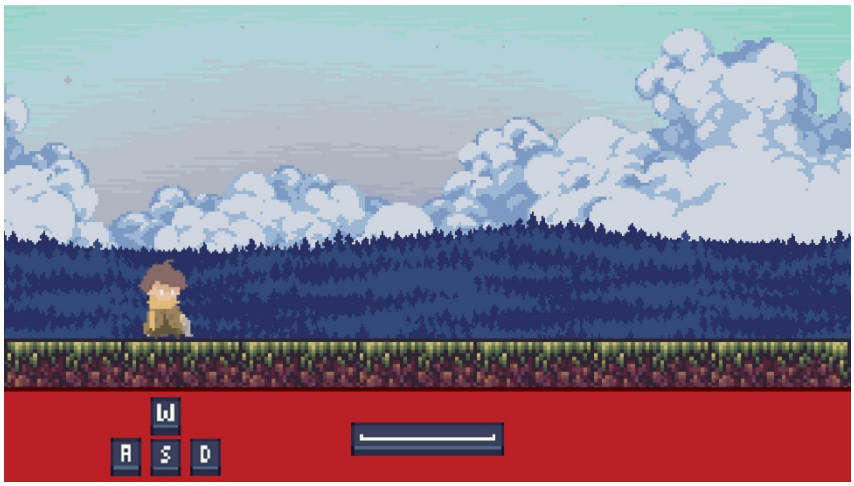
Berikut adalah blok diagram dari aplikasi game yang akan dibuat :



Gambar 5.3
Block Diagram Game Menu
(Sumber gambar : Dokumen pribadi)

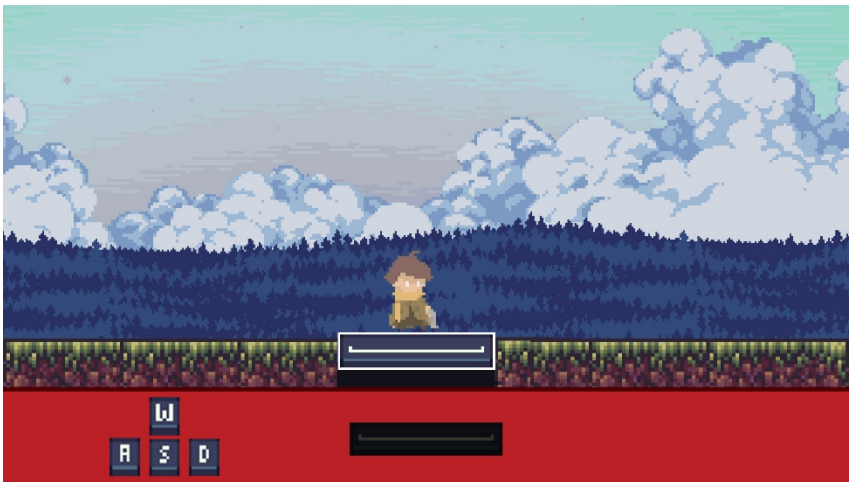
5.4 Gameplay

Aplikasi game yang akan dibuat adalah game yang berbasis pada PC atau *personal computer*. Game ini adalah 2D *platformer* game dengan goal utama mengalahkan musuh yang ada, dan mencapai tujuan akhir level. Untuk mencapai tujuan akhir level, pemain harus melalui beberapa rintangan, seperti jurang, gunung, danau ataupun kondisi *terrain* lain yang menghalangi pemain untuk berjalan lebih lanjut. Pergerakan player menggunakan layout standar seperti w, a, s, d untuk berjalan dan space bar untuk melompat. Namun yang berbeda adalah semua key yang dipakai akan memiliki interface pada layer game.



Gambar 5.4
Konsep Interface Game
(Sumber gambar : Dokumen pribadi)

