PENGEMBANGAN *GAME* 2D *PLATFORMER* "*VIRUS MUST DIE*" BERBASIS *ANDROID* MENGGUNAKAN *UNITY*

TUGAS AKHIR

Disusun Untuk Memenuhi Syarat Kelulusan Program Strata I pada
Sekolah Tinggi Manajemen Informatika dan Komputer
(STMIK) Palangkaraya

OLEH

DIMAS PRAYOGA
C1855201055
PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
(STMIK) PALANGKARAYA
2022

PENGEMBANGAN *GAME* 2D *PLATFORMER* "*VIRUS MUST DIE*" BERBASIS *ANDROID* MENGGUNAKAN *UNITY*

TUGAS AKHIR

Disusun Untuk Memenuhi Syarat Kelulusan Program Strata I pada
Sekolah Tinggi Manajemen Informatika dan Komputer
(STMIK) Palangkaraya

OLEH

DIMAS PRAYOGA
C1855201055
PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
(STMIK) PALANGKARAYA
2022

LEMBAR PERNYATAAN

Yang bertanda tangan di bawah ini:

Nama Mahasiswa : **DIMAS PRAYOGA**
N I M : **C1855201055**

menyatakan bahwa Tugas Akhir dengan judul :

PENGEMBANGAN *GAME* 2D *PLATFORMER* "*VIRUS MUST DIE*" BERBASIS *ANDROID* MENGGUNAKAN *UNITY*

adalah hasil karya saya dan bukan merupakan duplikasi sebagian atau seluruhnya dari karya orang lain, kecuali bagian yang sumber informasi dicantumkan.

Pernyataan ini dibuat dengan sebenar-benarnya secara sadar dan bertanggungjawab dan saya bersedia menerima sanksi pembatalan Tugas Akhir apabila terbukti melakukan duplikasi terhadap Tugas Akhir atau karya ilmiah lain yang sudah ada.

Palangka Raya, 19 Januari 2022

Yang Membuat Pernyataan,

DIMAS PRAYOGA

PERSETUJUAN

PENGEMBANGAN GAME 2D PLATFORMER "VIRUS MUST DIE" BERBASIS *ANDROID* MENGGUNAKAN UNITY

Tugas Akhir Ini Telah Disetujui Untuk Diujikan pada
Tanggal 19 Januari 2021

Pembimbing I,

Lili Rusdiana, M.Kom
NIK. 198707282011007

Pembimbing II,

Fenroy Yedithia, S.Kom, M.TI.
NIK. 199208112019102

Mengetahui
Ketua STMIK Palangkaraya,

Suparno, M.Kom.
NIK. 196901041995105

PENGESAHAN

PENGEMBANGAN GAME 2D PLATFORMER "VIRUS MUST DIE" BERBASIS *ANDROID* MENGGUNAKAN UNITY

Tugas Akhir Ini Telah Diujikan, Dinilai, dan Disahkan
Oleh Tim Penguji pada Tanggal 22 Januari 2022

Tim Penguji Tugas Akhir :

1. Veny Cahya Hardita, M.Kom
   Ketua

2. Elok Faiqotul Himmah, S.Si., M.Sc.
   Sekretaris

3. Sulistyowati, S.Kom., M.Cs.
   Anggota

4. Lili Rusdiana, M.Kom.
   Anggota

5. Fenroy Yedithia, S.Kom, M.TI.
   Anggota

MOTTO DAN PERSEMBAHAN

Kesuksesan didapat dari semangat dan optimisme dalam mencapai sesuatu, walaupun harus mengorbankan sedikit kesenangan dalam perjalanan mencapai nya. Percayalah, kesenangan yang didapat setelah mendapatkan kesuksesan merupakan kesenangan yang sesungguhnya.

Tugas Akhir ini kupersembahkan untuk

- *Kedua orang tua :*

  Ahmad Pahruka, S.Pd., M.MP.

  Berlian, S.Pd.

- *Saudari :*

  Umayroh Sabifa

- *Teman-teman :*

  Kelas B Teknik Informatika angkatan 2018 dan semua teman-teman lainnya yang telah mendukung , memberi bantuan dan saran.

INTISARI

Dimas Prayoga, C1855201055, 2022. *Pengembangan Game 2D Platformer "VIRUS MUST DIE" Berbasis Android Menggunakan Unity,* Pembimbing I Lili Rusdiana, M.Kom., M.Kom., Pembimbing II Fenroy Yedithia, S.Kom, M.TI.

Pengembangan *game* bergenre 2D *Platformer* di era sekarang menjadi tantangan tersendiri dalam dunia industri *game*, dengan tampilan grafis 2D yang di suguhkan tidak begitu banyak peminatnya dibandingkan dengan grafis tiga dimensi (3D) yang lebih realistik dan hidup dari segi desain grafis tetapi mekanik permainan yang di suguhkan *genre* 2D Platformer dapat bersaing dalam industri pengembangan *game*.

Mengembangkan *game 2D Platformer* Berbasis *Android* yang mengimplementasikan AI (*artificial intelligence*) pada objek tertentu dan menambahkan unsur edukasi didalam nya yang dapat membantu pengguna untuk memberikan hiburan serta pembelajaran secara bersamaan saat bermain *game*.

Metode yang digunakan dalam penelitian ini meliputi studi pustaka, observasi dan percobaan. Metode pengembangan perangkat lunak yang digunakan untuk alur pengerjaan penelitian ini menggunakan metode perangkat lunak SDLC (*System Development Life Circle)* dengan pendekatan pengembangan model air terjun (*Waterfall*). Perangkat lunakyang digunakan dalam pembuatan *Game 2D Platformer "Virus Must Die " Berbasis Android* adalah *Unity* versi 2020.3.1f1 (LTS).

Pada penelitian ini dilakukan pengujian *black box testing* atau pengujian kesalahan program dan hasilnya program dapat berjalan dengan baik. Berdasarkan hasil kuesioner yang telah dilakukan pada aplikasi *game 2D platformer Virus Must Die* Berbasis *Android* dengan 20 responden maka *game* ini dapat dikatakan layak digunakan dan mendapatkan hasil sangat baik dengan nilai persentase 82,4%.

Kata kunci : *Android*, AI, Edukasi, *Game*, *Unity, Platformer 2D*

ABSTRACT

The development of the 2D Platformer genre game in the current era is a challenge in the world of the game industry, with the 2D graphic display that is served is not so much demand compared to the more realistic and lively three-dimensional (3D) graphics in terms of graphic design but the game mechanics that are served 2D Platformer genre can compete in the game development industry.

Develop an Android-based 2D Platformer game that implements AI (artificial intelligence) on certain objects and adds an educational element in it that can help users to provide entertainment and learning simultaneously while playing games.

The method used in this research includes literature study, observation, and experiment. The software development method used for this research workflow uses the SDLC (System Development Life Circle) software method with a waterfall model development approach. The software used in making the 2D Platformer Game "Virus Must Die" Based on Android is Unity version 2020.3.1f1 (LTS).

In this study, black-box testing or program error testing was carried out and the results were that the program could run well. Based on the results of the questionnaire that has been conducted on the Android-Based 2D Virus Must Die platformer game application with 20 respondents, this game can be said to be feasible to use and get very good results with a percentage value of 82.4%.

Keywords : *Android*, Education, *Game*, *Unity, Platformer 2D*

KATA PENGANTAR

Puji dan syukur saya panjatkan kepada Allah Swt. atas ridanya saya dapat menyelesaikan penyusunan tugas akhir ini. Adapun judul skripsi yang saya ajukan adalah **"Pengembangan *Game 2D Platformer "VIRUS MUST DIE"* Berbasis *Android* Menggunakan *Unity*"**

Tugas akhir ini diajukan untuk memenuhi syarat kelulusan mata kuliah Tugas akhir di Program Studi Teknik Informatika STMIK Palangkaraya. Tidak dapat disangkal bahwa butuh usaha yang keras dalam penyelesaian pengerjaan tugas akhir ini. Namun, karya ini tidak akan selesai tanpa orang-orang tercinta di sekeliling saya yang mendukung dan membantu. Terima kasih saya sampaikan kepada:

1. Lili Rusdiana, M.Kom selaku Dosen pembimbing I yang banyak memberikan saran, koreksi dan bimbingan dalam menyelesaikan tugas akhir ini.
2. Fenroy Yedithia, S.Kom, M.TI. selaku Dosen pembimbing II yang telah memberikan bimbingan, arahan, pengetahuan, koreksi dan berbagai pengalaman kepada penulis.
3. Responden yang telah mengisi kuesioner untuk keperluan penyelesaian TA.

Semoga segala kebaikan dan pertolongan semuanya mendapat berkah dari Allah Swt. Dengan kerendahan hati mengharapkan saran dan kritik yang sifatnya membangun dari semua pihak demi membangun laporan penelitian ini.

Palangka Raya, 19 Januari 2022

Penulis

DAFTAR ISI

DAFTAR TABEL

-=

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

Latar Belakang Masalah

Kemajuan teknologi di era digital semakin signifikan, adanya persaingan antar individu untuk mengembangkan teknologi, termasuk bidang pengembangan *game. Game* dibuat sebagai permainan yang menyenangkan dengan unsur grafis gambar yang digerakan oleh pemain yang memainkannya dapat dirasakan langsung dan nyata. Ada banyak *genre game* yang sudah diciptakan sebelumnya seperti *genre First Person Shooter (FPS), Role Playing game (RPG), Arcade* dan salah satunya *Platformer* sub *genre* permainan dengan grafis dua dimensi (2D) pemain dapat menggerakkan karakter di dalam *game* tersebut untuk melewati rintangan dan dapat melawan musuh untuk mencapai suatu tujuan.

Pengembangan *game* bergenre 2D *Platformer* di era sekarang menjadi tantangan tersendiri dalam dunia industri pengembangan *game* dikarenakan *game* dengan grafis 2D sendiri tidak begitu banyak peminatnya dibandingkan dengan grafis tiga dimensi (3D) yang lebih realistik dan hidup dari segi desain grafis tetapi mekanik permainan yang di suguhkan *genre* 2D Platformer dapat bersaing dalam industri pengembangan *game*. Dengan adanya teknologi pada zaman sekarang *game* dapat dimainkan di *smartphone* seperti *Android* dan *IOS* yang digunakan sebagai *platformer* untuk menjalankan *game* yang ada di *mobile/handphone*. Sehingga *game* dapat dimainkan dimanapun dan kapanpun tanpa harus menggunakan

Personal Computer (PC). Menggabungkan unsur *genre* permainan *platformer* dan juga ditambah sedikit unsur *genre* edukasi dapat dikembangkan menjadi *gameplay* yang menarik dan melatih kemampuan motorik pemain karena menggunakan jari sebagai *control* karakter di dalamnya serta melatih pola pikir pemain dengan ada nya pertanyaan yang disuguhkan dari segi edukasi.

Berdasarkan uraian di atas, peneliti ingin melakukan pengembangan *game* 2D *Platformer* berbasis *Android* dengan menambahkan unsur *genre* edukasi yang akan dibuat dengan menggunakan Unity sebagai *Software* pengembang *game* dengan menggunakan Bahasa pemrograman C#. Peneliti menggunakan metode *Finite State Machine* dalam implementasi *kecerdasan buatan* atau non player object didalamnya yang dapat membuat tindakan dinamis seperti *patroling, idle,* dan *attack* dengan pembangunan aplikasi dengan menggunakan metode *Waterfall approach versi pressman.*

Perumusan Masalah

Berdasarkan latar belakang yang telah diuraikan diatas, maka rumusan yang dapat diangkat sebagai berikut :"Bagaimana membangun aplikasi *game* 2D *Platformer* berbasis *Android* yang memiliki unsur edukasi ?".

