Penerapan Mini Environment Linux Menggunakan System.IO Pada Game 3D Kevin in Terminal Console

Diajukan sebagai salah satu persyaratan untuk memperoleh gelar sarjana terapan D-IV Teknologi Permainan Jurusan Desain

Disusun oleh:

Steven Julian

20210077

PROGRAM STUDI D-IV TEKNOLOGI PERMAINAN

JURUSAN DESAIN

POLITEKNIK NEGERI MEDIA KREATIF

JAKARTA

Penerapan Mini Environment Linux Menggunakan System.IO Pada Game 3D Kevin in Terminal Console

Diajukan sebagai salah satu persyaratan untuk memperoleh gelar sarjana terapan D-IV Teknologi Permainan Jurusan Desain

Disusun oleh:

Steven Julian

20210077

PROGRAM STUDI D-IV TEKNOLOGI PERMAINAN

JURUSAN DESAIN

POLITEKNIK NEGERI MEDIA KREATIF

JAKARTA

LEMBAR PENGESAHAN TUGAS AKHIR

Judul Tugas Akhir	: Penerapan Mini Environment Linux Menggunakan System.IO Pada Game 3D Kevin in Terminal Console
Penulis	: Steven Julian
NIM	: 20210077
Program Studi	: Teknologi Permainan
Jurusan	: Desain

Tugas Akhir ini telah dipertanggungjawabkan di hadapan Tim Penguji Tugas Akhir di kampus Politeknik Negeri Media Kreatif pada hari

Kamis, 18 Juli 2024

Disahkan oleh:

Ketua Penguji,

Trifajar Yurmama Supriyanti, S.Kom., MT.,
NIP. 198011122010122003

Anggota 1	Anggota 2
Yeni Nurhasanah, S.Pd., M.T.	Prilly Fitria Aziz, S.Kom., M.Kom.,
NIP. 198607062019032010	NIP. 199104192019032015

Mengetahui,
Ketua Jurusan Desain

Trifajar Yurmama Supriyanti, S.Kom., M.T
NIP. 198011122010122003

LEMBAR PERSETUJUAN SIDANG TUGAS AKHIR

Judul Tugas Akhir	: Penerapan Mini Environment Linux Menggunakan System.IO Pada Game 3D Kevin in Terminal Console
Penulis	: Steven Julian
NIM	: 20210077
Program Studi	: Teknologi Permainan
Jurusan	: Desain

Tugas Akhir ini telah diperiksa dan disetujui untuk disidangkan.

Ditandatangani di Jakarta, 9 Juli 2024

Mengetahui

Pembimbing I	Pembimbing II
Prily Fitria Aziz, M.Kom	**Andrian, S.Kom, M.Kom.**
NIP. 199104192019032015	NIP.198611302020121004

Mengetahui, Koordinator Program Studi Teknologi Permainan

Prily Fitria Aziz, M.Kom

NIP. 199104192019032015

PERYATAAN ORIGINALITAS TUGAS AKHIR DAN BEBAS PLAGIARISME

Yang bertanda tangan dibawah ini:

Nama	: Steven Julian
NIM	: 20210077
Program Studi	: Teknologi Permainan
Jurusan	: Desain
Tahun Akademik	: 2023 -2024

dengan ini menyatakan bahwa Tugas Akhir saya dengan judul:

Perancangan Game 3D Platformer Kevin in Terminal Console Bergenre Puzzle Adventure

adalah original, belum pernah dibuat oleh pihak lain, dan bebas dari plagiarisme.

Bilamana pada kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, saya bersedia dituntut dan diproses sesuai dengan ketentuan yang berlaku.

Demikian pernyataan ini dibuat dengan sesungguhnya dan dengan sebenar-benarnya.

Jakarta, 9 Juli 2024

Yang menyatakan,

Steven Julian

NIM: 20210077

PERNYATAAN PUBLIKASI KARYA ILMIAH

Sebagai civitas academica Politeknik Negeri Media Kreatif, saya yang bertanda tangan di bawah ini:

Nama	: Steven Julian
NIM	: 20210077
Program Studi	: Teknologi Permainan
Jurusan	: Desain
Tahun Akademik	: 2023 - 2024

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Politeknik Negeri Media Kreatif **Hak Bebas Royalti Noneksklusif** (***Non-exclusive Royalty-Free Right***) atas karya ilmiah saya yang berjudul: Perancangan Game 3D Platformer Kevin in Terminal Console Bergenre Puzzle Adventure beserta perangkat yang ada (jika diperlukan).

Dengan Hak Bebas Royalti Noneksklusif ini Politeknik Negeri Media Kreatif berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangakalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Jakarta, 9 Juli 2024

Yang Menyatakan,

Steven Julian

NIM: 20210077

ABSTRAKSI

"Kevin In Terminal Console" incorporates a mini Linux environment using System.IO in the C# programming language. This study identifies several issues, such as the limitations of Linux commands, the constraints of classes and methods within System.IO, and the need for error exceptions to create a user experience akin to a real Linux terminal. The research aims to implement a mini Linux environment in the game, create an authentic playing experience, and provide an understanding of Linux commands for novice players. The Game Development Life Cycle (GDLC) methodology is employed in this development, with phases including initiation, pre-production, production, testing, beta, and release.

Keywords: *Game Programmer, Mini Linux Environment, System.IO, Game Development Life Cycle, Kevin In Terminal Console.*

Game Kevin In Terminal Console mengusung konsep mini environment Linux dengan menggunakan System.IO dalam bahasa pemrograman C#. Studi ini mengidentifikasi beberapa masalah, seperti keterbatasan command Linux, keterbatasan class dan method dalam System.IO, serta perlunya error exception untuk menciptakan pengalaman pengguna yang mirip dengan terminal Linux sebenarnya. Penelitian ini bertujuan untuk menerapkan mini environment Linux dalam game, menciptakan pengalaman bermain yang autentik, dan memberikan pemahaman terhadap command Linux bagi pemain yang awam. Dalam metodologi pengembangan, Game Development Life Cycle (GDLC) digunakan dengan tahapan inisiasi, pra-produksi, produksi, testing, beta, dan release.

Kata kunci: *Game Programmer, Mini Environment Linux, System.IO, Game Development Life Cycle, Kevin In Terminal Console.*

PRAKATA

Dengan penuh rasa syukur, penulis memanjatkan puji dan terima kasih ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penyusunan proposal tugas akhir ini. Proposal ini menjadi langkah awal dan jembatan penting menuju penyelesaian studi di Politeknik Negeri Media Kreatif Jakarta bidang Teknologi Permainan.

Proposal tugas akhir ini tidak akan selesai dengan baik tanpa bantuan, bimbingan, dan dorongan dari orang-orang yang berada di sekitar penulis. Oleh karena itu, penulis ingin mengucapkan terima kasih banyak kepada:

1. Dr. Tipri Rose Kartika., MM., Direktur Politeknik Negeri Media Kreatif.
2. Dr. Handika Dany Rahmayanti, M.Si., Wakil Direktur Bidang Akademik.
3. Trifajar Yurmama Supriyanti, S.Kom., MT., Ketua Jurusan Desain
4. Lani Siti Noor Aisyah, S.Ds., M.Ds, Sekretaris Jurusan Desain
5. Prilly Fitria Aziz, S.Kom., M.Kom., Koordinator Program Studi Teknologi Permainan
6. Prily Fitria Aziz, S.Kom., M.Kom., Pembimbing 1
7. Andrian, S.Kom., M.Kom., Pembimbing 2
8. Para dosen dan tenaga kependidikan Politeknik Negeri Media Kreatif yang telah melayani mahasiswa selama penulis menempuh pendidikan di sini.

Penulis menyadari masih banyak kekurangan dalam laporan magang industri ini. Oleh sebab itu, penulis mengharapkan saran dan kritik yang membangun untuk laporan ini.

Jakarta, 28 Januari 2024

Penulis,

Steven Julian
20210077

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

A. Latar Belakang Masalah

Linux adalah sebuah operasi open-source yang populer yang menarik perhatian para pengguna komputer di seluruh dunia selama beberapa dekade terakhir. Keunggulannya dalam hal stabilitas, keamanan, dan fleksibilitas telah mendorong adopsinya di berbagai sektor, mulai dari server dan workstation hingga perangkat mobile dan embedded. Salah satu fitur utama Linux yang membedakannya dari sistem operasi lain adalah kemampuannya untuk dikontrol melalui terminal, sebuah antarmuka berbasis teks yang memungkinkan pengguna untuk menjalankan perintah dan mengelola sistem dengan presisi dan efisiensi.

Di Indonesia, dengan populasi internet terbesar keempat di dunia, penggunaan Linux masih terbilang minim dibandingkan dengan Windows dan macOS. Data dari Statcounter GlobalStats pada Mei 2024 menunjukkan bahwa pangsa pasar Windows di Indonesia mencapai 84,88%, diikuti oleh macOS dengan 4,63%, dan Linux hanya 2,25%. Hal ini menunjukkan bahwa masih banyak masyarakat Indonesia yang belum familiar dengan Linux dan potensinya. Minimnya popularitas Linux di Indonesia dapat dikaitkan dengan beberapa faktor, antara lain:

1. Kurangnya edukasi dan sosialisasi: Banyak masyarakat Indonesia yang belum mengetahui tentang Linux dan manfaatnya. Hal ini disebabkan oleh minimnya edukasi dan sosialisasi tentang Linux di berbagai media dan platform.
2. Dominasi Windows: Windows telah lama menjadi sistem operasi default di Indonesia, sehingga banyak orang yang terbiasa menggunakannya dan tidak mau beralih ke sistem operasi lain.