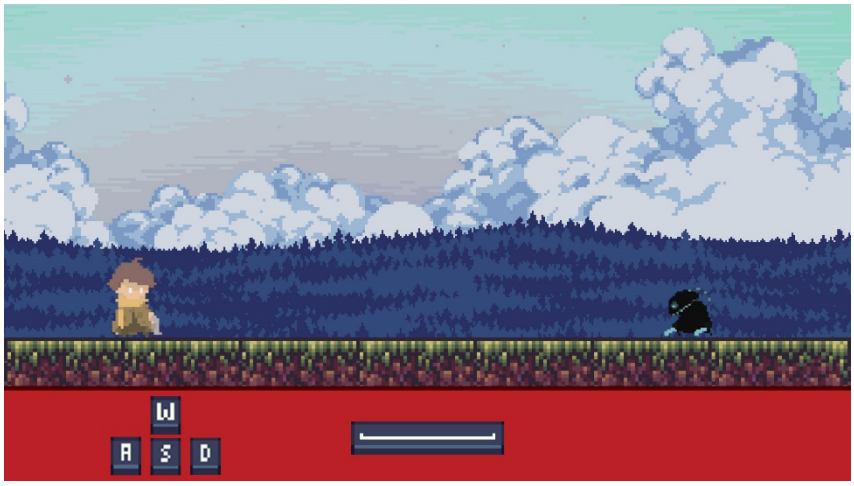
Interface button tersebut dapat digunakan untuk membantu player menyelesaikan halangan yang ada. *Interface button* dapat di click dan drag ke dalam *map* sehingga menjadi bagian dari *map*, atau *object*. *Object key* inilah yang dapat berinteraksi dengan *ingame model, enemy* ataupun *terrain* pada level. Namun jika *key* berada pada *map*, maka fungsionalitas dari *key* tersebut akan hilang. Sehingga walaupun user menekan *key* pada keyboard, game tidak akan merespons *input key* tersebut.



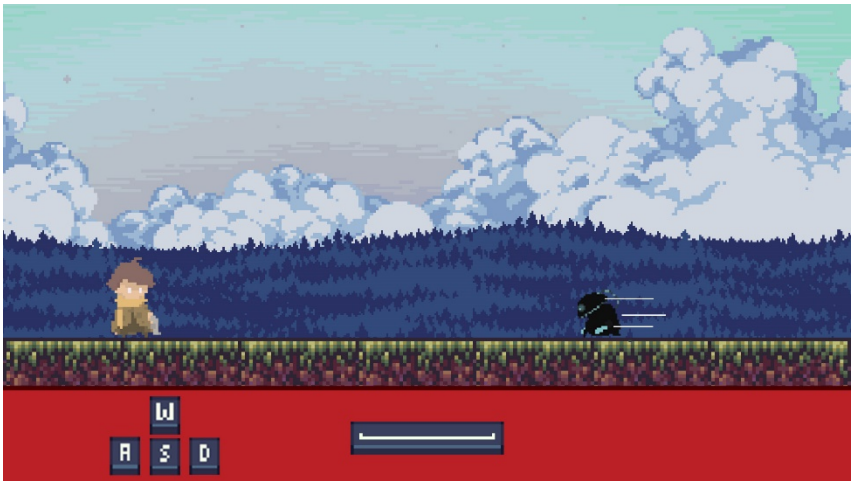
Gambar 5.5
Konsep Usable Key
(Sumber gambar : Dokumen pribadi)

Contoh pada gambar 4.3, tombol spacebar digunakan sebagai jembatan antar dua pulau. Maka pemain dapat berjalan di atas *key* spacebar

tersebut, namun pemain tidak dapat melompat karena tombol spacebar sedang dipakai sebagai *object key*. *Object key* juga dapat digunakan untuk mempengaruhi AI *behaviour*. Dimisalkan terdapat sebuah AI yang agresif kepada pemain, dan AI tersebut berusaha mengejar pemain seperti gambar 4.4 dan 4.5. Hal tersebut diakibatkan *action* AI yang berubah sesuai dengan AI *Behaviour Tree* yang dibuat.



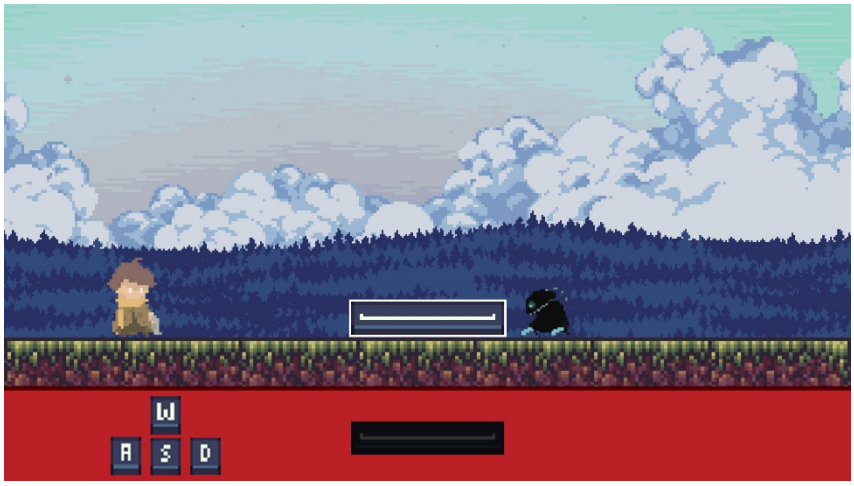
Gambar 5.6
Aggressive AI Behaviour
(Sumber gambar : Dokumen pribadi)