Batasan Masalah

Adanya aspek-aspek didalam *game* yang akan dibangun maka diperlukan batasan masalah yang jelas untuk menghindari ketidakpahaman

dan kerancuan dalam pembahasan. Adapun batasan masalahnya sebagai berikut :

a) Aplikasi dijalankan pada *platform Android* dengan spesifikasi minimal sistem operasi *Android* 5.0 *"Lollipop"*dan maksimal *Android* 11.0

b) *Game* dimainkan secara *single player*

c) Bahasa pemrograman yang digunakan adalah C#.

d) Pembangunan *game* 2D *Platformer side-scrolling* dengan rintangan dan halangan didalamnya.

e) Mengimplementasikan unsur edukasi kedalam *game* 2D yang bergenre *platformer*.

f) Menggunakan metode *Finite State Machine* dalam implementasi kecerdasan buatan kepada AI di *game* tersebut.

Tujuan dan Manfaat

a) Tujuan

Tujuan penelitian ini, mampu membangun aplikasi *game* yaitu *2D Platformer "*Virus Must Die" menggunakan Unity berbasis *Android* dengan menggunakan Bahasa Pemrograman C# yang mengimplementasikan metode penelitian *Finite State Machine* kepada *AI* dan juga memberikan unsur edukasi di dalam *game.*

b) Manfaat

Manfaat dari penelitian tugas akhir adalah sebagai berikut :

1) Penulis dan pembaca mengetahui script yang dapat membentuk sistem *game* yang baik dengan Unity.

2) Penulis dan pembaca mengetahui cara menulis, membaca, dan mengelola data-data dalam *game* yang dibuat.

3) Penulis dan pembaca mengetahui mekanik *game* yang dibuat dalam penelitian ini.

4) Dapat memahami dan mengerti metode *Finite State Machine* yang digunakan dalam implementasi AI di *game*.

5) Dapat memahami dan mengetahui pengimplementasian unsur edukasi ke dalam *game 2D* bergenre *platformer*.

BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Dalam melakukan suatu penelitian diperlukan dukungan dari hasil beberapa penelitian yang relevan yang sebelumnya telah ada dan berkatian dengan penelitian yang sedang diteliti. Penulis akan menjelaskan serta menguraikan secara singkat beberapa penelitian yang relevan yang memiliki topik serupa atau mendekati persamaan yang digunakan sebagai acuan penelitian, Setelah penulis melakukan telaah terhadap beberapa penelitian, ada beberapa yang memiliki keterkaikatan dengan yang penulis lakukan, ditunjukan pada Tabel 1.

Tabel 1. Tinjauan Pustaka

No	**Penulis/ Tahun**	**Topik Penelitian**	**Metode**	**Pembahasan**	**Hasil**
1	Ade Mastya Pratama, dkk/ 2021	Perancangan permaianan *Platformer* "Ninja Runner" berbasis desktop dengan *Tools Unity 2D game engine*	*FSM (Finite State Machine), SUS (System Usability Scale)*	Penelitian ini bertujuan untuk melakukan perancangan *game platformer 2D* dengan menggunakan metode *FSM (Finite State Machine).*	*Game* yang dibuat dapat dimainkan dengan desktop sebagai *platform* tempat memainkan nya dengan tipe *game single player* atau hanya dapat dimainkan satu orang.

No	Penulis/ Tahun	Topik Penelitian	Metode	Pembahasan	Hasil
2	Dwi Rahmad Yanuar Rizki RT, dkk/2018	Game alam *the adventure* berbasis *android* menggunakan metode *Finite State Machine*	*FSM (Finite State Machine), SLDC (System Development Life Cycle)*	Penelitian ini dibuat untuk pengembangan *game* yang menghasilkan media *game* berbasis hiburan dan edukasi bencana yang berbasis *android* dengan. Juga mengimplementasikan metode FSM kedalam *game*.	*Game* yang dibuat dapat dimainkan di *platform android.* Unsur edukasi yang kuat pada materi bencana lingkungan juga menjadikan Game Alam The Adventure dapat dipakai oleh pengguna mulai dari SD kelas 6 dan tidak menutup kemungkinan digunakan khalayak umum.
3	Toni Ardyanto, dkk/ 2017	Pembuatan *game* 2D pertualangan Hanoman berbasis *Android*	Pengembangan sistem GDLC (*game Developer Life Cycle)*	Untuk mengembangkan sebuah *game* berbasis *Android* yang di kembangkan dengan metode GDLC (*game Developer Life Cycle).*	*Game* yang dibuat dapat memberikan kesan *gameplay* yang dapat di ingat oleh pemain karena memiliki nilai budaya dan edukasi di dalam nya.

2.2 Kajian Teori

a. G*ame*

Game berasal dari bahasa Inggris. Dalam kamus bahasa Indonesia istilah *game* berarti permainan. Permainan adalah sebuah sistem pemain terlibat dalam konflik buatan. Pemain berinteraksi dengan sistem dan konflik dalam permainan. Dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan (Zamroni, et al., 2013).

Jika dilihat dari grafis yang digunakan, *game* dapat digolongkan menjadi dua jenis, yaitu 2D dan 3D. Sementara jika dilihat dari cara memainkannya *game* memiliki beberapa genre di antaranya: *First Person Shooter*, *Role Playing game*, *Arcade*, *Simulation*, *Racing*, dan sebagainya (Wafda, 2015) .

b. **Sistem Operasi *Android***

Android adalah *Software* untuk perangkat mobile yang mencakup sistem operasi, middleware dan aplikasi kunci. Pengembangan aplikasi pada platform *Android* menggunakan bahasa pemrograman Java. Serangkaian aplikasi inti *Android* antara lain klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Dengan menyediakan sebuah *platform* pengembangan yang terbuka, pengembang *Android* menawarkan kemampuan untuk membangun aplikasi yang sangat kaya dan inovatif. Pengembang bebas untuk mengambil keuntungan dari perangkat keras, akses informasi lokasi, menjalankan *background services*, mengatur alarm, tambahkan pemberitahuan ke status bar, dan banyak lagi.

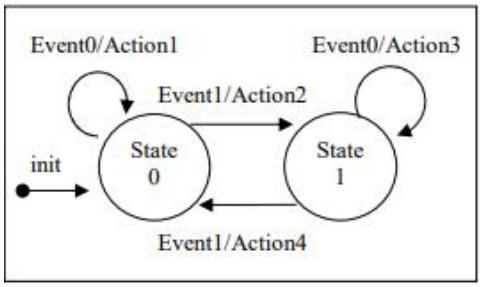
Android bergantung pada versi Linux 2.6 untuk layanan sistem inti seperti keamanan, manajemen memori, manajemen proses, network stack, dan model *driver*. Kernel juga bertindak sebagai lapisan abstraksi antara hardware dan seluruh *Software* stack (Fadjar Efendy Rasjid, 2010).

c. ***Finite State Machine***

Finite State Machines (FSM) adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan *state* (Keadaan), *event* (kejadian) dan *action* (aksi) (Millington & Funge, 2009).

Sistem dapat beralih atau bertransisi ke state lain jika mendapatkan masukan atau event tertentu, baik yang berasal dari perangkat luar atau komponen dalam sistemnya itu sendiri. Perpindahan keadaan ini umumnya juga disertai oleh aksi yang dilakukan oleh sistem ketika menanggapi masukan yang terjadi. Aksi yang dilakukan tersebut dapat berupa aksi yang sederhana atau melibatkan rangkaian proses yang relatif kompleks (Setiawan, 2006).

FSM dengan dua buah state dan dua buah input serta empat buah output yang berbeda seperti terlihat pada gambar, ketika sistem mulai dihidupkan, sistem akan bertransisi menuju State0, pada keadaan ini sistem akan menghasilkan Action1 jika terjadi masukan Event0, sedangkan jika terjadi Event1 maka Action2 akan dieksekusi kemudian sistem selanjutnya bertransisi ke keadaan State1 dan seterusnya. Ditunjukan pada Gambars 1.

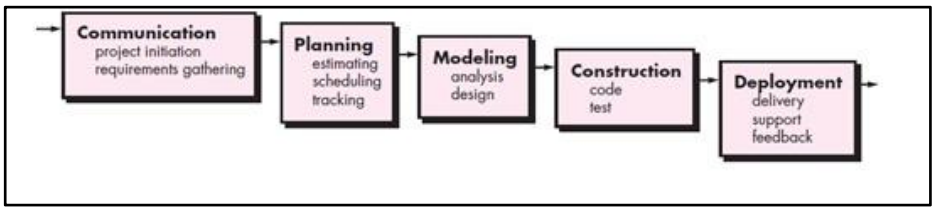


Gambar 1. Diagram state sederhana
Sumber: Setiawan, 2006

d. ***Waterfall Approach***

Metode *Waterfall* merupakan pendekatan SDLC (*System Development Life Circle)* paling awal yang digunakan untuk pengembangan perangkat lunak. Urutan dalam Metode *Waterfall* bersifat serial yang dimulai dari proses perencanaan, analisa, desain, dan implementasi pada sistem.

Metode ini dilakukan dengan pendekatan yang sistematis, mulai dari tahap kebutuhan sistem lalu menuju ke tahap analisis, desain, coding, testing/verification, dan maintenance. Langkah demi langkah yang dilalui harus diselesaikan satu per satu (tidak dapat meloncat ke tahap berikutnya) dan berjalan secara berurutan. Ditunjukan pada Gambar 2.



Gambar 2. *Waterfall* Pressman
Sumber: Pressman, 2015

Ada lima tahapan dalam *Waterfall versi* Pressman yaitu *Communication, Planning, Modeling, Construction* dan *Deployment* (Pressman, 2015).[i]

1) *Communication (Project Initiation & Requirements Gathering)* , membicarakan dan mencari permasalahan yang dihadapi dalam proses pengembangan plojek nanti.

2) *Planning* (*Estimating, Scheduling, Tracking),* berbicara tentang entimasi waktu pengerjaan, penjadwalan kerja yang dilaksanakan, dan *tracking* proses pengerjaan sistem.

3) *Modeling (Analysis & Design),* tahap perancangan dan arsitektur dari *game* yang dikerjakan seperti UI, karakter, lingkungan, dan *assets-assets* lain yang berhubungan dengan projek yang dikerjakan.

4) *Construction (Code & Test),* proses penerjemahan bentuk desain menjadi kode atau bahasa yang dimengerti oleh mesin atau masuk ke tahap *scripting*, setelah itu masuk ke tahap pengujian untuk mengetahui kesalahan yang mungkin terjadi untuk nantinya diperbaiki.

5) *Deployment (Delivery, Support, Feedback*), bertujuan untuk menyebarkan *game* yang telah di buat oleh pengembang. Kemudian dilakukan nya pemeliharaan, perbaikan, evaluasi dan pengembangan berdasarkan *feedback* yang diberikan agar pengembangan *game*

tetap berjalan dan sesuai dengan keinginan pengembang juga pemain.