3. Kesulitan penggunaan: Bagi pengguna awam, Linux mungkin terlihat rumit dan sulit digunakan, terutama dalam hal navigasi dan penggunaan perintah di terminal.
4. Kurangnya dukungan software: Beberapa software populer yang sering digunakan di Indonesia mungkin tidak tersedia untuk Linux, sehingga menjadi penghalang bagi pengguna untuk beralih.

Meskipun penggunaan Linux di Indonesia masih tergolong rendah, mempelajari perintah Linux dapat memberikan banyak manfaat bagi masyarakat, antara lain:

1. Meningkatkan kontrol dan efisiensi: Perintah Linux memungkinkan pengguna untuk mengontrol sistem dengan lebih presisi dan efisien dibandingkan dengan menggunakan antarmuka grafis.
2. Menyelesaikan masalah dengan cepat: Dengan perintah Linux, pengguna dapat menyelesaikan berbagai masalah sistem dengan cepat dan mudah tanpa harus bergantung pada bantuan teknis.
3. Meningkatkan kemampuan problem solving: Mempelajari perintah Linux dapat melatih kemampuan problem solving dan berpikir logis pengguna.
4. Membuka peluang kerja: Keterampilan Linux semakin diminati di berbagai industri, sehingga membuka peluang kerja yang lebih luas bagi yang menguasainya.
5. Mempelajari perintah Linux mungkin membutuhkan waktu dan usaha, namun manfaat yang diperoleh sepadan dengan usaha yang dikeluarkan. Dengan meningkatkan edukasi dan sosialisasi tentang Linux, serta menyediakan sumber belajar yang mudah diakses, diharapkan minat masyarakat Indonesia untuk mempelajari dan menggunakan Linux akan semakin meningkat.

Game merupakan media hiburan dan edukasi yang cukup menarik bagi masyarakat Indonesia. Melalui game, diharapkan pembelajaran dan pengenalan yang dilakukan di dalam game tersebut dapat tersampaikan dengan baik kepada masyarakat Indonesia.

Kurangnya game yang mengangkat tema tentang linux membuat Masyarakat kurang tertarik juga terhadap sistem operasi Linux. Untuk membuat pengguna internet lebih paham lagi dan mengenalkan tentang sistem operasi linux, penulis membuat game yang mengangkat tema tentang linux dan mengadopsi terminal didalam linux dalam hal mekanik game tersebut. Pengangkatan tema game yang mengadopsi mekanik terminal didalam linux, diterapkan oleh penulis memanfaatkan namespace System.IO dan beberapa exception error untuk membuat pengalaman user lebih nyata.

B. Identifikasi Masalah

Berdasarkan latar belakang pendahuluan yang penulis sudah tulis diatas, berikut adalah identifikasi masalah yang dapat dijabarkan:

1. Kurangnya *game* yang mengangkat tema tentang Linux sebagai bentuk edukasi.
2. Bagaimana *System.IO* bisa dimanfaatkan untuk membuat *environment* linux (*terminal console* dan *filesystem*).

C. Batasan Masalah

Berdasarkan latar belakang dan identifikasi masalah yang telah dipaparkan, batasan masalah dalam penelitian ini adalah:

1. Laporan dan pengerjaan proyek tugas akhir akan berfokus terhadap kurang media sarana dan prasarana pengenalan tentang sistem operasi linux, khususnya di bidang media game.
2. Laporan dan pengerjaan proyek tugas akhir ini akan berfokus pada pemanfaatan *System.IO* dalam pembuatan *environment* linux (*terminal console* dan *filesystem*).

D. Rumusan Masalah

Dalam pembuatan game Kevin In Terminal Consol memiliki rumusan masalah sebagai berikut:

1. Apa saja tantangan yang dihadapi dalam mengembangkan game edukasi tentang linux Kevin in Terminal Console?
2. Apa saja kelas-kelas yang ada didalam System.IO yang bisa dimanfaatkan dalam pembuatan environment linux di game Kevin in Terminal Console?
3. Apakah fitur simulasi *environment* linux setidaknya bisa didapatkan dan diingat oleh *player* setelah memainkan game Kevin in Terminal Console?

E. Tujuan Penelitian

Tujuan penelitian pada game Kevin In Terminal Consol sebagai berikut:

1. Mengidentifikasi faktor-faktor yang menyebabkan rendahnya penggunaan linux di Indonesia.
2. Mengembangkan game edukasi tentang linux yang menarik dan interaktif untuk memperkenalkan sistem operasi linux terhadap masyarakat di Indonesia.
3. Membuat orang-orang yang awam terhadap linux setelah memainkan game ini mendapatkan pengalaman.

F. Manfaat Penelitian

1. Bagi Player

   Memberikan pengalaman user dalam memainkan game Kevin In Terminal Console menjadi nyata.
2. Bagi Politeknik Negeri Media Kreatif Jakarta

   Penulisan ini menjadi acuan untuk meningkatkan daya tarik masyarakat terhadap Linux dan meningkatkan masyarakat umum dalam membuat game bertema *environment* Linux.
3. Bagi Penulis.

Dapat mengembangkan dan menerapkan ilmu yang telah dimiliki dan dipelajari selama masa perkuliahan dan juga dapat menjadikan persiapan dalam dunia kerja.

BAB II
KAJIAN SUMBER

A. Linux

Linux adalah sebuah sistem operasi berbasis Unix yang bersifat *open-source*. Dikembangkan oleh Linus Torvalds pada tahun 1991, Linux menjadi salah satu sistem operasi yang paling populer di dunia, terutama dalam lingkup server dan sistem *embedded*.

B. Perintah Dasar Linux

Dalam pengoperasiannya, Linux merupakan sistem operasi yang berbasis *text*. Untuk melakukan beberapa operasi di Linux, pengguna dapat mengetikkan beberapa perintah dasar didalam Linux. Perintah-perintah tersebut disebut sebagai *Command Line*, sedangkan untuk perintah dasar pada linux disebut *Basic Command Line*.

Cat	Menampilkan isi file di layar
Cd	Digunakan untuk berpindah direktori
Ls	Digunakan untuk menampilkan sebuah direktori
Mkdir	Membuat sebuah direktori
Passwd	Digunakan untuk mengganti kata sandi

Berikut adalah perintah dasar linux yang biasa digunakan menurut Erlangga Purwa pada buku Perintah Dasar Linux (2014).

C. Namespace

Berdasarkan dokumentasi di web microsoft tentang C#, *Namespace* adalah cara untuk mengelompokkan kelas, struktur, dan antarmuka yang terkait. Hal ini membantu untuk mengatur kode dan menghindari konflik nama antara kelas dari *library* yang berbeda. Berikut adalah beberapa list namespace yang ada di C#:

1. *System: Namespace* dasar yang berisi kelas-kelas fundamental seperti *Object, String, Console*, dll.
2. *System.Collections*: Berisi koleksi non-generik seperti *ArrayList, Hashtable*, dll.
3. *System.Collections.Generic*: Berisi koleksi generik seperti *List<T>, Dictionary<TKey, TValue>,* dll.
4. *System.Linq*: Menyediakan fungsionalitas *Language-Integrated Query (LINQ)* untuk manipulasi data.
5. *System.Text*: Berisi kelas-kelas untuk pengelolaan teks, seperti *StringBuilder, encoding*, dll.
6. *System.IO*: Berisi kelas-kelas untuk operasi *input/output* seperti *File, Stream, StreamReader*, dll.
7. *System.Threading*: Berisi kelas-kelas untuk pemrograman *multi-threading seperti Thread, Task, Mutex*, dll.
8. *System.Net*: Berisi kelas-kelas untuk operasi jaringan seperti *HttpClient, WebClient*, dll.
9. *System.Xml*: Berisi kelas-kelas untuk bekerja dengan XML, seperti *XmlDocument, XmlReader, XmlWriter*, dll.
10. *System.Data*: Berisi kelas-kelas untuk operasi *database* seperti *DataSet, DataTable, SqlConnection*, dll.

D. System.IO

System.IO merupakan *Namespace* didalam C# yang memungkinkan untuk membaca dan menulis sebuah *file* dan aliran data, serta menyediakan dukungan

dasar untuk *file* dan direktori. *System.IO* mempunyai beberapa kelas atau *methode* yang mendukung aliran data. Berikut adalah daftarnya:

1. *BinaryReader*: Membaca data primitif dari aliran biner.
2. *BinaryWriter*: Menulis data primitif ke aliran biner.
3. *BufferedStream*: Menyediakan *buffer* untuk aliran lain, untuk meningkatkan efisiensi baca/tulis.
4. *Directory*: Menyediakan metode statis untuk membuat, memindahkan, dan menamai ulang direktori.
5. *DirectoryInfo*: Menyediakan properti dan metode *instance* untuk membuat, memindahkan, dan menamai ulang direktori.
6. *DriveInfo*: Menyediakan informasi tentang *drive* sistem.
7. *EndOfStreamException*: Pengecualian yang dilemparkan ketika mencoba membaca melewati akhir aliran.
8. *File*: Menyediakan metode statis untuk membuat, menyalin, menghapus, memindahkan, dan membuka *file*, serta membantu dalam pembuatan objek *FileStream*.
9. *FileInfo*: Menyediakan properti dan metode *instance* untuk membuat, menyalin, menghapus, memindahkan, dan membuka file.
10. *FileStream*: Menyediakan aliran untuk membaca dan menulis ke file.
11. *FileSystemInfo*: Kelas dasar abstrak untuk *FileInfo* dan *DirectoryInfo*.
12. *MemoryStream*: Menyediakan aliran memori yang mendukung pengalihan data ke *array byte*.
13. *Path*: Menyediakan metode untuk bekerja dengan *string* yang berisi informasi jalur (*path*).
14. *Stream*: Kelas dasar abstrak untuk aliran data.
15. *StreamReader*: Membaca karakter dari aliran *byte* tertentu dalam *encoding* tertentu.
16. *StreamWriter*: Menulis karakter ke aliran *byte* tertentu dalam *encoding* tertentu.