Gambar 5.7
Aggressive AI Behaviour
(Sumber gambar : Dokumen pribadi)

Player dapat menggunakan tombol *interface* lain untuk memanipulasi cara kerja AI. Misalkan pada gambar 4.6, pemain menggunakan tombol shift untuk menghalangi *line of sight* AI. Sehingga AI tidak mengetahui posisi pemain, dan menjadi tidak agresif. Atau seperti

gambar 4.7, pemain menggunakan tombol shift sebagai penghalang jalan agar AI tidak dapat mendekati pemain. Dengan menggunakan *object key*, *action* AI yang awalnya *seek player*, dapat diubah menjadi *idle*.



Gambar 5.8
AI Without Line Of Sight
(Sumber gambar : Dokumen pribadi)

Gambar 5.9
AI Without Path to Chase
(Sumber gambar : Dokumen pribadi)

Hal ini merupakan salah satu mekanik inti untuk menyelesaikan puzzle dan rintangan yang diberikan pada *campaign* nantinya. AI juga dapat merespons dengan aksi yang berbeda jika terdapat *key-key* lain yang digunakan pada map. Mekanik itu akan digunakan hingga level terakhir. Jika level terakhir berhasil terselesaikan, maka *campaign* dianggap tamat.

**Konsep gambar hanya digunakan untuk menjelaskan mekanik game, dan bukan merupakan hasil akhir gameplay maupun tampilan dari game yang akan dibuat.*

Game ini akan mempunyai beberapa peraturan seperti :

- Input kontrol game dilakukan dengan mouse dan keyboard.
- User dapat melanjutkan *campaign* ke level selanjutnya jika level saat ini berhasil terselesaikan.
- Pemain mati jika gagal melalui *obstacle*, terserang musuh atau memilih untuk melakukan *restart* level.
- Jika pemain mati, maka akan bangkit kembali pada *checkpoint* yang tersedia, secara *default*, *checkpoint* adalah posisi start di awal level.
- Pemain dapat menyimpan data *campaign* untuk dilanjutkan lain waktu.

5.5 Fitur

Game ini memiliki beberapa fitur kunci yang dibuat untuk menghasilkan game yang lebih menarik dan interaktif. Fitur pada game ini dapat dibagi menjadi dua kategori utama, yaitu fitur dari game secara umum, dan fitur dari gameplay yang ada dalam permainan.

Secara umum, aplikasi game ini akan memiliki fitur sebagai berikut :

- **Play Campaign**

  Dalam *play mode* ini, player dapat menjalankan *campaign story*. *Campaign* akan terdiri dari beberapa level, level awal akan mencakup *tutorial* dan pengenalan mekanik. Level selanjutnya adalah pengenalan musuh dan tipe musuh, beserta *problem solving* untuk puzzle yang diberikan. Level terakhir akan memiliki *boss battle* yang akan memerlukan mekanik yang telah diberikan kepada pemain untuk mengalahkan boss tersebut. *Campaign* dapat di *pause* dengan *option* menu, dan dapat di *save and load*.
- **Settings**

  User dapat mengubah setting sesuai kebutuhan dan keinginan. Untuk membuka setting, user dapat memilih menu setting pada startup panel game. Jika user masuk ke menu setting, akan ada beberapa submenu yang dapat dipilih user. Sub-menu berupa video setting, dan audio setting. Sesuai Namanya, Video setting digunakan untuk mengubah tampilan grafik, seperti *fullscreen* atau *windowed*. Audio setting digunakan untuk mengubah setting suara game seperti volume *game theme*.

Terdapat beberapa fitur dari *gameplay* sendiri, fitur-fitur tersebut adalah :

- **Checkpoint**

  Fitur *Checkpoint* ini disediakan agar player dapat bangkit kembali jika player melakukan kesalahan seperti terserang musuh, gagal melompati jurang ataupun memilih *restart* level. Akan disediakan

beberapa *checkpoint* di dalam sebuah level. *Checkpoin*t pada awal level adalah posisi awal pemain, yang kemudian akan diperbaharui saat pemain melewati *checkpoint* selanjutnya.

- **Editable Game Object**

Level pada sebuah *campaign* dapat diubah dengan *key* yang dimiliki pemain, sehingga mengubah kondisi level tersebut. *Key* yang dimiliki pemain ini saat berada pada map, akan dianggap sebagai *object* yang dapat digunakan untuk berinteraksi. Interaksi mencakup melompat, berjalan dan *movement* lain yang dapat dilakukan pemain.