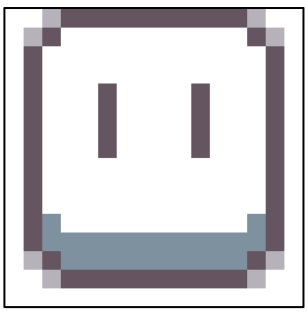
e. Unity

Gambar 3. Icon Unity

Unity merupakan suatu aplikasi yang digunakan untuk mengembangkan *game* multi platform yang didesain untuk mudah digunakan. Unity penuh perpaduan dengan aplikasi yang profesional. Editor pada Unity dibuat degan user interface yang sederhana (Horachek, 2014).

Unity digunakan pada iPhone, iPod dan iPad operating system yang mana iOS ada sebagai add-ons pada Unity editor yang telah ada lisensinya, dengan cara yang sama pada *Android* (Herman, 2017).

f. Aseprite



Gambar 4. Icon Aseprite

Aseprite adalah alat favorit dari banyak seniman piksel, dan dengan berbagai alat khusus piksel pikselnya, itu untuk alasan yang bagus. Segala sesuatu tentang Aseprite adalah 100% terfokus pada piksel - bahkan UI itu sendiri dilakukan dalam gaya pixel art.

Dimana Aseprite bersinar adalah sebagai alat untuk pembuatan dan animasi sprite. Ini memiliki lapisan yang kuat dan sistem animasi berbasis frame dengan beberapa fitur yang tidak hadir dalam banyak aplikasi seni pixel terfokus (Kezz Bracey, 2017).

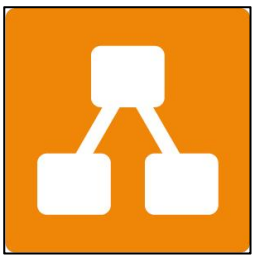
g. Visual Studio Code

Gambar 5. Icon Visual Studio Code

Visual Studio Code adalah editor *source code* yang dikembangkan oleh Microsoft untuk Windows, Linux dan MacOS. Ini termasuk dukungan untuk *debugging, GIT Control* yang disematkan, penyorotan sintaks, penyelesaian kode cerdas, cuplikan, dan kode *refactoring.* Hal ini juga dapat disesuaikan, sehingga pengguna dapat mengubah tema editor, *shortcut keyboard*, dan preferensi. Visual Studio Code bisa di *download* gratis dan *open-source*, meskipun unduhan resmi berada di bawah lisensi *proprietary*.

Visual Studio Code adalah kode editor sumber yang dikembangkan oleh Microsoft untuk Windows, Linux dan macOS. Ini termasuk dukungan untuk debugging, kontrol git yang tertanam dan GitHub, penyorotan sintaksis, penyelesaian kode cerdas, snippet, dan refactoring kode (Edy Winarno & Ali Zaki, 2014).

h. Draw.io



Gambar 6. Icon Draw.io

Draw.io merupakan *Software* diagram gratis untuk membuat flowchart, process diagrams, org charts, UML, ER dan network diagrams. Apliaksi ini berfokus pada pembuatan alur perancangan pada suatu sistem untuk menjadi *Blueprint* dari penelitian yang akan dibuat (Raska Lathif, 2019)

i. *Android* SDK *(Software Development Kit)*

Android Software Development Kit (SDK) merupakan kit yang bisa digunakan oleh para developer untuk mengembangkan aplikasi berbasis *Android*. Di dalamnya, terdapat beberapa tools seperti *debugger*, *software libraries*, *emulator*, dokumentasi, *sample code* dan *tutorial*.

Java SE Development kit adalah salah satu contoh *Android* SDK dan menjadi bahasa pemrograman yang paling sering digunakan untuk mengembangkan aplikasi Android (Dimas Catur Wibowo, 2019)

j. *Software* Dalam Pembuatan *Game*

Untuk mendukung pengembangan *game* dalam penelitian ini, adapun *Software* pendukung dalam pengembangan *game*. Dapat dilihat pada tabel 2.

Tabel 2. *Software* yang digunakan

No	Perangkat Lunak	Keterangan
1	Unity	Aplikasi pengembangan yang di khususkan dalam pembuatan *game*
2	Aseprite	alat untuk pembuatan dan animasi sprite. Di fokus kan ke arah *pixel art style*
3	Visual Basic Studio	Digunakan untuk pembuatan *script* dan editor *source code*
4	Draw.IO	Digunakan dalam pembuatan *flowchart, diagram, desain UI* dan membantu proses perancangan penelitian.
5	*Android* SDK	Digunakan untuk mengembangkan aplikasi untuk platform *Android*
6.	*Free License Assets*	Penggunaan *assets* gratis dalam pengembangan *game*

BAB III

METODE PENELITIAN

3.1 Perencanaan Alat dan bahan

Perencanaan alat dan bahan dibutuhkan untuk menunjang penyelesaian projek yang akan dibuat. Dari *hardware* yang digunakan dalam proses pembuatan dan *Software* yang digunakan dalam pembuatan *assets/*objek yang digunakan dalam pembuatan projek yang ingin dibuat. Penentuan perangkat didasarkan pada hasil analisis kebutuhan sistem yang telah dilakukan sebelumnya.

Perangkat lunak/*Software* yang digunakan adalah :

a. Sistem Operasi *Windows* 10
b. Aseprite, untuk membuat model dan gambar dalam bentuk *pixel*.
c. *Visual Studio Code*, untuk mengedit *source code* pada pengembangan projek.
d. Sistem Operasi *Android* 11
e. Unity, merupakan aplikasi untuk membuat *game.*

Perangkat keras/*hardware* yang digunakan adalah :

Komputer dengan spesifikasi :

a. Processor INTEL CORE i5-9400f 2.9Ghz
b. RAM 16GB
c. Kartu Grafis GEFORCE GTX 1650 *super* RAM 4GB
d. NVMe SSD 500GB

e. *Hard Disk* 500GB

f. Layar monitor 1920 x 1080 60Hz

g. *Keyboard* dan *mouse*

h. Pentab

3.2 Jenis Penelitian

Jenis penelitian yang digunakan dalam pengembangan *game* 2D ini merupakan salah satu pendekatan SDLC paling awal yang digunakan untuk pengembangan perangkat lunak, yaitu *Waterfall Approach versi Pressman.* Metode ini dilakukan dengan pendekatan yang sistematis, ada lima tahap yaitu :

a. *Communication (Project Initiation & Requirements Gathering),* berbicara tentang permasalahan yang akan dibangun pada penelitian ini, menyangkut *game* seperti apa yang akan dibangun, mencari latar belakang masalah dari *game* yang akan dibuat, jalannya permainan, unsur edukasi yang akan ditambahkan ke dalam *game* yang bertemakan *platformer* dan desain *asssets* pada *game.*

b. *Planning* (*Estimating, Scheduling, Tracking),* membuat jadwal kerja untuk projek.

c. *Modeling (Analysis & Design),* merealisasikan rancangan yang dibuat dan menjadikannya *assets* untuk digunakan pada *game,* .

d. *Construction (Code & Test),* merancang *script* yang digunakan untuk pengembangan *game*, dari *control player,enemy, menu* dan elemen-elemen lain yang menunjang keberhasilan pembuatan *game*.

e. *Deployment (Delivery, Support, Feedback*), tahap akhir yaitu uji coba setelah *game* selesai dibuat dan menerima *feedback* dari yang mencoba *game* yang telah dibuat.

3.3 Teknik Pengumpulan Data

Pengumpulan data dilakukan untuk keperluan penelitian. Tahap ini meliputi pengumpulan data dari buku-buku referensi yang relevan dengan pengembangan *game* 2D *platformer "Virus Must Die",* yaitu dengan mengumpulkan referensi-referensi yang berhubungan dengan penelitian dan mendukung dalam kebutuhan sistem yang akan dibuat. Referensi dapat diperoleh dari buku maupun artikel *online*.

3.4 Analisis Kebutuhan

Berdasarkan observasi yang dilakukan, analisis kebutuhan dalam pengembangan *game* yaitu data, proses, kelemahan.

a. Analisis Data

Berdasarkan observasi yang dilakukan oleh penulis terhadap penelitian ini, data-data yang dibutuhkan dalam pengembangan *game* 2D *platformer* adalah berupa data referensi yang bisa dijadikan atau diolah menjadi objek ataupun *assets* ke dalam pengembangan *game* 2D *platformer* yang dibuat.

Data-data referensi bisa dicari melalui website atau di kehidupan nyata, contoh nya seperti bentuk bangunan, tanaman, lingkungan, hewan(darat/laut), manusia dan lain-lain yang bisa dijadikan data sebagai komponen dalam pengembangan *game* 2D *platformer.*

b. Analisis Proses

Pada pengembangan *game* 2D *platformer* ini, metode yang digunakan dalam membangun nya adalah SLDC (*System Development Life Cycle) Waterfall Approach.* Metode ini dilakukan dengan pendekatan yang sistematis, mulai dari tahap kebutuhan sistem lalu menuju ke tahap analisis, desain, coding, testing/verification, dan *maintenance*. Langkah demi langkah yang dilalui harus diselesaikan satu per satu (tidak dapat meloncat ke tahap berikutnya) dan berjalan secara berurutan, oleh karena itu di sebut *Waterfall* (Air Terjun). Tidak hanya digunakan dalam pengembangan perangkat lunak (*software),* metode ini juga dapat di implementasikan ke pengembangan *game* juga, karena memiliki kesamaan pola dalam cara pembuatan nya. Ditunjukan pada Gambar 2.

c. Analisis Kelemahan

Ditemukan beberapa kelemahan dalam proses pengembangan *game* 2D yang akan dibuat, yaitu :

1) Dari segi konsep memiliki kesamaan dengan kebanyakan orang yaitu genre *platform*, yang membedakan nya adalah di *storyboard*, karakter, lingkungan, dan mekanik dari *gameplay*.

2) Kurangnya unsur edukasi dalam *game sub-genre action, platform.*

3) Beberapa *assets* yang perlu dibuat sendiri, kemungkinan bisa memakan waktu dalam proses pembuatan nya

3.5 Desain *game*

Dalam perencanaan desain *game* meliputi bebrapa hal yaitu gambaran umum dari *game* "Virus Must Die", rancangan umum, desain *user interface game*, serta *scenario game.*

a. Gambaran Umum

Gambaran umum *game "Virus Must Die" adalah* g*ame* bergenre *platformer* yaitu sub genre dari *game action* dengan tampilan *Side-scrolling* 2D berbasis *Android*. Sama hal nya dengan *gameplay game platformer* kebanyakan, yang membedakan disini adalah tujuan, misi, karakter, lingkungan, dan mekanik nya.

b. Rancangan Umum

Rancangan umum dibuat untuk menjadi patokan berjalan nya proses pengembangan *game.* Adapun rancangan umum nya yaitu :

1) *Game* dimainkan oleh satu orang atau biasa disebut dengan *single player.*

2) Karakter dapat bergerak kekiri dan kekanan hingga melompat sesuai arah *control* pemain.

3) Virus (musuh) dapat menyerang pemain.

4) Karakter pemain memiliki 5 nyawa untuk awal permainan.

5) Nyawa berkurang 1 setiap terkena Virus (musuh).

6) Karakter dapat menembak untuk melindungi diri dari Virus (musuh).