17. *StringReader*: Mengimplementasikan aliran input yang membaca dari *string*.
18. *StringWriter*: Mengimplementasikan aliran output yang menulis ke *string*.
19. *TextReader*: Kelas dasar abstrak untuk membaca blok karakter.
20. *TextWriter*: Kelas dasar abstrak untuk menulis blok karakter.

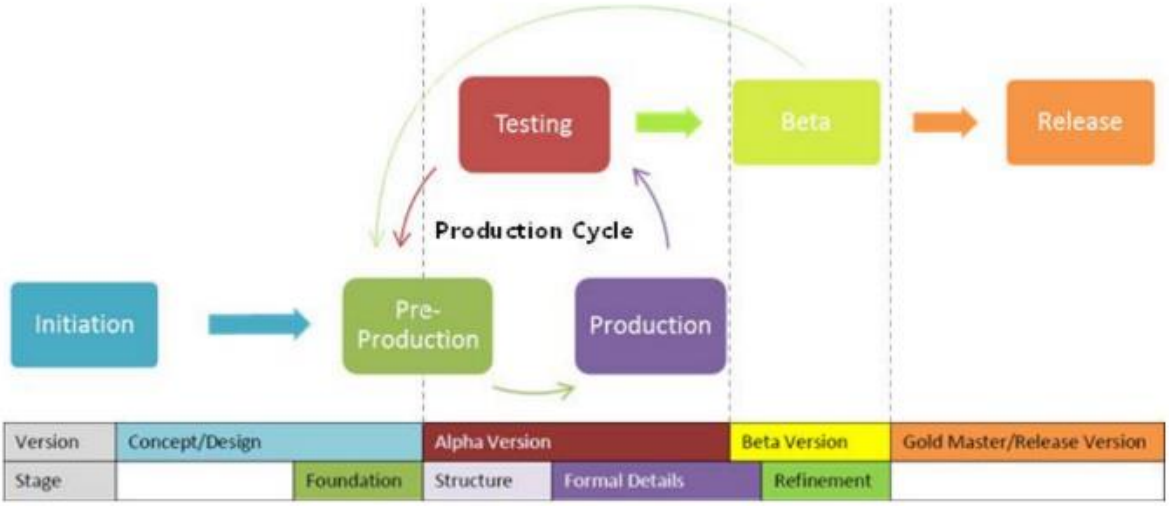
E. Unity

Unity adalah *game engine* yang dirancang untuk membuat *game* dengan antarmuka yang mudah digunakan dan grafis berkualitas tinggi menggunakan OpenGL dan DirectX, sangat cocok untuk sistem operasi 64-bit (Badri & Al Habib, 2020). Unity juga merupakan *game engine multiplatform* yang memungkinkan game dapat dipublikasikan ke berbagai platform seperti *Standalone* (.exe), web, Android, iOS, XBOX, dan PS3. Meskipun Unity dapat dipublikasikan ke berbagai *platform*, beberapa *platform* memerlukan lisensi khusus. Namun, Unity menyediakan versi gratis untuk pengguna yang memungkinkan publikasi dalam bentuk Standalone (.exe) dan web (Supardi, 2021).

Gambar 2.1 Unity Engine

F. Game Development Life Cycle (GDLC)

Menurut Alamsyah M. R. (2022), pengembangan *game* membutuhkan panduan khusus untuk mendukung prosesnya. Panduan ini dikenal sebagai *Game Development Life Cycle (GDLC)*. *GDLC* adalah metode pengembangan *game* yang melibatkan tahapan berulang. Definisi lain menyatakan bahwa *GDLC* adalah proses pengembangan game yang terdiri dari enam tahap utama: inisiasi, pra-produksi, produksi, pengujian, *beta test*, dan rilis. Tahapan *GDLC* dapat dilihat pada gambar di bawah ini



Gambar 1.2 GDLC Table (Ariyana, Susanti, Ath-Thaariq, & Apriadi, 2022

BAB III
METODOLOGI PENCIPTAAN

A. Metodologi Pengembangan

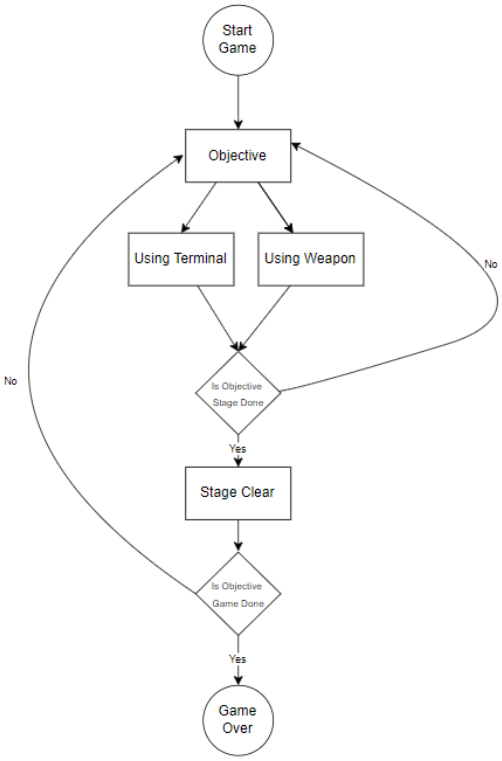
Game Development Life Cycle (GDLC) merupakan sebuah metode yang menangani pengembangan *game* dimulai dari titik awal hingga paling akhir. Dimulai dari tahap pembuatan ide dan konsep mengenai game yang akan dibuat, sedangkan tahap akhir dari *game development* adalah saat game dirilis. *GDLC* menggunakan pendekatan bertahap atau tahapan-tahapan untuk melakukan analisa dan membangun game menggunakan siklus yang spesifik dan lebih kompleks. Berikut adalah penjelasan dari tahap-tahap dalam sebuah GDLC.

1. Inisiasi

Tahap inisiasi adalah titik awal dari pengembangan game, di mana ide game dirumuskan. Pada tahap ini, developer berkumpul untuk *brainstorming* dan berdiskusi mengenai game seperti apa yang akan dibuat.

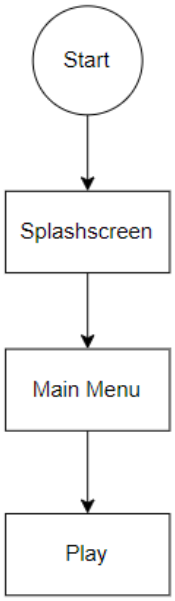
2. Pra-Produksi

Tahap pra-produksi adalah tahap perencanaan dan perancangan game. Pada tahap ini, *game design* didokumentasikan dalam *Game Design Document (GDD)* dan *prototype* dari game dibuat. Sebagai Programmer, ditahap ini programmer merancang *screenflow* untuk peraliran dari 1 *screen* ke *screen* yang lain, *flowchart* untuk mengetahui alur dari sebuah game yang akan dibuat, *onion design* untuk menentukan prioritas dari fitur yang harus diimplementasikan di game tersebut dan menentukan metode-metode apa saja yang diperlukan untuk mencapai mekanik yang diinginkan.



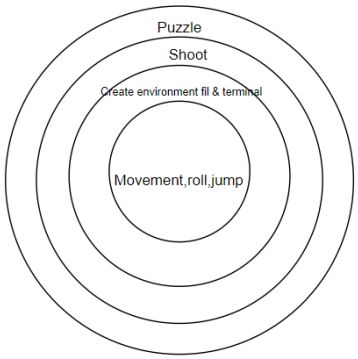
Gambar 2.1 Flowchart sederhana yang dibuat programmer

Flowchart diatas adalah alur dari sebuah *game* dan objektif dari mekanik game yang dibuat, player bisa memusnahkan musuh dengan cara mengetikan perintah linux ataupun menembak musuh sampai objektif *stage* selesai.



Gambar 3.2 Screenflow sederhana yang dibuat programmer

Screenflow diatas adalah alurasi dari perpindahan *screen game*, dimana ketika *start* akan ditampilkan *splashscreen* dan juga *video* animasi singkat, lalu ke *main menu*, dari *main menu* kita bisa ke *stage*.

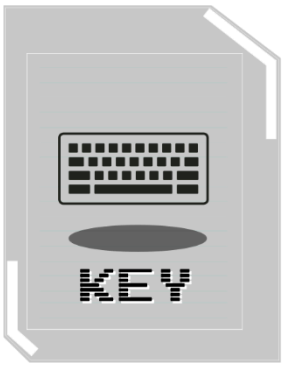


Gambar 3.3. Onion Design

Onion Design diatas dibuat untuk menentukan fitur mana yang terlebih dahulu didevelop berdasarkan lingkaran paling dalam.

A. Perancangan Kontrol Game

Game ini dimainkan menggunakan *keyboard* dan *mouse* dengan penjelasan kontrol seperti ini:



W – A – S – D	Movement
Shift	Run
CTRL	Roll
Spacebar	Jump
Tab	Open Terminal
LMB	Shoot
F	Interact

Gambar 3.4 Kontrol Game

Dari penjelasan diatas, apabila karakter ingin bergerak, bisa menggunakan W – A – S – D. Ketika Tombol W ditekan, karakter akan bergerak maju sedangkan A untuk mundur, W untuk ke kiri dan D untuk ke kanan. Player juga bisa melakukan *sprint/run* menggunakan tombol SHIFT, lompat dengan SPACEBAR, berputar menggunakan tombol LEFT CTRL, membuka terminal menggunakan TAB dan interaksi dengan benda disekitar menggunakan tombol F.