- **Multi AI NPC (Friendly)**

Terdapat NPC yang tidak akan menyerang player, NPC ini akan berupa karakter villager pada level pertama di dalam game. NPC ini dapat memiliki beberapa tipe AI. Detail tipe AI yang ada adalah sebagai berikut:

1. Villager akan menggunakan *action* AI *wandering* dan *idle*. Sehingga akan terlihat para villager tersebut seperti berjalan di sekeliling desa
2. Animal (Pet) akan menggunakan action AI *following* dan *idle*. Sehingga hewan seperti kucing atau anjing akan mengikuti pemain saat pemain berada di desa tersebut.

- **Multi AI NPC (Enemy)**

Musuh yang ada pada suatu level bisa memiliki tipe AI yang berbeda. Tipe AI dapat dipengaruhi oleh jenis musuh, *range* senjata musuh (apakah musuh *melee* atau *range*), kemampuan pergerakan musuh (apakah musuh dapat terbang atau hanya berjalan), ukuran *vision cone* musuh dan lain sebagainya. Berikut adalah detail dari NPC *enemy* yang dapat ditemukan di dalam game ini:

1. Dagger Droid merupakan AI *enemy* dengan tipe serangan *melee* dan tipe gerakan berjalan. Dagger Droid harus mendekati pemain terlebih dahulu sebelum dapat menyerang pemain. Sehingga werewolf akan menggunakan *action* AI *patrolling, idle, following,* dan *attacking*. Di mana *following* AI tersebut akan mendekati pemain hingga posisi pemain dan body AI sangat dekat atau bersentuhan, barulah AI akan berusaha menyerang pemain.
2. Iridesent Huntress merupakan AI *enemy* dengan tipe serangan *range* dan tipe gerakan berjalan. Sehingga Iridesent Huntress akan menggunakan action AI *idle*, dan *attacking*. Di mana ninja akan ditempatkan di posisi tertentu, dan akan menyerang pemain jika berada pada area penglihatan atau *line of sight* dari ninja tersebut. Karena tipe serangan iridescent huntress adalah *range*, maka iridescent huntress tidak perlu bergerak untuk menyerang, selama pemain masih terlihat.
3. Shadow Imp merupakan AI *enemy* dengan tipe serangan melee dan tipe gerakan terbang. Shadow Imp harus

mendekati pemain terlebih dahulu sebelum dapat menyerang, namun karena shaman dapat terbang, maka shaman tidak dapat didapat terpengaruh oleh key yang menghalangi *pathing* seperti kedua AI sebelumnya. Sehingga shaman akan menggunakan *action* AI *patrolling, idle, following*, dan *attacking*.

- **AI behaviour Manipulation**

  AI pada musuh juga dapat dimanipulasi oleh *action* dari player. Player dapat melakukan hal seperti menutupi *line of sight* AI, berpindah tempat dari posisi terakhir AI menangkap posisi player dan lain sebagainya. *Manipulation behaviour* ini dapat merubah kondisi AI maupun *action* yang sedang digunakan AI tersebut.

- **Save Campaign**

  Pemain dapat menyimpan data *campaign* yang sedang dimainkan saat ini pada in *game option*. *Save campaign* secara *default* akan melakukan *overwrite save fil*e pada *current campaign* sebelumnya.

- **Load Campaign**

  Jika terdapat data *save file*, maka player dapat melakukan *load file* untuk melanjutkan *campaign* yang disimpan pada *playtime* sebelumnya. Seluruh data player, *map, enemy* akan di *load* sesuai kondisi saat s*ave campaign* dilakukan.

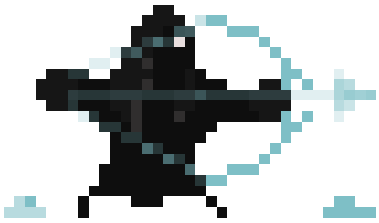
5.6 Mockup Tampilan

Ini adalah beberapa gambar hasil mockup dan model representasi karakter yang akan digunakan untuk game ini.

Gambar 5.10
Mock Up Tampilan Game
(Sumber gambar : Dokumen pribadi)

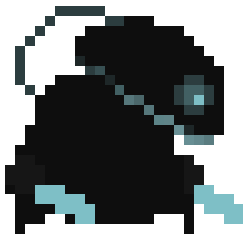
Gambar 5.11
Konsep Character Utama Game
(Sumber gambar : Dokumen pribadi)

Gambar 4.8 adalah gambar tata aturan *interface* dan tampilan game secara umum. Sehingga dapat dilihat perkiraan tinggi map, lebar dan Panjang level serta *terrain* yang ada.