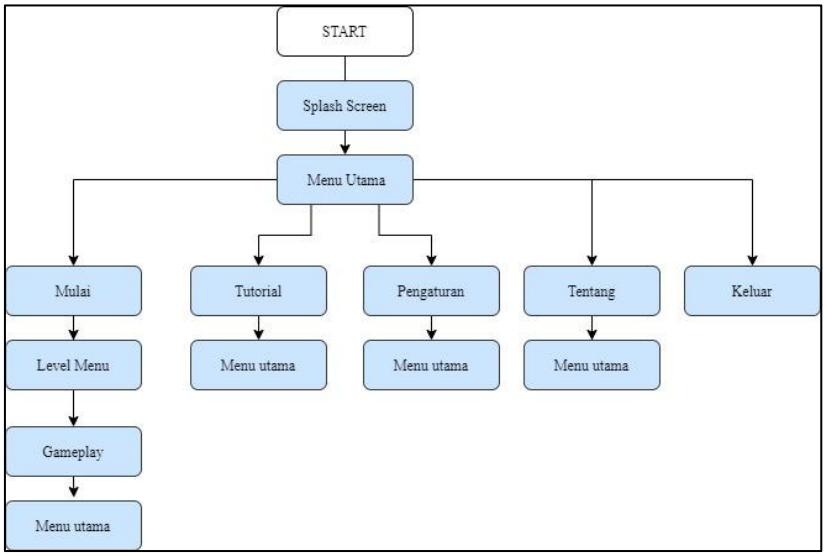
7) Memiliki beberapa rintangan yang menghadang pemain.

8) Dalam permainan ini tidak memiliki batas waktu yang telah ditentukan.

9) Tujuan dari *game* ini adalah untuk mengantarkan Obat (item) ke tujuan yang sudah di tentukan.

c. Perancangan Diagram HIPO (Hierarki Input Proses Output)

Proses pengembangan dan desain game ini menggunakan diagram HIPO atau biasa disebut Hierarki Input Proses Output. Proses ini dilakukan dalam proses pencarian informasi secara manual yang akan digunakan oleh user kemudian ditampilkan oleh sistem. Sedangkan user yang dimasksud ini adalah pengguna game yang merupakan masyarakat umum atau siapapun yang ingin memainkan game ini. Diagram HIPO yang menunjukan menu apa saja yang dipanggil. Perancangan diagram HIPO dapat dilihat pada Gambar 7.



Gambar 7. Diagram HIPO

Setelah dibuat nya rancangan sistem UI menggunakan diagram HIPO, setiap bagan pada diagram mempunyai fungsi yang berbeda-beda berdasarkan skenario atau tindakan yang berlangsung setelah memilih bagan tersebut. Adapun skenario pada sistem UI, yaitu :

1) Skenario 1.0 *Splash Screen*

Merupakan layar pembuka yang menampilkan logo dan logo buatan sendiri.

2) Skenario 2.0 Menu Utama

Merupakan awal tampilan game setelah *splash screen,* yang berisi tombol-tombol untuk navigasi pada menu utama.

a) Skenario 2.0 *Play*/Mulai

Merupakan tombol untuk berpindah *scene* ke *scene* menu level.

(1) Menu Level

Pada tombol menu level, tersedia beberapa tombol untuk memilih level yang ingin dimainkan dan tombol kembali untuk berpindah ke *scene* menu utama.

(2) *Gameplay*

Pada tombol *gameplay*, menampilkan permainan *game* "*virus must die"* sesuai level yang dipilih di menu level sebelumnya.

b) Skenario 3.0 Tutorial

Pada tombol tutorial menampilkan penjelasan tentang *game* yang dimainkan, objektif, dan mekanik dari *game* itu sendiri dan tombol kembali untuk berpindah ke menu utama.

c) Skenario 4.0 *Option/*Pengaturan

Pada tombol *option*/pengaturan menampilkan slider interaktif yang digunakan untuk mengatur suara keseluruhan game, dan tombol kembali untuk berpindah *scene* ke menu utama.

d) Skenario 5.0 *About/*Tentang

Pada tombol *about/*tentang menampilkan informasi dari pengembang *game "virus must die"* dan tombol kembali untuk kembali ke *scene* menu utama.

e) Skenario 6.0 Exit/Keluar

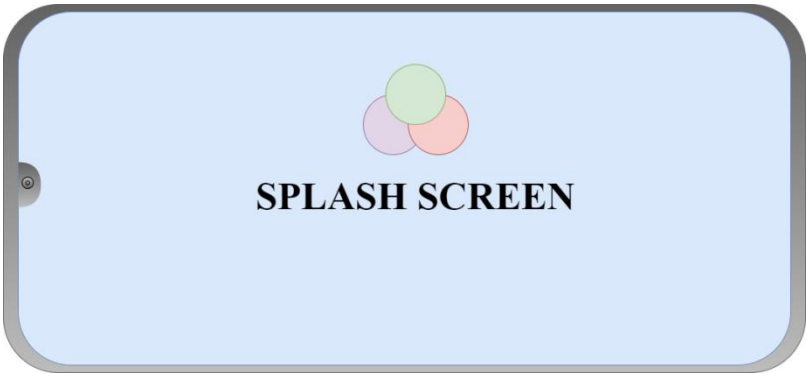
Merupakan tombol untuk keluar dari *game* atau menutup *game* tersebut.

d. Desain Antarmuka (*User Interface*)

Perancangan antarmuka atau biasa kita sebut dengan *user interface*, digunakan untuk memudahkan dalam mengimplementasikan perangkat lunak yang akan dibangun. Antarmuka ini juga berfungsi sebagai jembatan interaksi antara pemain dengan *game.* Perancangan antarmuka yang baik adalah dengan mengatur letak menu dan tombol yang ada dalam *game* dan mengatur letak halaman yang menapilkan isi dari sistem. Berikut perancangan antarmuka dari *game "Virus Must Die"* berbasis *Android.*

1) Antarmuka *Splash Screen*

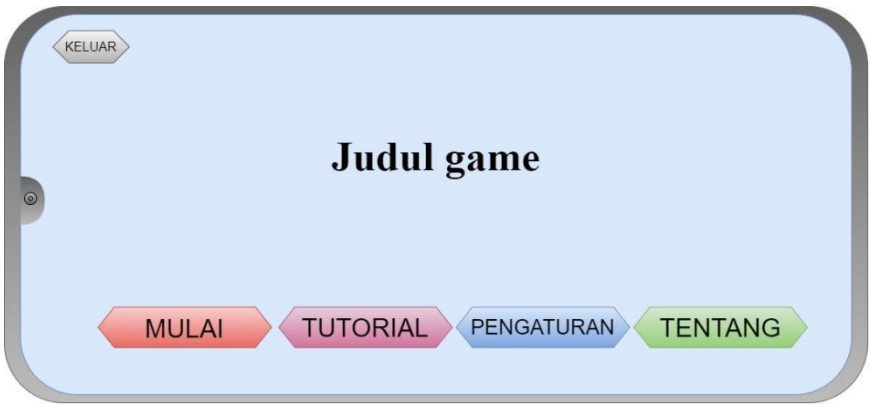
Tampilan *Splash Screen* merupakan tampilan yang pertama kali muncul ketika *game* dijalankan. Halaman ini sebagai pembuka agar *game* terlihat lebih menarik dapat dilihat pada Gambar 8 .



Gambar 8. UI Splash Screens

2) Antarmuka Menu Utama

Menu utama merupakan tampilan awal dari *game* setelah tampilan *splash screen*. Ada beberapa tombol yang dapat di akses dengan fungsi yang berbeda-beda. Ada tombol mulai yaitu fungsi nya untuk memulai game dengan berpindah dari *scene menu* ke *scene gameplay*. Tombol Tutorial fungsi nya untuk pindah ke *scene* tutorial dimana isi nya menjelaskan tentang cara bermain *game* tersebut seperti apa. Kemudian tombol pengaturan fungsinya untuk berpindah ke *scene settings* dan terakhir tombol Keluar, fungsinya untuk keluar dari *game*. Antarmuka Menu Utama dapat dilihat pada Gambar 9 .



Gambar 9. UI Menu Utama

3) Antarmuka Tutorial

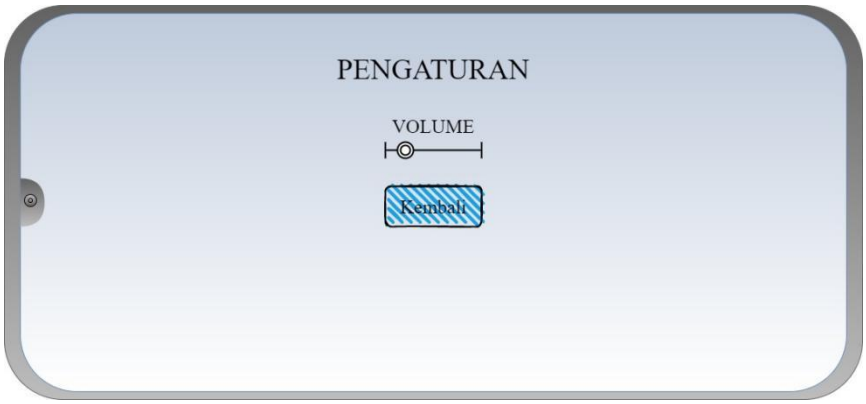
Jika pemain menekan tombol Tutorial, maka akan masuk antarmuka tutorial yang berisi penjelasan tentang cara bermain *game* tersebut seperti apa dan bagaimana. Tersedia juga tombol kembali untuk berpindah ke antarmuka menu utama. Antarmuka tutorial dapat dilihat pada gambar 10.

Gambar 10. UI Tutorial

4) Antarmuka Pengaturan

Jika pemain menekan tombol pengaturan, maka *scene* akan berpindah dari menu utama ke antarmuka pengaturan. Di

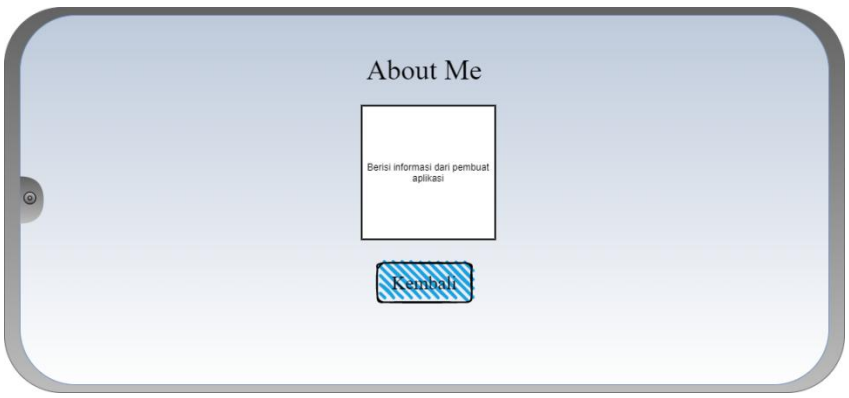
antarmuka pengaturan bisa mengatur tingkat volume suara game, kemudian ada tombol tentang dan kembali. Dapat dilihat pada Gambar 11 .



Gambar 11. UI *Settings*

5) Antarmuka Tentang

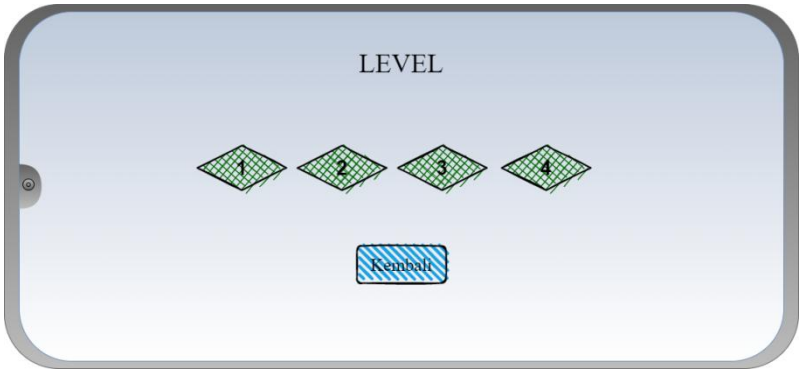
Tombol tentang fungsinya untuk menampilkan data diri dari pengembang *game.* Memiliki tombol kembali untuk kembali ke menu pengaturan dan tombol kembali yang ada di pengaturan jika di tekan akan kembali ke antarmuka menu utama dapat dilihat pada Gambar 12 .