B. Perancangan Mekanik Utama Mini Environment Linux

Programmer memanfaatkan System.IO untuk penerapan mini environment linux. System.IO dipilih karena kemampuan kelas didalamnya untuk mengolah dan mengalirkan data. Berikut adalah list table dari kelas-kelas beserta fungsi yang diterapkan didalam game:

Table 1. Daftar Kelas di namespace System.IO dan Fungsinya penerapannya

File.Exists	Berfungsi untuk memeriksa apakah file tersebut ada, misalnya ketika player memasukkan command untuk membaca file, maka file tersebut harus diperiksa dulu apakah ada atau tidak dan jika tidak maka terminal akan memberikan *response* "*file doesn't exists"* dan proses pembacaan file tidak berhasil.
File.Delete	Berfungsi untuk menghapus file, misalnya ketika player memasukkan command untuk menghapus file maka file tersebut akan dihapus menggunakan kelas tersebut.
File.WriteAllLines	Berfungsi untuk membuat sebuah file dan menuliskan string ke dalamnya, misalnya sistem otomatis akan membuat file enemy1.txt di level 1 jika game di jalankan dengan isi file

	tersebut berupa *health, damage, speed* dan atribut lainnya.
File.ReadAllText	Berfungsi untuk membaca isi dari sebuah file, misalnya ketika si player menggunakan command untuk membaca enemy1.txt, maka nantinya isi dari file tersebut akan dikeluarkan.
Path.Combine	Berfungsi untuk melakukan combine antara path default dan nama folder untuk setiap scenes.
Path.GetFileNameWithoutExtension	Berfungsi untuk mendapatkan nama file tetapi tidak beserta extensinya, contohnya ketika Player menuliskan perintah untuk menghapus file enemy1.txt maka sistem akan membaca enemy1 saja dan akan menghapus GameObject enemy1.
Path.GetExtension	Berfungsi untuk membaca extensi dari

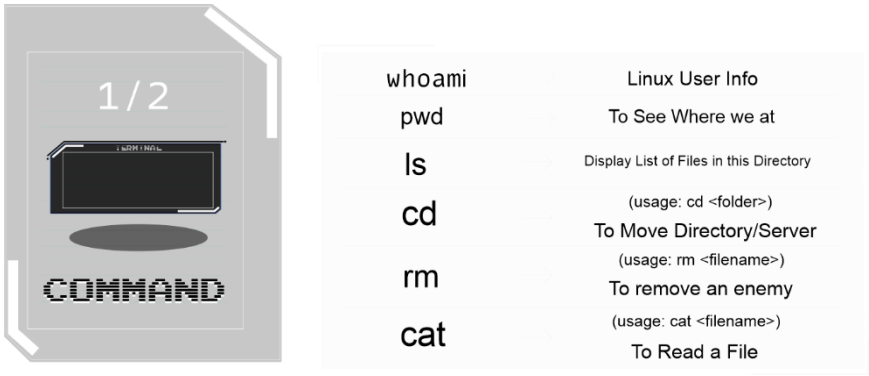
	nama file yang dipilih, contohnya ketika di level2 player harus menghapus file .mp3 dengan maksimal 3x penghapusan file mp3 jika salah maka gameover, maka di kode akan melakukan validasi untuk membaca kalau 3x salah itu benar-benar file dengan extensi .mp3 dan bukan file lain seperti enemy1.txt
Directory.Exists	Berfungsi untuk memerika apakah folder directory tersebut ada atau tidak, contohnya untuk level1 adalah representasi dari folder documents, maka sistem akan membuat sebuah folder documents, tetapi sebelum itu dilakukan pengecekan terlebih dahulu ada atau tidak folder documents.
Directory.CreateDirectory	Berfungsi untuk membuat sebuah folder baru, misalnya ketika

	game dijalankan ada 2 folder didalam persistentdatapath yang akan dibuat yaitu, documents dan music.
Directory.GetFiles	Berfungsi untuk mendapatkan isi dari dalam directory yang dipilih, contohnya ketika player menuliskan perintah ls maka sistem akan membaca isi dari folder tempat kita berada.
FileInfo	Berfungsi untuk mendapatkan data-data dari file yang dipilih seperti nama, size, terakhir diubah dan permissionnya.
FileAttributes	Berfungsi untuk melihat berbagai macam dari attribut file yang dipilih seperti apakah file tersebut tersembunyi atau apakah file tersebut read only.

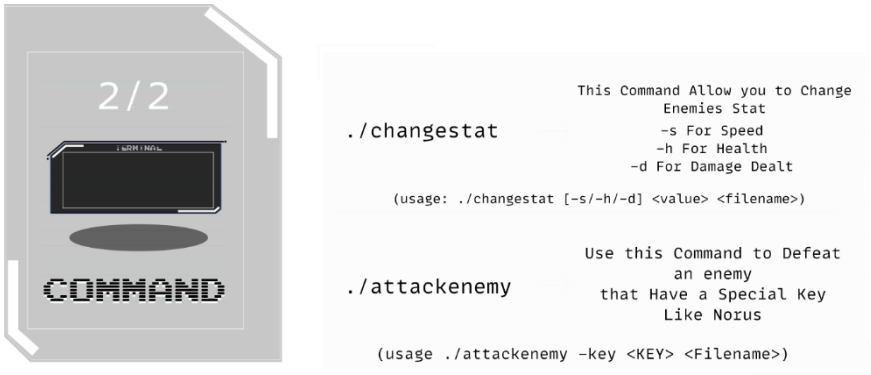
C. Perancangan Perintah Linux

Programmer memilih perintah apa saja yang memungkinkan untuk diadopsi dari list perintah yang diberikan oleh game designer,

maka didapatlah beberapa perintah yang memungkinkan untuk diterapkan:



Gambar 3.5 Perintah Linux 1 dari 2



Gambar 3.6 Perintah Linux 2 dari 2

Berdasarkan gambar diatas, penulis akan menjelaskan tentang command yang diterapkan beserta dengan mekanisme penerapannya di dalam game:

Table 2. Command Linux yang tersedia di game

whoami	Disini programmer hanya menerapkan print biasa yaitu user yang dipakai player seolah-olah bahwa player sedang berada didalam linux secara ril.
pwd	Programmer akan membaca persistentpath atau folder persistent yang dibuat sistem dan menggabungkannya dengan folder atau level dia

	sekarang. Misalnya, player berada dilevel2, level2 adalah representasi dari folder music maka ketika player mengetik pwd terminal akan menampilkan persistentdatapath + namascene
ls	Programmer menerapkan Directory.GetFiles berdasarkan folder tempat player berada. Saat file atau folder yang ada didalam folder tempat kita berada didapatkan, maka akan disimpan diarray untuk nantinya dilihat attribut dari file-file tersebut seperti nama file, ukuran file, terakhir diubah dan permissionnya. Untuk melihat attribut-attribut tersebut digunakan FileInfo. Ketika sudah didapatkan semuanya maka semua informasi tersebut akan di print di terminal.
cd	Programmer menerapkan cd untuk membuat player pindah folder atau scene atau level. Di GameManager, akan dibuat folder sesuai dengan nama scene kita sekarang, misalnya kita berada di scene music maka GameManager akan membuat folder music untuk meletakkan file enemy, file interaksi dan lain-lain, ketika player menuliskan perintah cd music maka yang terjadi sebenarnya adalah sistem melakukan SceneManager.LoadScene(“music”) sehingga player pindah ke scene music.
rm	Programmer menerapkan rm file berdasarkan gameobject. Gameobject yang direpresentasikan sebagai file akan membuat file .txt/mp3 ataupun extensi lainnya berdasarkan nama GameObject mereka, contohnya nama GameObject enemy1

	akan membuat file dengan nama enemy1.txt. Saat player memerintahkan perintah untuk hapis, maka yang terjadi ada 2 yaitu: 1. Sistem akan menghapus GameObject berdasarkan nama file yang diketik (itu terjadi karena nama file dan nama GameObject sama) 2. Sistem juga akan menghapus file yang diketik oleh player.
cat	Programmer menerapkan cat untuk membaca isi file yang diketik dengan memanfaatkan methode File.ReadAllText dan menampilkannya di terminal
./changestat	Programmer menerapkan script bash changestat untuk mengubah isi file dari file yang dipilih berdasarkan opsi yang dimana nantinya setelah file diubah maka variabel tersebut akan dilemparkan ke inspector juga sehingga perubahan akan diterapkan.
./attackenemy	Programmer menerapkan script ini seperti menerapkan script rm, bedanya hanya ada validasi key saja yang diciptakan untuk menambahkan efek sulit yang diciptakan, key ini bisa didapatkan juga dengan membaca isi file terlebih dahulu.

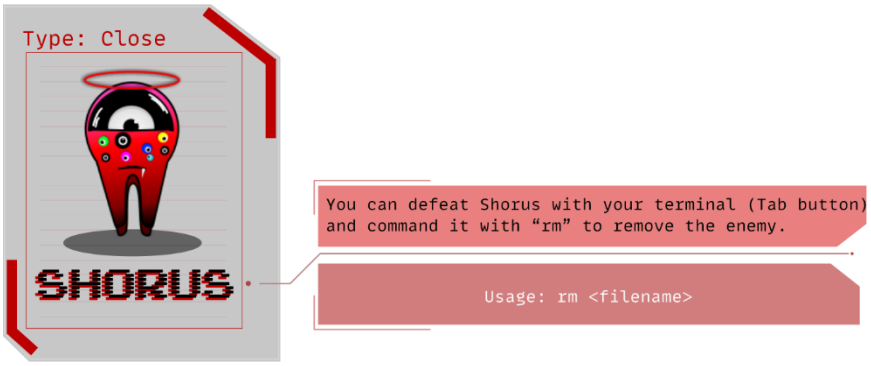
D. Perancangan Movement, Jump Dan Roll

Programmer menerapkan sistem pergerakan player berdasarkan input pemain dan elemen-elemen fisika. Sistem menangkap input horizontal dan vertikal dari pemain untuk menentukan arah gerakan. *Input* ini kemudian diterjemahkan menjadi vektor pergerakan. Contoh: Tekanan tombol "W" dan "A" secara bersamaan akan membuat pemain bergerak ke depan dan ke kiri. Vektor pergerakan yang dihasilkan dari input pemain kemudian digunakan untuk menggerakkan pemain dengan kecepatan yang telah ditentukan.