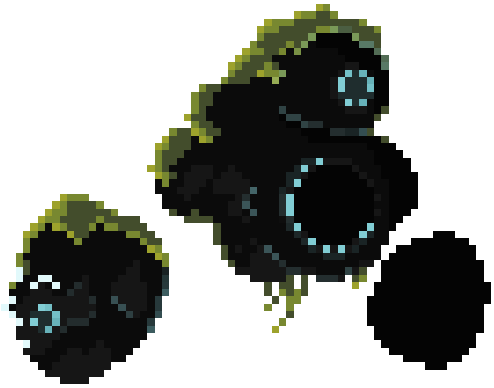


Gambar 5.12
Konsep Gambar Musuh Iridescent Huntress
(Sumber gambar : Dokumen pribadi)

Gambar 5.13
Konsep Gambar Musuh Shadow Imp
(Sumber gambar : Dokumen pribadi)



Gambar 5.14
Konsep Gambar Musuh Dagger Droid
(Sumber gambar : Dokumen pribadi)



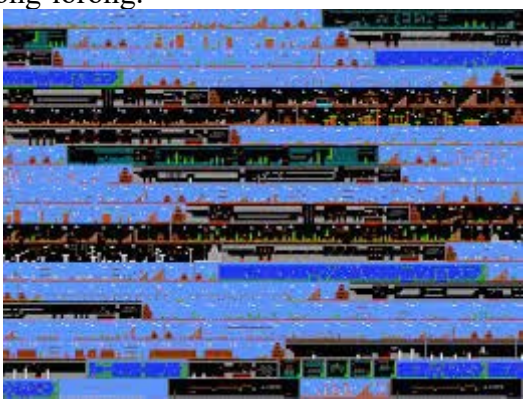
Gambar 5.15
Konsep Gambar Colossal Boss
(Sumber gambar : Dokumen pribadi)

**Konsep gambar dan mockup dapat berubah sesuai keperluan story maupun gameplay pada hasil akhir game*

5.7 Game Referensi

Jika berbicara tentang *platformer* game, tentunya game klasik seperti Super Mario Bros menjadi salah satu game yang paling dikenang. Game ini memiliki genre *platformer action*. Super Mario Bros adalah sebuah permainan yang menggunakan karakter dengan nama Mario yang berusaha menyelamatkan seorang putri dalam sebuah kastil dengan cara mengalahkan musuh yang menghadang di jalan. Dalam perjalanan Mario

dapat mengambil koin untuk point, jamur untuk tumbuh, dan jamur yang berkedip-kedip agar Mario dapat menembak musuh tidak hanya melompat saja. Latar atau map yang digunakan dalam Game Mario Bros berupa daratan dan lorong-lorong.



Gambar 5.16
Super Mario Map
(Sumber gambar : Dokumen pribadi)

Core Gameplay berpusat pada menghindari musuh dan sistem penggerakannya adalah kunci dari *core mechanics*. Fitur lompatan dari sistem bergerak atau *movement* pada Game Super Mario memiliki peranan penting semenjak digunakan untuk menghindari musuh dan mencapai ujung dari scenario Game tersebut. Semua ini bisa saja tercakup dalam aturan desain Game untuk memberikan rasa senang bagi para pemain Game.

Gambar 5.17
Super Mario Gameplay
(Sumber gambar : Dokumen pribadi)

Mario harus benar-benar melompat tepat pada kepala musuh untuk bisa mengalahkan musuh tersebut. Sehingga mengalahkan semua musuh yang ada juga merupakan bagian dari *gameplay*. Berdasarkan mekanik cara player dapat mengalahkan musuh, ini hanyalah versi terselubung dari "melompat". Karenanya bisa disimpulkan bahwa dalam Game Super Mario

Bros, mengalahkan semua musuh adalah sebuah aktivitas *core meta-Gameplay* di mana semua itu bisa dilaksanakan dengan menggunakan sebuah *core mechanics* Game yang seperti melompat, dan diberikan *reward (point)* untuk melakukan *action* melompat (sebuah aktivitas *core Gameplay Mechanic*).