Gambar 12. UI Informasi Pengembang

6) Antarmuka Level

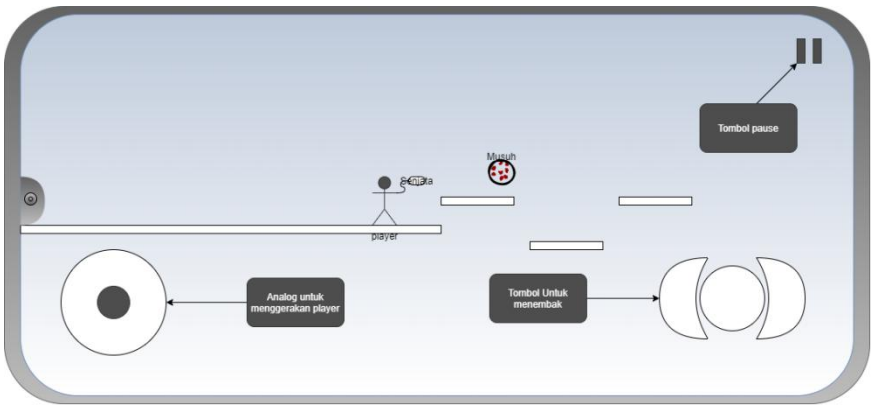
Jika tombol mulai di tekan, *scene* berpindah dari Antarmuka Menu Utama ke antarmuka Level. Di antarmuka level tersedia beberapa level yang dapat di pilih dan tersedia juga tombol kembali untuk berpindah ke antarmuka menu utama. Dapat dilihat pada Gambar 13 .



Gambar 13. UI Level Menu

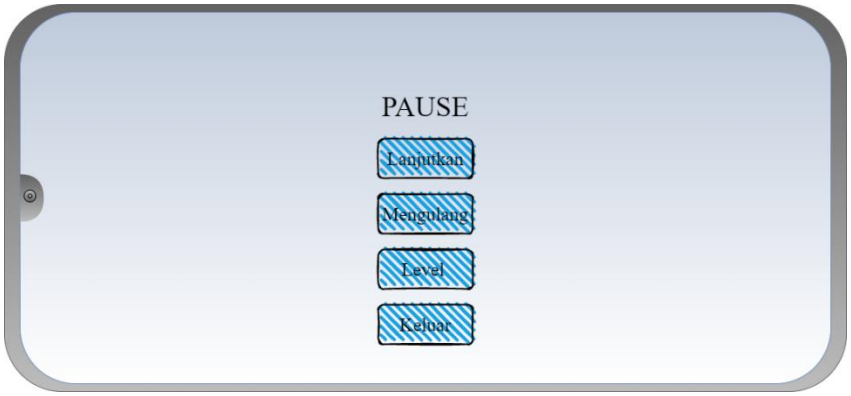
7) Antarmuka *Gameplay* dan Menu *Pause*

Jika pemain menekan tombol level sesuai pilihan pemain, maka *scene* akan berubah sesuai level yang dipilih dan *game* dimulai. *Scene* berganti ke *scene gameplay* dan ada beberapa tombol pada tampilan layar, seperti tombol analog yang fungsinya untuk menggerakkan pemain, tombol tembak dan juga tombol *pause* gunanya untuk masuk ke menu *pause.* Dapat dilihat Gambar 14 .



Gambar 14. *Gameplay* (rancangan sementara)

Jika tombol *pause* di tekan, maka akan muncul tampilan antarmuka menu *pause.* Ada beberapa tombol yang ada di antarmuka menu *pause* yaitu tombol lanjutkan, gunanya untuk melanjutkan game yang sedang berlangsung. Ada tombol mengulang, fungsinya untuk mengulang level atau *scene* yang sedang dimainkan. Kemudian tombol level yang dimana jika di tekan akan berpindah *scene* ke menu level. Dan terakhir tombol keluar untuk berpindah ke menu utama. Dapat dilihat pada Gambar 15.



Gambar 15. UI *Pause* Menu

e. **Desain Karakter**

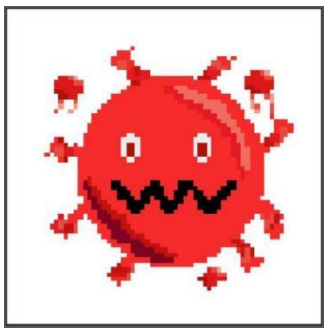
1) Player

Merupakan *playable character* yang dapat dimain oleh pemain, adapun desain *player* yang dibuat ditunjukan pada Gambar 16.

Gambar 16. Desain Player

2) *Enemy*

Merupakan karakter lawan dari pemain yang dapat melakukan fase menyerang ke arah pemain. Ditunjukan pada Gambar 17.

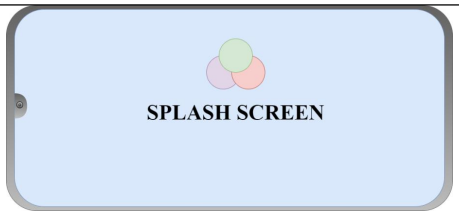
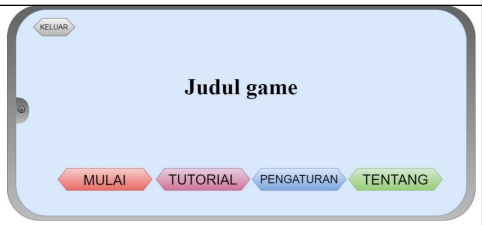
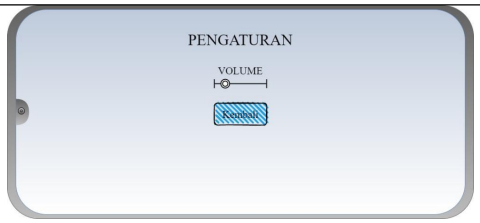
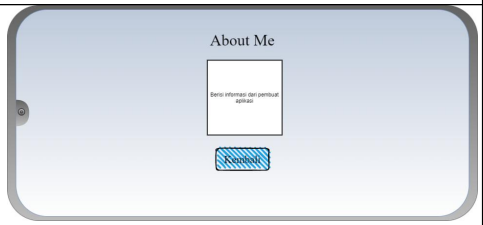


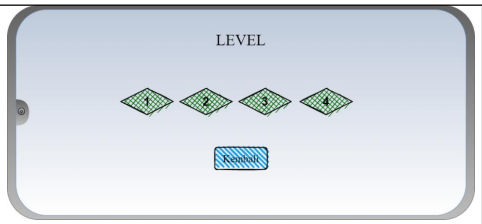
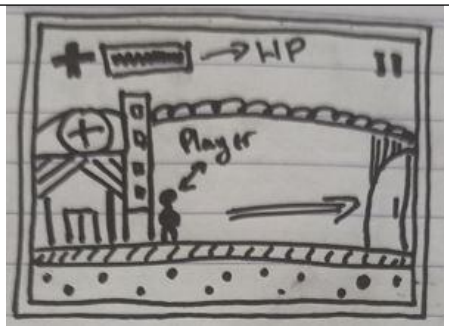
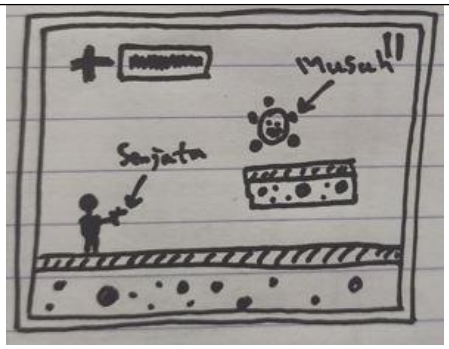
Gambar 17. Desain Enemy (virus)

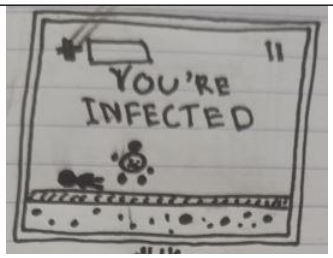
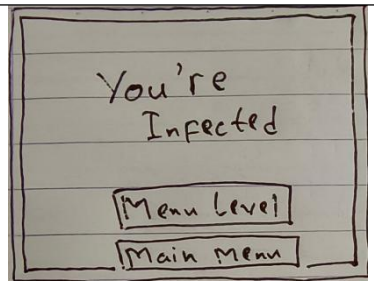
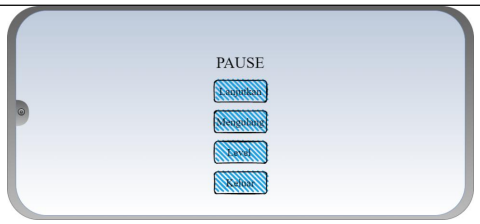
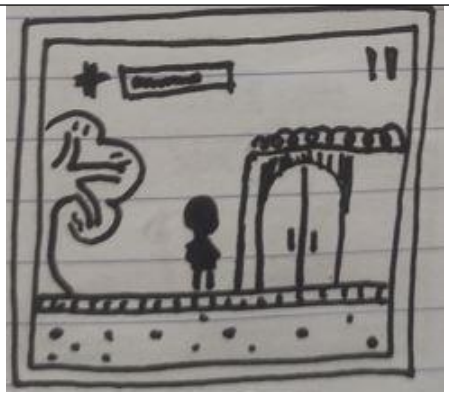
f. *Storyboard Game "Virus Must Die"*

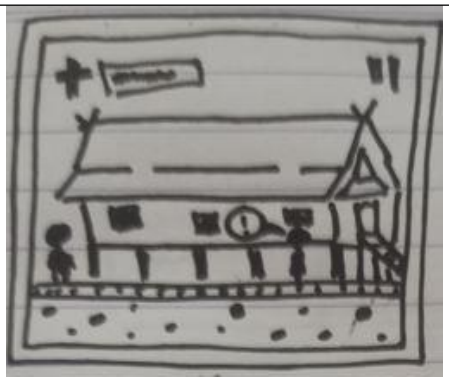
Storyboard digunakan untuk menjelaskan jalan atau alur cerita dari *game* yang akan di rancang. Bisa dilihat pada Tabel 3 berikut.