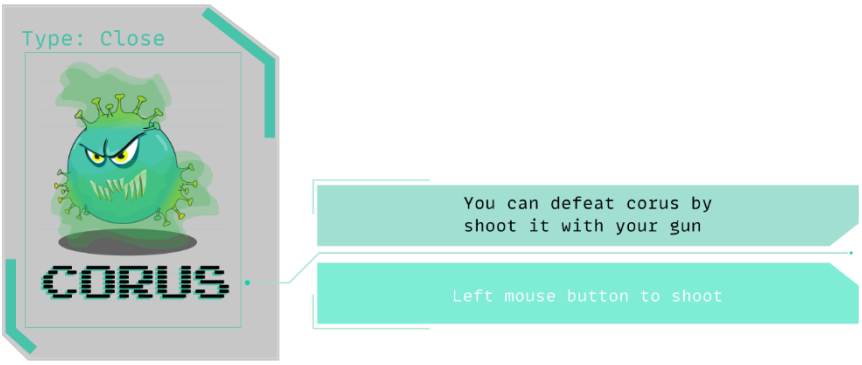
Untuk fitur *jump*, memungkinkan pemain untuk melompat saat berada di tanah dan stamina mencukupi. Saat tombol lompat ditekan, pemain akan melompat dengan kekuatan yang telah ditentukan, dan stamina akan berkurang. Contoh: Pemain menekan tombol lompat, karakter melompat ke udara dengan kekuatan *jumpForce* dan stamina berkurang 2 unit. Untuk fitur *roll*, Pemain dapat melakukan gerakan *roll* saat tombol tertentu ditekan dan stamina mencukupi. Gerakan *roll* ini akan mengubah ukuran dan posisi *collider* untuk menyesuaikan dengan animasi *roll*. Contoh: Pemain menekan tombol untuk *roll*, karakter melakukan gerakan berguling ke depan dengan durasi dan kekuatan yang telah ditentukan.

E. Perancangan Enemy

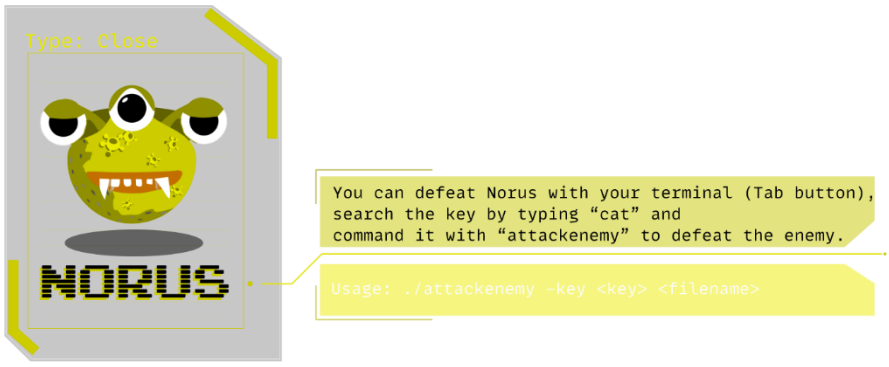
Setiap *enemy* memiliki atribut seperti *health*, *speed* dan *damage*. atribut tersebut akan dibuat di *File* dengan nama *file* sesuai dengan nama *GameObject* dan atribut tersebut akan di*assign* ke dalam *inspector GameObject* terkait. Setiap *Enemy* juga memiliki kemampuan untuk *aggro* ketika berada didekat *player*, yaitu bertindak aggresif dan *sprint* ke arah player secara otomatis jika berada dalam jangkauan. Berikut adalah 3 varian *enemy* yang telah dirancang. Untuk *Enemy* 1 bisa dikalahkan menggunakan command rm <*filename*>, *Enemy* 2 bisa dikalahkan dengan menembak *enemy* tersebut, sedangkan *Enemy* 3 bisa dikalahkan menggunakan *command ./attackenemy -key <KEY> <filename>*. Untuk lebih lengkapnya informasi tersebut tersedia pada gambar dibawah.



Gambar 3.7. Enemy 1 dari 3



Gambar 3.8. Enemy 2 dari 3



Gambar 3.9. Enemy 3 dari 3

3. Produksi

Tahap produksi adalah tahap di mana *game* diimplementasikan berdasarkan *GDD* dan *prototype*. Pada tahap ini, aset *game* dibuat, program *game* ditulis, dan asset dan program diintegrasikan.

Sebagai *game programmer* dalam pembuatan Game Kevin In Terminal Console, langkah-langkah yang harus dilakukan dalam tahap produksi adalah:

a. Membaca dan memahami *GDD* dan *Prototype Game*.

   Hal ini penting dilakukan untuk memastikan bahwa Game Programmer memiliki pemahaman yang jelas tentang mekanik dan keseluruhan Game yang akan dibuat.

b. Membuat desain arsitektur game di Unity.

   Mengikuti panduan *GDD* yang sudah ada, game programmer harus menentukan komponen-komponen apa saja yang perlu lebih dahulu untuk diintegrasikan ke dalam game.

c. Menulis kode game.

   Game Programmer tentu saja bertanggung jawab dalam penulisan kode agar game berjalan sesuai dengan GDD dan berfungsi dengan baik.

A. Pembuatan Movement Controller Player

Pada tahap produksi, untuk menerapkan *movement control* untuk *player*, *programmer* membuat script *PlayerController* yang berisi pergerakan *player*, melompat, *rolling* dan manajemen stamina.

1. Untuk pergerakan menggunakan *rigidbody* berdasarkan input horizontal dan vertikal dari keyboard diatur berdasarkan orientasi gerakan kamera.

   ```
   rb.velocity = new Vector3(moveDirection.x * speed, rb.velocity.y, moveDirection.z * speed);
   ```

2. Sedangkan untuk Lompat, ketika tombol *space* ditekan maka menerapkan gaya lompat ke *RigidBody* dengan *ForceMode.Impulse*

   ```
   rb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
   ```

3. Saat pemain berguling, perpindahan posisi (*transform.position*) ditambah dengan *rollDirection* (arah depan pemain) dikalikan dengan *rollForce* (kecepatan berguling) dan dikali dengan *Time.deltaTime* untuk menghitung pergerakan per *frame*.

   ```
   transform.position += rollDirection * (rollForce * Time.deltaTime);
   ```

4. Saat pemain berlari cepat, *speed* diatur menjadi *sprintSpeed* untuk meningkatkan kecepatan pergerakan.

B. Pembuatan Mini Environment Linux

Programmer membuat script *GameManager* untuk mengatur logika permainan dan pembuatan *mini environment* linux. *GameManager* dikatakan begitu karena didalam *GameManager programmer* memasukkan *methode* untuk menghapus semua *file* yang telah dibuat sistem lalu membuat folder baru lagi sesuai dengan

scene player berada. Berikut adalah penjabaran metode-metode yang telah dibuat:

1. removeAllFiles(): Menghapus semua *file* dalam folder *folderPath* jika ada.

```
void removeAllFiles()
{
    if (Directory.Exists(folderPath))
    {
        string[] files = Directory.GetFiles(folderPath);
        foreach (string file in files)
        {
            File.Delete(file);
            print("File deleted: " + file);
        }
    }
    else
    {
        print("Folder does not exist: " + folderPath);
    }
}
```

2. createfolderEnemy(): Membuat folder untuk menyimpan data musuh jika folder tersebut belum ada.

```
void createfolderEnemy()
{
    if (!Directory.Exists(folderPath))
    {
        Directory.CreateDirectory(folderPath);
        print("Folder created successfully.");
    }
    else
    {
        print("Folder already exists.");

    }
}
```

Setelah programmer membuat script *GameManager*, lalu ada *script* Enemy juga yang dipasang pada masing-masing enemy yang tujuannya untuk membuat sebuah file enemy di folder yang telah dibuat saat GameManager dijalankan, berikut penjabarannya:

1. createfileEnemy(): Membuat file untuk menyimpan atribut musuh jika file tersebut belum ada. Menyertakan nilai-nilai default untuk atribut seperti health, speed, damage, dan lainnya.

```
void createfileEnemy()
{
    string enemyFilePath = Path.Combine(folderPath, filenameCreate);


    if (!Directory.Exists(folderPath))
    {
        try
        {
            Directory.CreateDirectory(folderPath);
            Debug.Log("Created directory: " + folderPath);
        }
        catch (System.Exception ex)
        {
            Debug.LogError("Failed to create directory: " + ex.Message);
            return;
        }
    }

    if (!File.Exists(enemyFilePath))
    {
        try
        {
            int randomRange = Random.Range(0, 100000);
            string fileContent = "Health=100\nSpeed=1\nDamage=1\ncanAttack=true\ncanRemove=false\nKey="+randomRange;
            File.WriteAllText(enemyFilePath, fileContent);
            Debug.Log("File created successfully at: " + enemyFilePath);
        }
        catch (System.Exception ex)
```

```
        {
            Debug.LogError("Failed to create file: " +
ex.Message);
        }
    }
    else
    {
        Debug.Log("File already exists at: " +
enemyFilePath);
    }
}
```