Pada game Super Mario Bros, musuh tidak memiliki AI, atau setidaknya, AI yang ada pada masa tersebut, sudah sangat jarang dikategorikan sebagai AI pada saat ini. Karena tingkat simplisitas dari *intelligence* yang digunakan. Musuh pada game tersebut hanya memiliki *pathing* gerak ke kanan dan kiri, di mana musuh akan bergerak ke satu arah, dan berpindah ke arah lain saat musuh menabrak object, atau masuk ke dalam jurang jika *movement* dilanjutkan. Untuk musuh yang memiliki *movement* lebih dari satu jenis (seperti boss), dilakukan *random* untuk menentukan pergerakan apa yang akan dilakukan. Mengingat hal tersebut, *behaviour* dari musuh tidak dapat diubah dan sangat mudah diprediksi.

5.8 Perbandingan Game

Pembuatan aplikasi game ini merupakan game yang *memiliki system gameplay* yang mirip dengan *gameplay* dengan game Super Mario Bros yang ada pada *classic console* seperti PS1, Nintendo, ataupun *Arcade Machine* kuno lainnya.

Gambar 5.18
Tampilan Super Mario Bros
(Sumber gambar : Start Screen Game Super Mario Bros)

Mengingat AI Mario Bros yang hanya berupa pergerakan NPC ke suatu arah hingga menabrak atau mencapai jurang, maka NPC akan berjalan kearah sebaliknya. Kebanyakan NPC juga tidak memiliki pola serangan, karena damage dihitung saat Mario menyentuh NPC tersebut, tidak peduli apakah saat NPC bergerak, diam maupun pada kondisi lain. Akan terdapat

beberapa perbedaan dengan AI pada NPC di game 'Ctrl'+It. AI pada game ini akan memperhitungkan posisi player sebelum melakukan sebuah pergerakan. Karena AI di game ini menggunakan behavior tree sebagai decision making action AI, dan menggunakan AI moving algorithm untuk melaksanakan action yang ditentukan oleh behavior tree. Sehingga saat melihat player, NPC yang aggressive akan berjalan menuju player dan menyerang. NPC akan berusaha mencari cara tercepat mencapai player, dan menentukan serangan saat player berada di dalam area serangannya.

Pembuatan aplikasi game ini merupakan game yang *memiliki system gameplay* yang mirip dengan *gameplay* dengan game Super Mario Bros yang ada. Berikut ini adalah tabel perbandingan antara game Super Mario Bros. dengan aplikasi game yang akan dibuat :

Tabel 5.1
Tabel Perbandingan Game

No.	Fitur yang Dibandingkan	Super Mario Bros.	'Ctrl'+It
1	2D grafik dan gameplay	✓	✓
2	Singleplayer gameplay	✓	✓
3	Multiplayer Gameplay	×	×
4	Power Up	✓	×
5	AI Based Enemy	×	✓
6	Enemy Path / Patrol Route	×	✓
7	Behavior Tree For AI	×	✓
8	Multiple AI Algorithm	×	✓
9	Multiple AI Enemy	×	✓
10	Capable to Kill Enemy	✓	✓
11	Boss Battle	✓	✓
12	Dynamic Terrain	✓	✓
13	Multiple Level	✓	✓
14	Capable to Add Object to Level	×	✓
15	Capable to Use Object in the Level	×	✓

5.9 Uji Coba

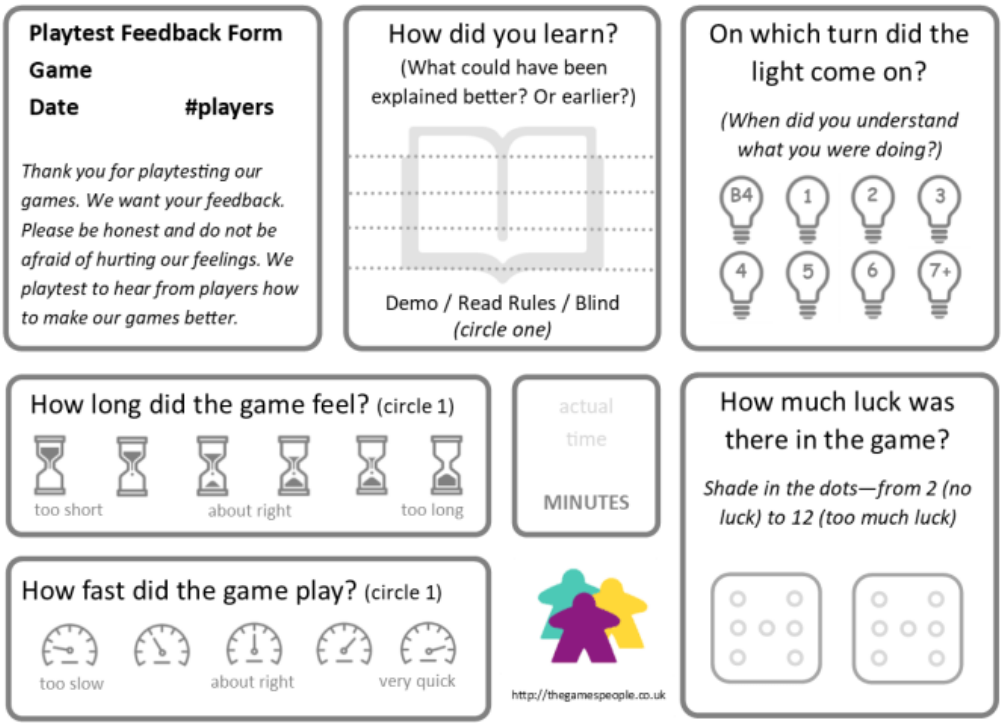
Setelah game *platforming* ini berhasil dibuat, maka akan dilakukan uji coba dengan melibatkan minimal 30 tester. Uji coba akan dilakukan pada laptop atau komputer masing-masing user, dengan ketentuan *operating system* minimal adalah Windows 7 dan versi lebih tinggi. Dengan input berupa keyboard dan mouse.