Tabel 3. *Storyboard Game* "Virus Must Die"

No	Nama	Desain	Keterangan
1	**2**	**3**	**4**
1	*Splash Screen*		1. Menampilkan Logo pembuat 2. Menampilkan Logo Unity 3. Lanjut kehalaman Menu Utama
2	Antarmuka Menu Utama		1. Judul Game 2. Tombol *PLAY*/Mulai 3. Tombol Tutorial 4. Tombol *Option*/Pengaturan 5. Tombol *About*/Tentang 6. Tombol *Exit*/Keluar
3	Antarmuka Turorial		1. Penjelasan tentang cara bermain *game.* 2. Tombol *Back*/Kembali
4	Antarmuka Pengaturan (*Option)*		1. Slider *Volume/Suara* 2. Tombol *Back*/Kembali
5	Antarmuka Tentang (*About)*		1. Data diri dari pengembang *game* 2. Tombol *Back*/Kembali

6	Antarmuka Menu Level		1. Tombol Level 1 2. Tombol Level 2 3. Tombol Level 3 4. Tombol Level 4 5. Tombol *Back*/Kembali
7	Level 1		1. Level 1, Diawali dengan ada nya dialog antara player dan NPC (*Non Playable Character)* 2. Memberikan misi ke player
8	*Gameplay Level 1*		1. *Game* dimulai dengan *player* keluar dari rumah sakit. 2. *Player* telah mendapatkan Misi 3. Diberikan senjata kepada *player* 4. *Player* harus keluar melalui gerbang untuk menuju tujuan 5. Tombol *Pause*
9	*Gameplay Level 1*		1. Dalam perjalanan *player* akan dihadapkan dengan rintangan dan Virus (musuh), untuk dilewati dan dikalahkan. 2. Virus (musuh) dapat menyerang *player* 3. *Player* dapat menembak untuk melindungi diri 4. Tombol *Pause*

10	*Gameplay Level 1*		1. Bar HP/nyawa *player* dapat habis 2. Muncul tulisan setelah *player* kalah dari Virus (musuh)
11	Antarmuka Menu Kalah		1. Tombol Menu Level 2. Tombol *Main Menu*/Menu Utama
12	Antarmuka P*ause Menu*		1. Tombol *Continue*/Lanjut 2. Tombol *Restart*/Mengulang 3. Tombol Level 4. Tombol *Exit*/Keluar (kembali ke menu utama)
13	*Gameplay Level 1*		1. Berhasil melawati rintangan dan mengalahkan musuh, *Player* hampir sampai di tempat tujuan 2. Harus melewati gerbang untuk masuk ke tempat tujuan 3. Akan muncul panel untuk pertanyaan, dengan topik yang mengedukasi seperti Perhitungan atau yang lain

14	*Gameplay Level 1*		1. *Player* berhasil masuk 2. *Player* telah sampai ke tujuan 3. Berjalan menuju tanda yang diberikan
15	*Gameplay Level 1*		1. Berjalan ke tanda seru 2. Dialog antara *player* dan NPC (*Non Playable Character)*
16	Antarmuka Menu Menang		1. Misi Selesai 2. Tombol *Main Menu* / Menu Utama

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil

Laporan tugas akhir ini mencoba untuk membuat sebuah *game* 2D *platformer* yang memiliki unsur edukasi didalamnya berbasiskan *android.* Dalam proses pembuatan *game* ini penulis menggunakan *software Unity.*

Game ini diberi nama *Virus Must Die,* menghasilkan *file* yaitu *VMD 1.0.apk.* Pembuatan *game* dubuat melalui *unity* dengan versi 2020.3.1f1 LTS*,* kemudian menggunakan modul *android* yang sudah ditambahkan ke dalam *unity*. *Unity* berfungsi sebagai editor bahasa pemrograman yang pada penelitian ini menggunakan bahasa C# sekaligus *emulator* untuk menguji coba hasil *running* dari program C# yang telah dibuat. Dari program ini menghasilkan *file .apk* sesuai dengan modul yang digunakan dan nantinya diaplikasikan ke dalam *android*.

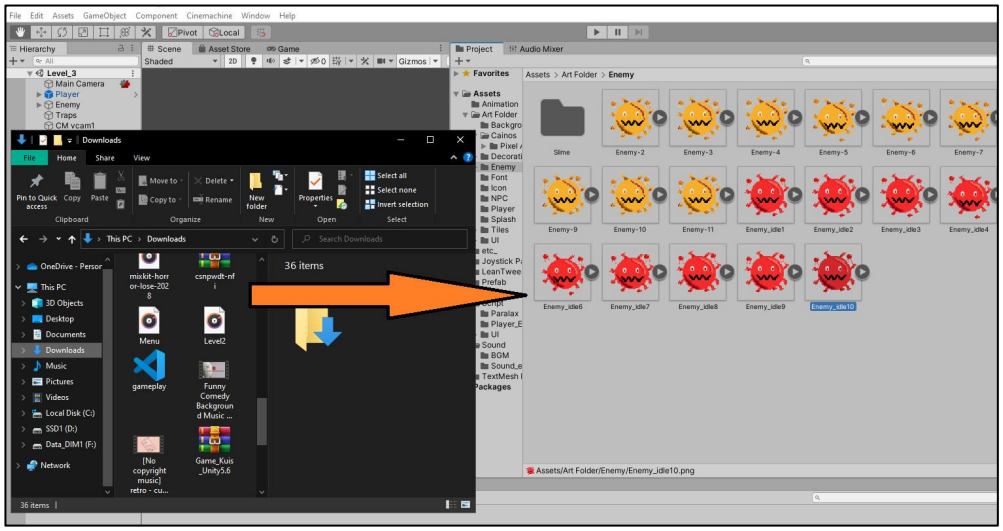
Agar dapat menjalankan aplikasi game *Virus Must Die* pada *handphone android,* pengguna hanya perlu mentransfer *file .apk* melalui kabel data ke dalam *memory handphone* atau *bluetooth.* Setelah aplikasi telah di pindahkan ke *handphone, install* aplikasi dan aplikasi *game* siap dimainkan.

4.1.1 Implementasi *Unity*

Tahap ini merupakan tahap yang paling utama dalam pembuatan *game.* Seluruh konten *game* berupa model 2D dan *script* disatukan dan dibuat didalam *software unity* kemudian seluruh *assets* digabungkan hingga membentuk satu kesatuan utuh yang siap dijalankan dalam bentuk aplikasi yang berdiri sendiri atau aplikasi independen.

a. *Importing Object 2D*

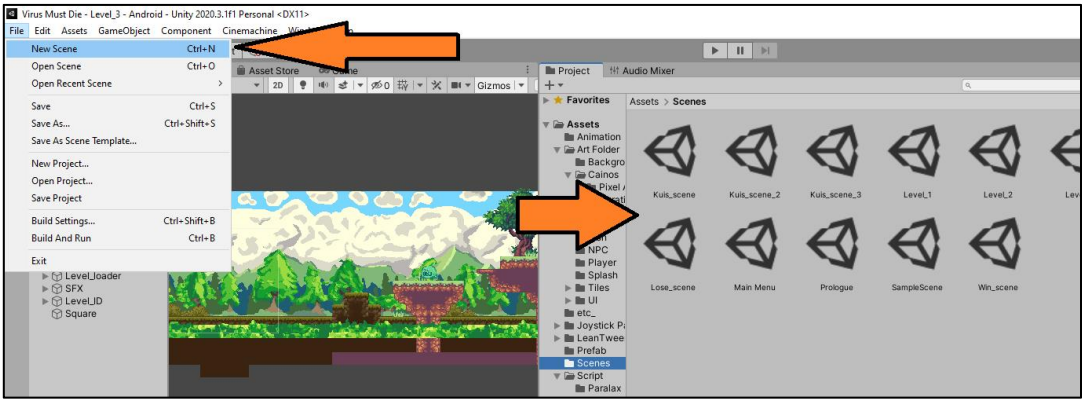
Object 2D yang telah di *export* dalam bentuk *.png,* di import kedalam *folder* yang telah dibuat didalam *unity,* serta *assets* lainnya seperti musik, *sound effect, font text*, dan *assets* pendukung dalam penyelesaian pembuatan *game* ditunjukan pada Gambar 18.



Gambar 18. Implementasi Unity I

b. *Scene*

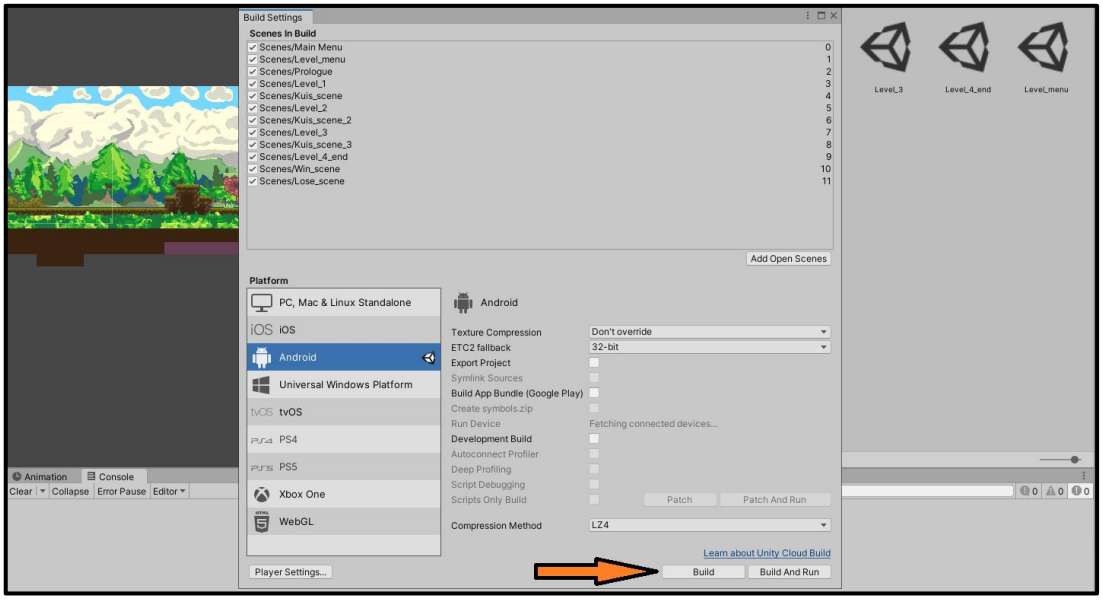
Tahap berikutnya adalah pembuatan *scene*. *Scene* disini berfungsi sebagai pemisah antar *event* (kejadian). Misalkan tampilan menu utama dan *gameplay*, dibuatlah *scene* untuk masing - masing *event* yang terjadi agar tidak menimbulkan *error* ditunjukan pada Gambar 19.



Gambar 19. Pembuatan Scene

c. *Build*

Tahap *build* merupakan tahap akhir dalam pembuatan *game.* Dari *scene* - *scene* yang telah dibuat kemudian disatukan menjadi aplikasi yang utuh dan dapat dijalankan ditunjukan pada Gambar 20.



Gambar 20. Build Aplikasi Game

4.1.2 Implementasi UI

a. *Splash Screen*

Tampilan ini merupakan tampilan saat pertama kali membuka dan menjalankan aplikasi *game* ditunjukan pada Gambar 21.

Gambar 21. *Splash Screen*

b. *Main Menu*

Setelah tampilan *splash screen* selesai, maka berpindah ke halaman menu utama dan memiliki beberapa tombol. Yaitu *button play, button tutorial, button options, button about,* dan *button close (tanda X)* ditunjukan pada Gambar 22.

Gambar 22. *Main Menu*

c. *Tutorial Menu*

Ketika menyentuh *button tutorial,* akan muncul tampilan panel *tutorial,* yang berisi informasi cara bermain *game Virus Must Die* ditunjukan pada Gambar 23.

Gambar 23. *Tutorial Menu*

d. *Options Menu*

Ketika menyentuh *botton options*, tampilan panel *options menu* akan muncul dan memiliki konten berupa pengaturan *volume* suara ditunjukan pada Gambar 24.