2. checkattributeEnemy(): Membaca file atribut musuh jika file tersebut ada, kemudian memuat nilai-nilai atribut ke dalam variabel yang sesuai seperti health, speed, damage, canRemove, dan randomKey.

```
void checkattributeEnemy()
{
    if (File.Exists(filePath))
    {
        try
        {
            string[] lines =
File.ReadAllLines(filePath);
            foreach (string line in lines)
            {
                string[] keyValue = line.Split('=');
                if (keyValue.Length == 2)
                {
                    string key = keyValue[0].Trim();
                    string value = keyValue[1].Trim();

                    switch (key)
                    {
                        case "Health":
```

```
                    health = float.Parse(value);
                    break;
                case "Speed":
                    speed = float.Parse(value);
                    break;
                case "Damage":
                    damage = float.Parse(value);
                    break;
                case "canRemove":
                    canRemove = bool.Parse(value);
                    break;
                case "Key":
                    randomKey = int.Parse(value);
                    break;
            }
        }
    }

    Debug.Log($"Health: {health}, Speed: {speed}, Damage: {damage}, CanRemove: {canRemove}, Key: {randomKey}");
    }
    catch (System.Exception ex)
    {
        Debug.LogError("Error reading file: " + ex.Message);
    }
}
else
{
```

```
            Debug.LogWarning("File does not exist at: "
+ filePath);
        }
    }
```

3. Dan deleteFile untuk menghapus file atribut enemy.

```
    public void deleteFile()
    {
        File.Delete(filePath);
    }
```

Lalu ada *script ConsoleMechanic* untuk mendukung pengalaman *player* seperti ril berada di dalam linux. *Console Mechanic* mengimplementasikan mekanisme konsol dalam permainan untuk berinteraksi dengan *file* dan *objek* di dalam game. Berikut adalah penjelasannya:

1. readFile(string fileName): Membaca isi file yang disebutkan jika file ada di direktori yang tepat dan memberikan respons sesuai.

```
    void readFile(string fileName)
    {
        string filePath =
Path.Combine(gm.folderPath, fileName);
        GameObject fileObject =
GameObject.Find(Path.GetFileNameWithoutExte
nsion(fileName));

        if (fileObject != null)
        {
            float distance =
Vector3.Distance(playerTransform.position,
fileObject.transform.position);
            if (distance > 20.0f)
            {
                typerec.typingResponse("Access
denied. File is too far away.");
                return;
            }
        }
```

```
        if (File.Exists(filePath))
        {
            string fileContent =
File.ReadAllText(filePath);
            typerec.typingResponse(fileContent);
        }
        else
        {
            typerec.typingResponse("File does not
exist: " + fileName);
        }
    }
```

2. viewPath(): Menampilkan path direktori saat ini.

```
void viewPath()
    {
        typerec.typingResponse(gm.folderPath);
    }
```

3. ViewFolderContents(): Menampilkan isi dari folder saat ini, termasuk *file* dan direktori beserta atribut mereka.

```
void ViewFolderContents()
    {
        if (Directory.Exists(gm.folderPath))
        {
            string response = "";

            // Header
            response += "Permissions\tSize\tLast
Modified\t\tName\n";
            response += "----------------------------------
----------\n";

            // Files
            string[] files =
Directory.GetFiles(gm.folderPath);
            foreach (string file in files)
            {
                FileInfo fileInfo = new FileInfo(file);
```

```
                response += $"{GetFilePermissions(fileInfo.Attributes)}\t{fileInfo.Length,-10} bytes\t{fileInfo.LastWriteTime,20}\t{fileInfo.Name}\n";
            }

            // Directories
            string[] directories = Directory.GetDirectories(gm.folderPath);
            foreach (string directory in directories)
            {
                DirectoryInfo directoryInfo = new DirectoryInfo(directory);
                response += $"{GetDirectoryPermissions(directoryInfo.Attributes)}\t\t\t\t{directoryInfo.LastWriteTime,-20}\t{directoryInfo.Name}/\n";
            }

            typerec.typingResponse(response);
        }
        else
        {
            typerec.typingResponse("Directory does not exist: " + gm.folderPath);
        }
    }
```

4. changeMap(string level): Mengubah scene ke scene lain setelah menunjukkan animasi loading dengan sprite yang berbeda.

```
    public void changeMap(string level)
    {
        string sceneName = level;
        if (checkSceneName(sceneName))
        {
            if (loadingSprites.Length > 0)
            {
                StartCoroutine(ChangeImage());
```

```
            }

StartCoroutine(LoadSceneAfterDelay(sceneNam
e));
        }
        else
        {
            typerec.typingResponse("Directory    does
not exist: " + gm.folderPath + "\\" + sceneName);
        }
    }

    private IEnumerator ChangeImage()
    {
        canvasLoadingScreen.SetActive(true);
        float elapsedTime = 0f;

        while (elapsedTime < totalLoadingDuration)
        {
            yield                return                new
WaitForSeconds(imageChangeInterval);
            elapsedTime += imageChangeInterval;
            int                randomIndex                =
UnityEngine.Random.Range(0, 3);
            loadingImage.sprite                        =
loadingSprites[randomIndex];
        }
    }
```

```
    private IEnumerator LoadSceneAfterDelay(string scene)
    {
        yield return new WaitForSeconds(totalLoadingDuration);
        SceneManager.LoadScene(scene);
    }
```

5. deleteFile(string fileName): Menghapus *file* dari *folder*, dengan pengecekan terlebih dahulu apakah *file* tersebut dapat dihapus berdasarkan kondisi yang diatur. Berikut adalah kode yang sudah dipotong

```
if (canRemove)
    {
        // Check distance only if the object is deletable
        if (enemyObject != null)
        {
            float distance = Vector3.Distance(playerTransform.position, enemyObject.transform.position);
            if (distance > 20.0f)
            {
                typerec.typingResponse("Access denied. Object is too far away.");
                return;
            }
        }

       // If all conditions are met, proceed withdeletion
        try
```

```
        {
            if (File.Exists(fileToDelete))
            {
                File.Delete(fileToDelete);
                typerec.typingResponse(fileName + "
has been deleted successfully!");
            }
            else
            {
                typerec.typingResponse("File does not
exist: " + fileName);
            }
        }
        catch (System.Exception ex)
        {
            typerec.typingResponse("Error deleting
file: " + ex.Message);
        }
    }
    else
    {
        typerec.typingResponse("Cannot delete file: "
+ fileName + ". Permission denied.");
    }
```

6. changeStatCommand(): Mengubah nilai atribut dalam file berdasarkan perintah yang diberikan.
7. attackEnemyCommand(): Menyerang objek musuh dengan verifikasi kunci yang sesuai.
8. Dan terakhir adalah *script* untuk menangkap dan membagi inputan pengguna dari *terminal*

```
void HandleCommand(string mainCommand)
```

```
{
        switch (mainCommand)
        {
            case "cd":
                if (splitCommand.Length > 1)
                {
                    string fileName = splitCommand[1];
                    changeMap(fileName);
                }
                else
                {
                    typerec.typingResponse("Usage:   cd
<folder>");
                }
                break;
            case "cat":
                if (splitCommand.Length > 1)
                {
                    string fileName = splitCommand[1];
                    readFile(fileName);
                    print(fileName);
                }
                else
                {
                    typerec.typingResponse("Usage:  cat
<filename>");
                }
                break;
            case "rm":
                if (splitCommand.Length > 1)
                {
```

```
                string fileName = splitCommand[1];
                deleteFile(fileName);
                print(fileName);
            }
            else
            {
                typerec.typingResponse("Usage:   rm
<filename>");
            }
            break;
        case "whoami":
            typerec.typingResponse("root");
            break;
        case "ls":
            ViewFolderContents();
            break;
        case "pwd":
            viewPath();
            break;
        case "./changestat":
            changeStatCommand();
            break;
        case "./attackenemy":
            attackEnemyCommand();
            break;
        default:
            typerec.typingResponse("Command not
found!.");
            break;
    }
  }
```

C. Pembuatan Terminal Linux

Programmer juga menerapkan simulasi *terminal* linux yang bisa menangkap *command* yang diketik oleh *player* lalu dikelola dan kemudian dikembalikan kembali menjadi *response* dalam sebuah *console* di *terminal*. Berikut adalah penjelasan untuk metode dan fungsi utamanya:

1. Didalam *update*, ada pengelolaan input dari pengguna dan mengambil tindakan berdasarkan input tersebut.

```
void Update()
{
    if (Input.anyKeyDown)
    {
        inputKey = Input.inputString;

        if (inputKey.Length > 0 && inputKey[0] == '\b')
        {
            if (originalText.text.Length > 0)
            {
                originalText.text = originalText.text.Substring(0, originalText.text.Length - 1);
            }
        }
        else if (Input.GetKey(KeyCode.LeftControl) || Input.GetKey(KeyCode.RightControl))
        {
            if (Input.GetKeyDown(KeyCode.C))
            {
                originalText.text = "";
```

```
                responseText.text = "";
            }
        }
        else
        {
            originalText.text += inputKey;
        }
    }

    if (Input.GetKeyDown(KeyCode.Return) ||
Input.GetKeyDown(KeyCode.KeypadEnter))
    {
        responseAll = responseAll+"\nroot@linux:
"+originalText.text+"";
        cm.getCommand(originalText.text);
        originalText.text = "";
        Canvas.ForceUpdateCanvases();
        scrollRect.verticalNormalizedPosition = 0f;
    }
```

2. typingResponse(string res): Menampilkan *response* dari konsol dalam *responseText*.

```
public void typingResponse(string res)
  {
    Canvas.ForceUpdateCanvases();
    responseAll = responseAll+"\n"+res+"\n";
    responseText.text = responseAll;
    res = "";
    scrollRect.verticalNormalizedPosition = 0f;

  }
```

4. Testing

Tahap *testing* adalah tahap pengujian game untuk mendeteksi *bug* dan memastikan bahwa *game* berfungsi dengan baik. Pengujian dilakukan oleh internal *developer team* dan *beta tester*.