User akan diminta menyelesaikan semua level yang diberikan. Setelah user berhasil menyelesaikan semua level yang ada, user akan diminta mengisi sebuah *form* tentang opini user akan semua hal yang dirasakan saat bermain. Mulai dari tingkat kesulitan level, tanggapan tentang musuh, desain karakter dan *game story*. User juga akan diminta

melakukan *new game* setelah bermain untuk menguji *overwrite save file function* yang dimiliki.

Pada form yang akan diisi user nantinya, juga akan ditanyakan beberapa informasi mengenai user agar seluruh user yang menjadi tester game ini dapat diklasifikasikan berdasarkan informasi tersebut. Beberapa contoh informasi tersebut akan mencakup umur user saat form diisi, berapa lama user telah bermain video game, berapa banyak game yang telah dimainkan, apakah user memiliki pengetahuan tentang pembuatan video game (programming, art maupun desain) dan lain sebagainya. Sehingga user nantinya dapat diklasifikasikan berdasarkan rentang usia, tipe gamer (casual, avid, hardcore dan lain sebagainya), dan pengetahuan user mengenai video game. Berikut merupakan beberapa referensi mengenai playtesting feedback from yang menjadi acuan dalam pembuatan pertanyaan pada form nantinya.

Gambar 5.19
Form Game Playtesting Feedback Game Salute
(Sumber gambar : Game Salute Playtest Fillable Form)



Gambar 5.20
Form Game Playtesting Feedback The Games People
(Sumber gambar : The Games People Graphical Playtest Feedback Form)

Testing juga dilakukan untuk mengecek apakah ada *bug* yang ditemukan oleh user. Jika user menemukan *bug*, maka user dapat melaporkan *bug* tersebut beserta deskripsi dan *screenshot* saat *bug* terjadi jika memungkinkan untuk membantu *debugging*. Jika *bug* memang benar ada dan dapat di rekreasi secara konsisten, maka akan dilakukan *patching* pada proses pembetulan selanjutnya untuk memperbaiki *bug* tersebut.

VI. Batasan Masalah

Batasan-batasan yang ada pada perancangan game ini secara umum adalah:

1. Game dirancang untuk desktop atau laptop dengan *operating system* windows atau *operating system* lain dengan syarat memiliki *C++ visual redistributable*.
2. Game dimainkan dengan *input* keyboard dan mouse, dengan *output* ke monitor berupa tampilan game dan speaker atau *hardware audio* lainnya berupa suara *theme* dan *gameplay effect*.
3. Game dirancang dengan menggunakan grafik 2D dan 2D gameplay.
4. Tidak memerlukan koneksi internet atau menggunakan *system offline.*
5. Game merupakan game *singleplayer* tanpa *multiplayer* entah berupa *lan, coop,* maupun *internet server connected*.
6. Tidak ada fitur pembelian *item ingame* menggunakan uang sesungguhnya.