Gambar 24. *Options Menu*

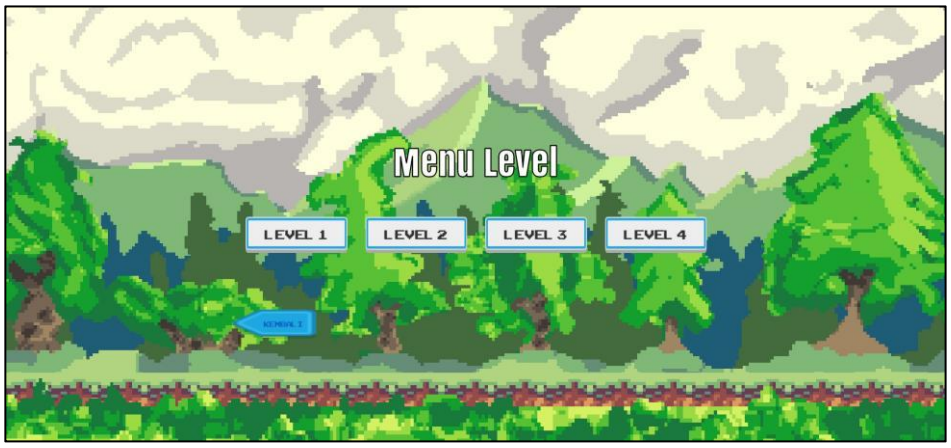
e. *About Menu*

Pada tampilan panel *about,* menampilkan informasi dari developer *game Virus Must Die* . Ditunjukan pada Gambar 25.

Gambar 25. *About Menu*

f. *Level Menu*

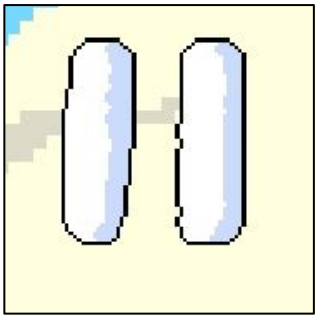
Tampilan menu level memiliki pilihan, bisa memilih bebas level atau memulai nya dari level 1 kemudian menyelesaikan nya secara bertahap. Tersedia *button kembali* untuk berpindah *scene* ke menu utama ditunjukan pada Gambar 26.



Gambar 26. *Level Menu*

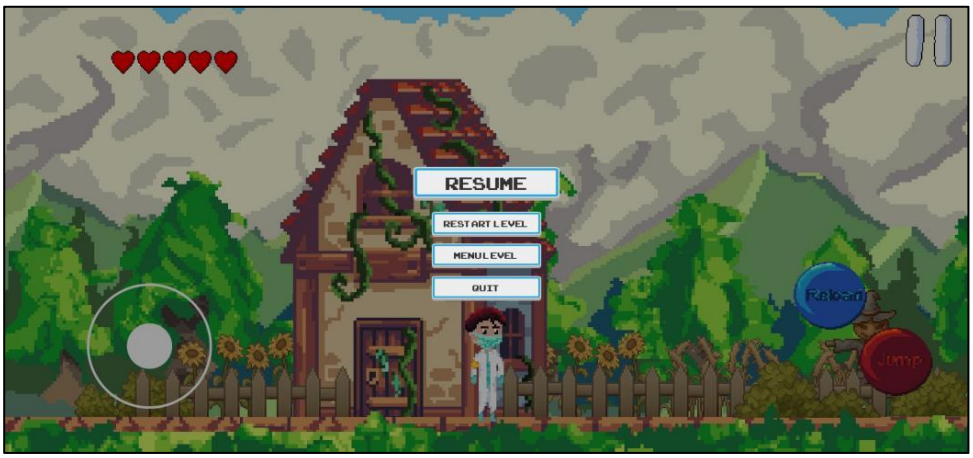
g. *Pause Menu*

Setelah memilih level dan berpindah *scene* ke *scene gameplay, player* dapat menjeda *game* dengan menekan *button pause* untuk membuka *pause menu* . *Button pause* ditunjukan pada Gambar 27.



Gambar 27. *Pause Button*

Tampilan panel *pause menu* memiliki beberapa *button* yaitu *resume (*melanjutkan permainan*), restart level(*mengulang level yang dimainkan*), menu level(*berpindah *scene* ke *scene* menu level*),* dan *quit(*keluar kembali ke mau utama) ditunjukan pada Gambar 28.



Gambar 28. *Pause Menu*

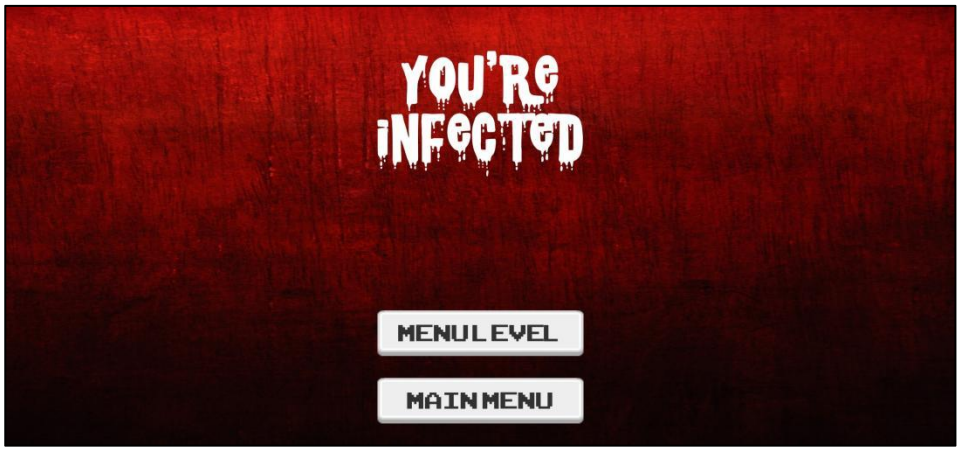
h. *Win Scene*

Tampilan *scene Win* muncul jika *player* berhasil menyelesaikan level terakhir (level 4). Pada *scene* ini memilki *button main menu* untuk kembali ke menu utama. Ditunjukan pada Gambar 29.

Gambar 29. *Win Scene*

i. *Lose Scene*

Lose scene terjadi jika *player* kehabisan *health(*nyawa). Memiliki pilihan *button* yaitu *button menu level* untuk berpindah ke *scene* menu level dan *button main menu* untuk kembali ke menu utama. Ditunjukan pada Gambar 30.



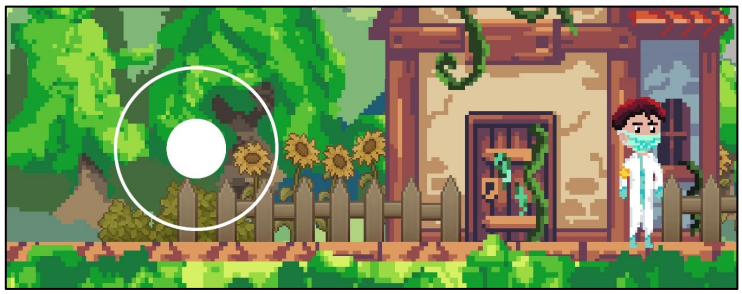
Gambar 30. *Lose Scene*

4.1.3 Implementasi *Game Mechanic*

a. Player

1) *Movement Player*

Player dalam *game Virus Must Die* ini menggunakan *joystick* sebagai kontrol gerakan nya. Ditunjukan pada Gambar 31.



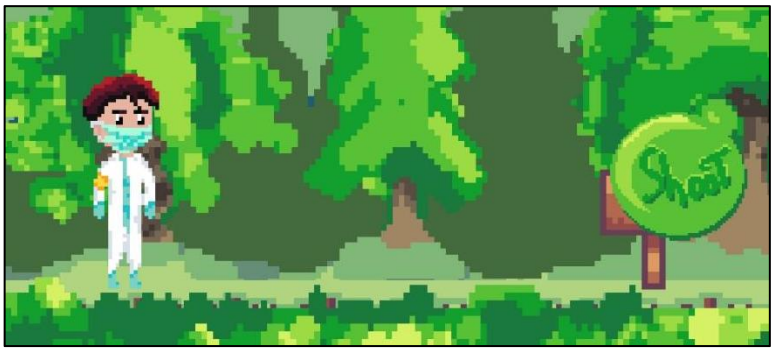
Gambar 31. *Joystick*

2) *Shoot* dan *reload*

Player dapat menembakan peluru sebanyak 3 tembakan dan sebelum menembak *player* harus menyentuh *button reload* terlebih dahulu, jika tidak maka *button shoot* tidak akan tampil. Ketika peluru sudah habis maka *button shoot* dinonaktifkan dan *button reload* aktif. *Button reload* ditunjukan pada Gambar 32.

Gambar 32. *Button Reload*

Jika *button reload* disentuh, maka *button shoot* muncul dan *player* dapat menembak yang tembakan tersebut dibatasi sebanyak 3(tiga) tembakan. Ditunjukan pada Gambar 33.



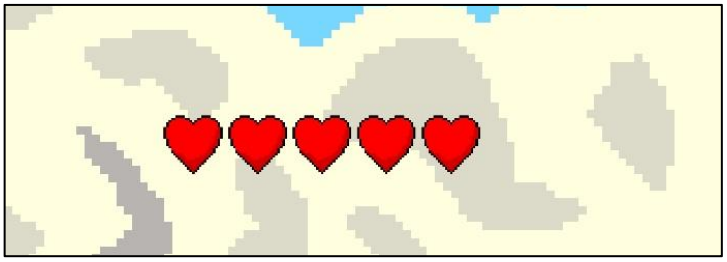
Gambar 33. *Button Shoot* (aktif)

Setelah peluru ditembakan sebanyak 3 kali, maka peluru habis dan *button shoot* menghilang. Ditunjukan pada Gambar 34.

Gambar 34. Peluru Habis (*Button Reload* aktif)

3) *Player Health*

Player memiliki 5 *health* atau nyawa yang perlu dijaga. *Health* dapat berkurang jika terkena Enemy (musuh) dan jika *health player* habis maka *player* akan terinfeksi (kalah dan berpindah *scene* ke *scene lose).* Ditunjukan pada Gambar 35.



Gambar 35. *Health Player*

b. Enemy

1) *Enemy AI*

Menambahkan unsur AI kepada *enemy* menghasilkan gerakan yang dinamis berdasarkan *behavior* (perilaku) yang dibuat.

a) *Patrol*

Enemy bergerak kedepan kemudian jika mencapai batas gerak yang sudah ditentukan maka *enemy* berbalik arah dan mengulangi *behavior* nya lagi. Ditunjukan pada Gambar 36.

Gambar 36. *Enemy Patrol*

b) *FSM (Finite State Machine)*

Enemy bergerak dalam kondisi tertentu, jika kondisi itu dicapai maka akan menyebabkan kondisi yang baru. Kondisi *idle* / diam ditunjukan pada Gambar 37.

Gambar 37. *Enemy Idle*

Pola gerakan musuh berubah yang awal nya *idle/*diam menjadi *attack*/menyerang, hal ini terjadi karena kondisi tertentu terpenuhi. Ditunjukan pada Gambar 38.

Gambar 38. *Enemy Agro*

Kondisi dari awal nya *idle/*diam menjadi *attack*/menyerang terjadi karena musuh memiliki kondisi. Jika *player* masuk kedalam lingkaran dan mendekati musuh maka musuh akan menyerang *player* dan sebaliknya, jika *player* keluar dari lingkaran tersebut maka musuh akan berhenti menyerang dan kembali ke kondisi awal. Ditunjukan pada Gambar 39.