5. Beta

Tahap *beta* adalah tahap pengujian game oleh pengguna *eksternal*. Pada tahap ini, *game* diuji untuk mengetahui apakah *game* diterima oleh pengguna dan untuk mendeteksi *bug* yang tidak terdeteksi pada tahap *testing*.

6. Release

Tahap rilis adalah tahap di mana game dirilis ke publik. Pada tahap ini, *final build* dari game dirilis ke *platform* target.

B. Kebutuhan Perangkat

Berikut adalah rincian kebutuhan perangkat keras yang dibutuhkan *game programmer* dalam pembuatan game Kevin In Terminal Console:

1. Perangkat Keras

a. Processor : Intel I5-11400H
b. RAM : 16GB DDR4
c. GPU : RTX 3050

2. Perangkat Lunak

Berikut adalah perangkat lunak yang digunakan dalam proses pembuatan game Kevin In Terminal Consol sebagai berikut:

a. Unity 2021.3.25f1

Unity adalah *Game Engine* yang dapat digunakan untuk membuat berbagai jenis aplikasi interaktif, termasuk *game*, simulasi dan pengalaman imersif lainnya. Game yang dibuat di Unity dapat diterbitkan ke berbagai *platform*, termasuk PC, Mac, mobile (iOS dan

Android), konsol (Playstation dan Xbox) dan Web. Untuk bahasa yang digunakan dan direkomendasikan oleh Unity dalam pembuatan *Game* di Unity adalah C#.

b. Visual Studio Code

Visual Studio Code adalah software editor kode yang populer untuk berbagai jenis proyek, termasuk pengembangan *web*, pengembangan *game* dan pemrograman lainnya.

C. Target User

Target pengguna game Kevin in Terminal Console adalah kelompok usia 13 tahun keatas. Pengelompokkan usia ini berdasarkan Peraturan Menteri Kominfo Nomor 11 Tahun 2016 tentang Klasifikasi Permainan Interaktif Elektronik sekaligus pengenalan *Indonesia Game Rating System* (IGRS). Penulis mengambil *rating* umur pada BAB II pasal 7 Kominfo karena sesuai dengan kriteria yang ada pada game Kevin in Terminal Console.

D. Metode Pengumpulan Data

Metode pengumpulan data pada perhitungan data yang masuk mengunakan skala liker$\frac{Total\ Skor}{Y} \times 100$

Berikut merupakan beberpa pertanyaan pada kuisoner :

1. Apakah kamu sudah pernah menggunakan sistem operasi linux sebelumnya?
2. Bagaimana pendapatmu tentang tingkat kesulitan game ini?
3. Apa fitur favoritmu dalam game ini dan mengapa?
4. Bagaimana pengalaman kamu memainkan game simulasi linux ini? Apakah pengalaman seperti berada di linux sungguhan itu nyata?
5. Seberapa baik anda dalam memahami command linux yang ada didalam game ini

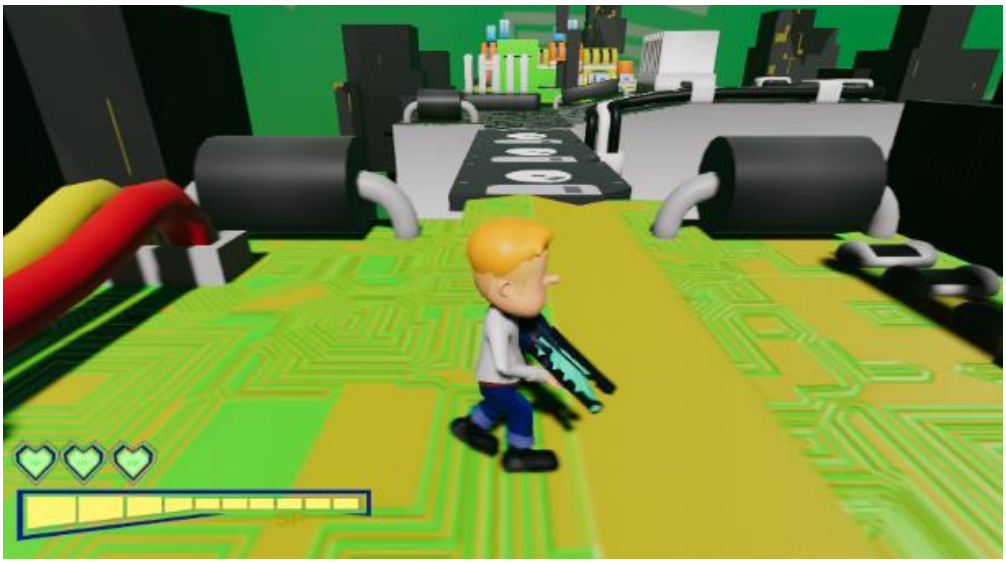
BAB IV
HASIL DAN PEMBAHASAN

A. Hasil Pengembangan Fitur Onion Design

Dalam Pengembangan fitur yang diatur berdasarkan prioritas di Onion Design, berikut adalah hasil pengembangan tersebut

1. Movement, Roll dan Jump

Fitur ini ter-deliver dengan baik sehingga player dapat melakukan movement, roll dan jump sesuai dengan keinginan dari Game Designer



Gambar 4.1 Movement

Gambar 4.2 Jump Player

Gambar 4.3 Roll Player

2. Create Environment File & Terminal

Fitur ini ter-*deliver* dengan baik, player sehingga pengalaman seperti di linux bisa dirasakan dengan baik karena beberapa bagian dari fitur yang dikembangkan in

Gambar 3. Terminal

Gambar 4.4 Folder and File Created

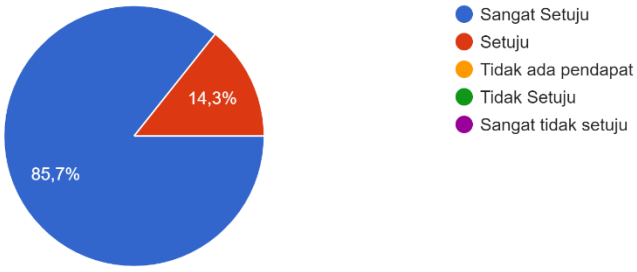
B. Hasil Pengujian User

Pengujian pada *user* bertujuan untuk mendapatkan informasi apakah gim Kevin in Terminal Console sudah berjalan dengan baik berdasarkan pertanyaaan yang disampaikan pada *user*.

Table 3. Daftar Pertanyaan dan penilaian

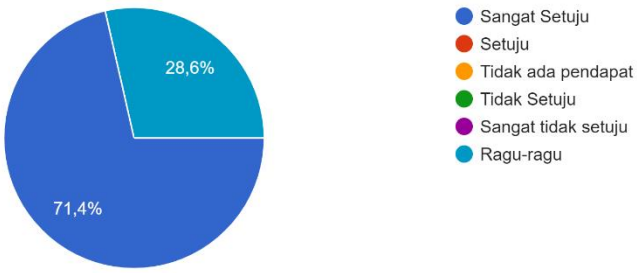
No.	Pertanyaan	Penilaian				
		Sangat Tidak Setuju	Tidak Setuju	Ragu-ragu	Setuju	Sangat Setuju
1.	Apakah mekanik didalam game ini seperti *movement*, *jump* dan *roll* sudah cukup baik?	0	0	0	1	6
2.	Apakah game ini sangat sulit?	0	0	2	0	5
3.	Apa fitur simulasi Linux ini tersampaikan dengan baik?	0	0	1	3	3
4.	Apakah pengalaman seperti berada di linux sungguhan itu nyata?	0	0	1	1	5
5.	Seberapa baik anda dalam memahami command linux yang ada didalam game ini?	0	0	2	2	3
Jumlah		0	0	4	7	22

Apakah meknaik didalam game ini seperti movement, jump dan roll sudah cukup baik?
7 jawaban



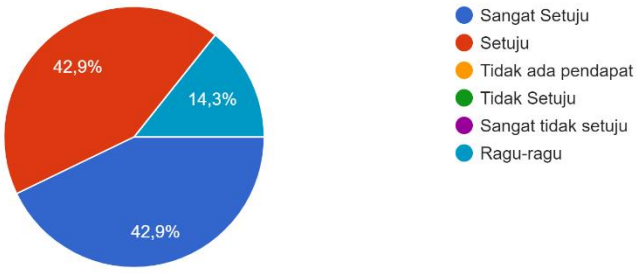
Gambar 4.5 Gambar Diagram 1

Apakah game ini sangat sulit?
7 jawaban

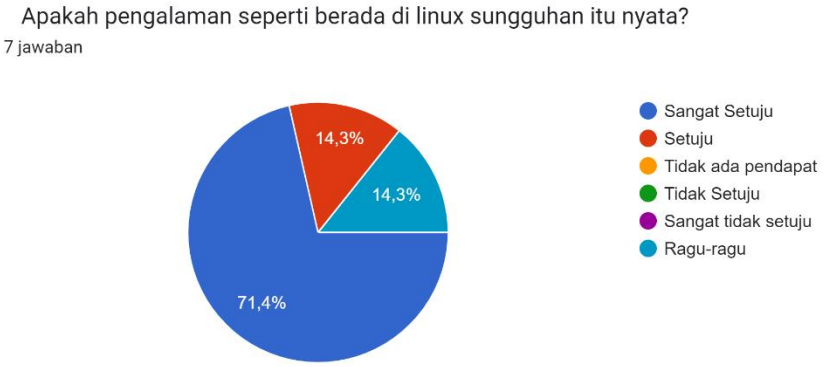


Gambar 4.6 Gambar Diagram 2

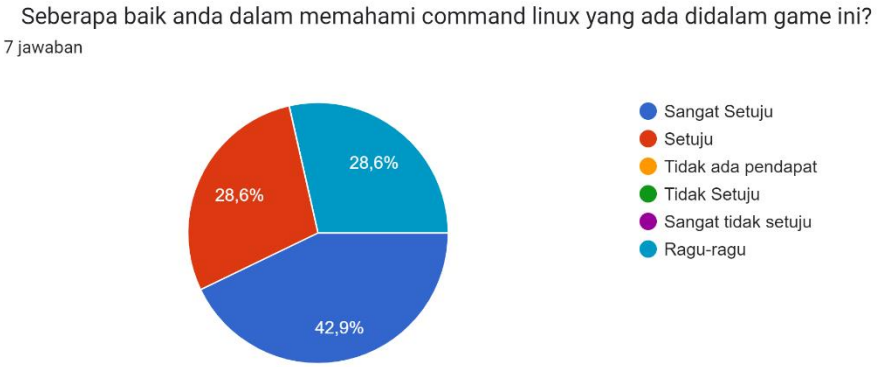
Apa fitur simulasi Linux ini tersampaikan dengan baik?
7 jawaban