Batasan-batasan yang ada pada perancangan game secara spesifik adalah :

1. Game platformer adventure ini dapat digerakan menggunakan tombol w, a, s, d, left shift, dan spacebar. Dengan detail sebagai berikut:
    - ‘w’ digunakan untuk bergerak ke atas, jika terdapat interface seperti lift, tangga, atau pipa yang dapat digunakan untuk naik ke atas.
    - ‘a’ digunakan untuk bergerak ke kiri, jika tidak terdapat halangan, musuh ataupun border level.
    - ‘s’ digunakan untuk bergerak ke bawah, jika terdapat interface seperti lift, tangga, atau pipa yang dapat digunakan untuk turun ke bawah.
    - ‘d’ digunakan untuk bergerak ke kanan, jika tidak terdapat halangan, musuh ataupun border level.
    - ‘spacebar’ digunakan untuk melompat, sehingga player dapat mencapai tempat yang lebih tinggi.
    - Setiap tombol tersebut akan memiliki 2d interface yang dapat digunakan sebagai object pada game.
2. Membuat game *platformer adventure* yang dapat berinteraksi dengan menggunakan tombol mouse left click dan mouse right click. Dengan detail sebagai berikut :
    - ‘left click’ digunakan untuk memilih dan menarik (*drag*) *interface* tombol *movement* untuk dijadikan *object* pada game. *Object* ini dapat digunakan untuk bersembunyi, tempat pijakan maupun penghalang bagi musuh. Selama button berada pada game sebagai *object*, maka pemain tidak dapat menggunakan fungsi button tersebut.
    - ‘right click’ digunakan untuk berinteraksi dengan *object-object* atau NPC pada map.
3. Membuat AI pada game platformer yang memiliki beberapa *action* berupa *idle, patrol,* dan *attack*
4. Membuat AI pada *game platformer* yang memiliki beberapa *moving algorithm* berupa *seek,* dan *wander*.
5. Membuat 3 level dasar dengan detail sebagai berikut :
    - Level 1 sebagai tutorial untuk *game mechanic* yang akan dipakai selama game ini dimainkan.
    - Level 2 yang merupakan level game untuk pengenalan semua musuh dan tipe AI yang ada di dalam game ini.
    - Level 3 yang merupakan boss fight sebagai level terakhir yang harus diselesaikan untuk menamatkan game ini.
6. Membuat setiap level memiliki tingkat kesulitan yang meningkat jika permainan dilanjutkan ke level berikutnya.
7. Membuat level menjadi tidak dapat diselesaikan jika tidak semua *game mechanic* dipakai, namun tetap memberikan fleksibilitas bagaimana user dapat menyelesaikan gamenya.

VII. Metodologi Penyelesaian

Metodologi yang akan digunakan untuk menyelesaikan pembuatan tugas akhir adalah metode scrum dengan langkah-langkah sebagai berikut :

1. Pembuatan dan pengajuan proposal Tugas Akhir.
2. Mencari dan mengumpulkan referensi–referensi mengenai Unreal Engine 4 dan *Development Kit* milik Unreal.
3. Studi kepustakaan tentang cara pembuatan 2D *platformer game*
4. Mencari dan mengumpulkan *sprite, model* dan *resource* untuk pembuatan game.
5. Membuat desain level dan *terrain* yang akan digunakan untuk game.
6. Membuat *code* dan *script* yang akan digunakan di dalam game.
7. Uji coba game pada PC dan laptop dengan Windows *Operating System.*
8. Uji coba game dengan beberapa tester yang berbeda.
9. Lakukan testing agar game berjalan dengan baik tanpa *error*.
10. Mendokumentasi hasil testing dari awal hingga akhir dalam buku laporan tugas akhir.

VIII. Daftar Pustaka

1. Millington, Ian dan Funge, John. 2006. *Artificial Intelligence For Games Second Edition*. San Francisco : Morgan Kaufmann Publishers.
2. Seemann, Glenn dan Bourg, David. 2004. *AI For Game Developers*. California : O'Reilly Media.
3. Marcotte, R dan Hamilton, H. J. 2017. *Behavior Tree for Modelling Artificial Intelligence in Games: A Tutorial*. The Computer Game Journal. DOI:10.1007/s40869-017-0040-9
4. Champandard, A. J. dan Dunstan, P. 2019. *The Behavior Tree Starter Kit: A Tutorial*. CRC Press. DOI:10.1201/9780429055058-3
5. Sekhavat, Yoones A. 2017. *Behavior Tree for Computer Games.* International Journal of Artificial Intelligence Tools 26 (2). DOI:10.1142/S0218213017300010
6. Anonymous. No Year. *"Unreal Engine 4 Documentation". https://docs.unrealengine.com/4.27/en-US/*. Diakses pada 25 Januari 2022.
7. Simpson, Chris. 18 Juli 2014. *"Behavior Tree For AI: How Do They Work". https://www.gamedeveloper.com/programming/behavior-trees-for-ai-how-they-work*. Diakses pada 25 Januari 2022.
8. Anonymous. 2020. *"Mario through the years". https://mario.nintendo.com/history/.* Diakses pada 11 Juni 2021.