Gambar 4.9 Gambar Diagram 3



Gambar 4.4 Gambar Diagram 4



Gambar 4.11 Gambar Diagram 5

Rumus Perhitungan untuk penilaian Skala likert mengunakan rumus.

$$\frac{\boldsymbol{Total\ Skor}}{\boldsymbol{Y}} \times \mathbf{100}$$

Total skor adalah total dari penilaian pada table dengan perhitungan TxPn yaitu T = Total Responden dan Pn = Penilaian angka Skor Likert, maka :

- Sangat Tidak setuju (1) T x Pn = 0 x 1 = 0,
- Tidak Setuju (2) T x Pn = 0 x 2 = 0,
- Ragu-ragu (3) T x Pn = 4 x 3 = 12,
- Setuju (4) T x Pn = 7 x 4 = 21,
- Sangat Setuju (5) T x Pn = 22 x 5 = 110
- Maka Total Skor = 143

Maka nilai Y adalah skor tertinggi dari nilai likert x jumlah penilaian responden, maka 5 x 33 = 165.

Penyelesaian akhir diambil dari responden yang menilai setuju, maka :

$$\frac{\boldsymbol{Total\ Skor}}{\boldsymbol{Y}} \times \mathbf{100} = \frac{\mathbf{150}}{\mathbf{165}} \times \mathbf{100} = \ \mathbf{0,909} \times \mathbf{100} = \mathbf{90.9}$$

Berdasarkan hasil dari penghitungan skala likert diatas, mendapat hasil 90,9% maka dengan ini berada dalam kategori setuju.

BAB V
PENUTUP

A. Kesimpulan

Berdasarkan analisis yang telah dilakukan, berikut adalah kesimpulan yang penulis berikan

1. Studi ini mengidentifikasi beberapa masalah, seperti keterbatasan command Linux, keterbatasan *class* dan *methode* dalam *System.IO*, serta perlunya *error exception* untuk menciptakan pengalaman pengguna yang mirip dengan terminal Linux sebenarnya.
2. Penggunaan *System.IO* sangat efisien dalam pengembangan *game* ini karena kelas-kelas yang berhubungan dengan data dan *file* didalam *System.IO* mempermudah dalam membuat simulasi *folder* atau direktori dan *file* di game ini. Diantaranya, ada kelas *File* untuk pengolahan data dan alirannya, *Path* untuk melakukan operasi pada strings yang mengandung path *file* atau direktori dan direktori untuk membuat, memindahkan dan mengenumerasi isi direktori dan subdirektori.
3. Berdasarkan pengujian beta, didapatkan bahwa 90% setuju bahwa *game* ini sangat sulit namun player menikmati *controller* yang ada seperti *movement*, *roll* dan *jumping* serta player merasakan pengalaman seperti memakai linux sungguhan dengan pemahaman perintah dasar di linux sesuai dengan yang ada di game. Berdasarkan pengujian juga ditemukan ada narasumber yang ragu-ragu untuk menjawab beberapa kuisioner karena dinyatakan tidak mengertinya atau awamnya orang tersebut tentang sistem operasi linux.

B. Saran

Saran untuk pengembang berikutnya yang akan mengembangkan game serupa, ada beberapa hal yang perlu diperhatikan, yaitu:

1. Memahami target audiens sehingga keseimbangan antar level sesuai dengan kemampuan dan kebutuhan mereka
2. Pastikan elemen edukatif terintegrasi dengan baik dalam gameplay.

3. Sediakan tutorial dan panduan yang jelas di awal permainan untuk membantu pemain memahami mekanik dasar dan tujuan game.

DAFTAR PUSTAKA

Alamsyah, M. R. (2022). LAPORAN TUGAS AKHIR PERMAINAN 3D BERBASIS DESKTOP BERGENRE HOROR "KRAMAT" (3D MODELER, TEXTURING, ENVIRONMENT), 10.

Desktop Windows Version Market Share Worldwide Mei 2024

Linus Torvalds. (2022, February 25). Linux. Retrieved February 25, 2023, from https://www.kernel.org/

Microsoft. (2023, January 25). System.IO. Retrieved January 25, 2023, from https://docs.microsoft.com/en-us/dotnet/api/system.io

Pressman, R. S. (2016). Software engineering: A practitioner's approach (8th ed.). New York, NY: McGraw-Hill Education.

Renna Yanwastika Ariyana & Erma Susanti & Muhammad Rizqy Ath-Thaariq & Riki Apriadi (2022) Penerapan Metode Game Devlopment Life Cycle (GDLC) pada Pengembangan Game Motif Batik Khas Yogyakarta

R. Supardi, "Pembuatan Game Balap Kelinci dengan Unity Berbasis Android," J. Ilm. Rekayasa dan Manaj. Sist. Inf., vol. 7, no. 1, pp. 19–26, 2021.

Unity. (2023, January 25). Unity. Retrieved January 25, 2023, from https://unity.com/

Yong Zhang & Li Guo & Li-Jun Ma (2018) A Computational Protocol to Analyze Metatranscriptomic Data Capturing Fungal–Host Interactions: Methods and Protocols

LAMPIRAN

Lampiran 1. Biodata Penulis

Nama	: Steven Julian
NIM	: 20210077
Program Studi	: Teknologi Permainan
Jenis Kelamin	: Laki-laki
Tempat, Tanggal Lahir	: Jakarta, 15 Juli 2002
Agama	: Kristen
Kewarganegaraan	: Indonesia
Alamat	: Jl. Tipar rt04/07 Pekayon. Pasar Rebo, DKI Jakarta
Nomor Handphone	: 089658610626
E-mail	: sn22shop@gmail.com

Lampiran 2. Lembar Bimbingan TA

	KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,RISET, DAN TEKNOLOGI POLITEKNIK NEGERI MEDIA KREATIF JURUSAN DESAIN	FormTA-05
	LEMBAR PEMBIMBINGAN TUGAS AKHIR	

Nama : Steven Julian

NIM : 20210077

Program Studi : Teknologi Permainan

Pembimbing I : Prily Fitria Aziz, M. Kom

Judul Proposal : Penerapan Mini Environment Linux Menggunakan System.IO Pada Game 3D Kevin in Terminal Console

No	Waktu	Uraian Bimbingan	Paraf Pembimbing
1.	19-may	Bimbingan Bab 1	
2.	22-may	Revisi Bab 1	
3.	6-Juni	Bimbingan Bab 2	
4.	20-Juni	Revisi Bab 2	
5.	23-Juni	Bimbingan karya	
6.	1-Juli	karya 20%	
7.	6-Juli	karya 80%	
8.	8-Juli	Karya 100%	

Pembimbingan minimal 8 (delapan) kali

Mengetahui

Koordinator Prodi,

Prily Fitria Aziz, M. Kom

NIP. 199104192019032015

Pembimbing I

Prily Fitria Aziz, M. Kom

NIP. 199104192019032015

	KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI POLITEKNIK NEGERI MEDIA KREATIF JURUSAN DESAIN	
	LEMBAR PEMBIMBINGAN TUGAS AKHIR	

Nama : Steven Julian
NIM : 20210077
Program Studi : D4 – Teknologi Permainan
Pembimbing II : Andrian, S.Kom, M.Kom
Judul Proposal : Penerapan Mini Environment Linux Menggunakan System.IO Pada Game 3D Kevin in Terminal Console

NO	Waktu	Uraian Bimbingan	Paraf pembimbing
1	29 April 2024	Revisi Bab I, Latar Belakang, Identifikasi masalah, rumusan masalah	
2	29 Mei 2024	Revisi Kembali BAB I, Latar Belakang	
3	30 Mei 2024	Revisi BAB I, Identifikasi Masalah dan Rumusan Masalah	
4	28 Juni 2024	ACC BAB I, Lanjutkan laporan BAB II dan III	
5	1 Juli 2024	Revisi BAB II dan BAB III	
6	3 Juli 2024	ACC BAB II, Revisi BAB III	
7	5 Juli 2024	Demo Game	
8	8 Juli 2024	ACC BAB III, Lanjutkan BAB IV dan V	
9	9 Juli 2024	ACC BAB IV dan V	
10	9 Juli 2024	Sidang	

Pembimbingan minimal 8 (delapan) kali.

Mengetahui
Koordinator Prodi.

Prily Fitria Aziz, M.Kom
NIP. 199104192019032015

Pembimbing II

Andrian, S.Kom, M.Kom.
NIP. 198611302020121004

Lampiran 3. Dokumentasi Foto Kegiatan