Gambar 39. *FSM Enemy*

4.1.4 Implementasi Edukasi

Tiap level yang diselesaikan akan berpindah ke *scene kuis* dan jumlah pertanyaan yang perlu dijawab ada 5 dengan tipe soal yaitu Matematika Dasar, jika 1 saja jawaban salah maka *player* akan mengulang level yang sudah dimainkan sebelum nya. Ditunjukan pada Gambar 40.

Gambar 40. Kuis *Scene*

4.2 Pembahasan

Dalam hal ini Penulis akan menjabarkan tentang langkah-langkah implementasi yang dilakukan dalam menyelelesaikan *Game 2D Platformer "Virus Must Die"* berdasarkan teori dan materi yang telah dipelajari oleh Penulis adalah sebagai berikut :

a. Pencarian Permasalahan Sebagai Dasar Pembuatan Aplikasi *Game*

Aplikasi *game* yang dibangun bebasiskan *android* ini adalah sebuah aplikasi yang dalam pembuatan nya menggunakan *unity* dan juga menggunakan Java SE Development kit untuk mengembangkan aplikasi yang berbasiskan *android*. Penulis juga membuat diagram HIPO (*Hierarki Input Proses Output*), desain UI, dan *storyboard* untuk aplikasi *Game 2D Platformer "Virus Must Die"* ini sebagai objek penelitian. Dan juga penambahan unsur edukasi merupakan permasalahan yang didapat dan dibuat sebagai akhir penyelesain aplikasi *game* yang dibuat. Hal ini bertujuan untuk memperjelas arah dari aplikasi *game* yang dibuat agar dapat lebih dimengerti oleh pembaca, membuat pembaca mengerti aplikasi *game* ini dan menjelaskan kegunaan dari aplikasi *game* ini.

b. Merancang Estimasi Pengerjaan Projek Aplikasi *Game*

Perancangan estimasi pengerjaan projek sangat penting untuk memperjelas kegiatan apa saja yang dikerjakan dan menjaga projek yang dibuat agar tidak lewat dari waktu penyelesaian *game tersebut.*

c. Perancangan dan Arsitektur Model Untuk *Assets* Pada *Game*

Pada tahap ini dilakukan nya pembuatan model - model untuk dijadikan *assets* dalam perancangan aplikasi *game* yang dibuat. Mulai dari UI, karakter utama *game*, NPC, linkungan dan *assets* lainnya.

d. Penulisan *Coding* dan *Testing* code

Penulis mengetikkan kode - kode (coding) untuk rancangan aplikasi *game* menggunakan komputer sesuai dengan bahasa pemrogrman yang digunakan.

Coding yang ditulis didapat dari forum-forum dan blog di internet dengan memodifikasi dari *coding* yang sudah ada dan di *test* untuk memeriksa ada *error* atau tidak.

e. Pengujian Aplikasi *Game Virus Must Die*

Dilakukan pengujian dilakukan untuk mengetahui apakah aplikasi yang dibuat berjalan dengan benar sehingga bisa menghasilkan fungsi-fungsi yang dikehendaki. Pengujian dilakukan untuk mengetahui keterbatasan dan kelemahan aplikasi *game* yang dibuat agar dapat sebisa mungkin dilakukan penyempurnaan.

4.2.1 Hasil Response Pengguna

Data hasil response digunakan untuk mengevaluasi *game* yang dibuat berdasarkan *testing* serta respon dari para responden yang mencoba *game* yang dibuat. Diberikan pertanyaan dan beberapa pernyataan yang menjadi parameter penilaian dalam *testing* yang dilakukan para responden.

Pada penelitian ini menggunakan *Google Form* sebagai media untuk mencari data response dari responden. Dari data yang didapatkan, sebanyak 20 responden yang telah mengisi form penilain *game.*

Ada beberapa pertanyaan yang menjadi kunci utama dalam penilaian, yaitu pertanyaan tentang resolusi, versi *android* yang digunakan, dan penginstalan *game* responden dalam melakukan ujicoba. Hasil response data dalam bentuk Ditunjukan pada Gambar 41.

No.	Pertanyaan	Jawaban						
		qHD/Quarter HD (8:5)	HD (16:9)	Full HD (16:9)	Full HD+ (16:9, 18:9, 19:9)	2K/QHD/Quad HD (16:9, 18:9, 19:5:9)		
1	Berapa resolusi layar yang Smartphone (Android) Anda gunakan?	1	9	6	2	2		

No.	Pertanyaan	Jawaban						
		Android 6.0 (Marshmallow)	Android 7.0 – 7.1 (Nougat)	Android 8.0 – 8.1 (Oreo)	Android 9 (Pie)	Android 10 (Android Q)	Android 11	Android 12 (Snow Cone)
2	Sistem operasi Andoid versi berapa yang berjalan pada Smartphone Anda?	2	1	1		4	11	1

No.	Pertanyaan	Jawaban				
		STB (Sangat Tidak Baik)	TB (Tidak Baik)	S (Sedang)	B (Baik)	SB (Sangat Baik)
3	Bagaimana saat proses menginstal game "VIRUS MUST DIE" ?			1	11	8

Gambar 41. Tabel Jumlah Penilaian dari Hasil Kuesioner

a. Skala Likert (*Likert Scale*)

Dalam proses penilaian pada penelitian ini, yaitu menggunakan skala likert. Skala Likert adalah salah satu bentuk skala yang dilakukan untuk mengumpulkan data demi mengetahui atau mengukur data yang bersifat kualitatif (Agung .P, 2020).

Diberikan skala penilaian STB (Sangat Tidak Baik), TB (Tidak Baik), S (Sedang), B (Baik), dan SB (Sangat Baik) dengan nilai dari STB ke SB (Pn) yaitu 1, 2, 3, 4, dan 5. Ada beberapa tahap dalam penyelesain untuk mendapatkan hasil index persentase dari data kuesioner yang sudah diisi, yaitu :

1) Perhitungan Jumlah Data Responden

T x Pn

T = Total jumlah responden yang memilih

Pn = Pilihan angka skor Likert

2) Interpretasi Skor Perhitungan

Y = skor tertinggi likert x jumlah responden

X = skor terendah likert x jumlah responden

Jumlah skor tertinggi untuk item “Sangat Baik” adalah 5 x 20 = 100, sedangkan item “Sangat Tidak Baik” adalah 1 x 20 = 20.

3) Perhitungan Jumlah Data Responden

Rumus Index % = Total Skor / (Y * X)\

4) Rumus Interval

I = 100 / Jumlah Skor (Likert)

Maka = 100 / 5 = 20

Hasil (I) = 20

(Ini adalah intervalnya jarak dari terendah 0 % hingga tertinggi 100%)

Berikut kriteria interpretasi skornya berdasarkan interval:

- Angka 0% – 19,99% = Sangat (tidak setuju/buruk/kurang sekali)
- Angka 20% – 39,99% = Tidak setuju / Kurang baik)
- Angka 40% – 59,99% = Cukup / Netral
- Angka 60% – 79,99% = (Setuju/Baik/suka)
- Angka 80% – 100% = Sangat (setuju/Baik/Suka)

Hasil dari perhitungan menggunakan rumus yang dijelaskan, ditunjukan pada Gambar 42.

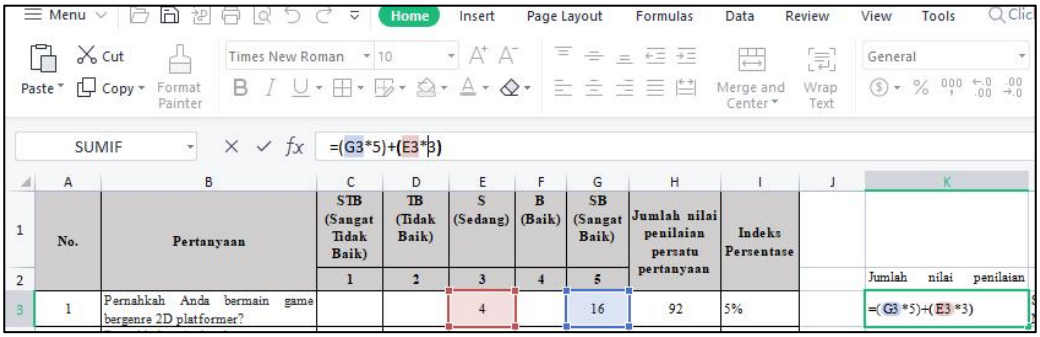
No.	Pertanyaan	STB (Sangat Tidak Baik)	TB (Tidak Baik)	S (Sedang)	B (Baik)	SB (Sangat Baik)	Jumlah nilai penilaian persatu pertanyaan	Indeks Persentase
		1	2	3	4	5		
1	Pernahkah Anda bermain game bergenre 2D platformer?			4		16	92	5%
2	Pernahkah Anda bermain game platformer yang memiliki unsur pendidikan didalamnya?			7		13	86	4%
3	Apakah Anda menikmati game dengan gaya 2D?			4		16	92	5%
4	Bagaimana saat proses menginstal game "VIRUS MUST DIE" ?			1	11	8	87	4%
5	Bagaimana dengan interaksi Anda pada menu-menu pada game?			2	11	7	85	4%
6	Bagaimana dengan tampilan/ UI pada game "VIRUS MUST DIE" ini?		1	2	13	4	80	4%
7	Apakah gameplay dari game "VIRUS MUST DIE" menarik?			3	13	4	81	4%
8	Bagaimana dengan cerita dan tujuan di game ini?			1	12	7	86	4%
9	Menurut Anda, Bagaimana dengan kontrol Player pada game?		1	3	13	3	78	4%
10	Bagaimana dengan desain level pada game "VIRUS MUST DIE"?			2	12	6	84	4%
11	Bagaimana dengan desain karakter, musuh, Background game dan musik pada game?			1	15	4	83	4%
12	Pada game "VIRUS MUST DIE" memiliki unsur Edukasi seperti Kuis Matematika Dasar (penjumlahan pengurangan) , Bagaimana menurut Anda?			1	12	7	86	4%
13	Bagaimana dengan kesulitan dalam menjawab Kuis pada game?			6	12	2	76	4%
14	Bagaimana dengan button pause yang muncul pada saat gameplay?			2	13	5	83	4%
15	Bagaimana dengan button yang muncul pada tampilan menu, apakah berfungsi dengan baik?		1	2	13	4	80	4%
16	Bagaimana dengan implementasi AI pada enemy seperti Patrol (patroli) dan Attack(menyerang)?		1	2	14	3	79	4%
17	Bagaimana menurut Anda, apakah implementasi AI berjalan baik?			2	13	5	83	4%
18	Bagaimana menurut Anda dengan tampilan menu Lose jika Player kalah dan tampilan Win jika Player berhasil menyelesaikan semua Level?			5	10	5	80	4%
19	Bagaimana dengan game ini, apakah masih memiliki Bug (kesalahan sistem)?		1	6	10	3	75	4%
20	Apakah Bug tersebut menggagu dalam memainkan game tersebut?		2	7	8	3	72	4%
Total Jumlah								82%

Gambar 42. Tabel Data Hasil Kuesioner

Untuk menghitung jumlah nilai persatu pertanyaan menggunakan rumus T x Pn dimana T adalah total responden memilih jawaban tersebut dan Pn adalah pilihan

angka skor Likert. Jumlah dari jawaban yang dipilih responden pada pertanyaan akan dikalikan sesuai dengan bobot skor nya.

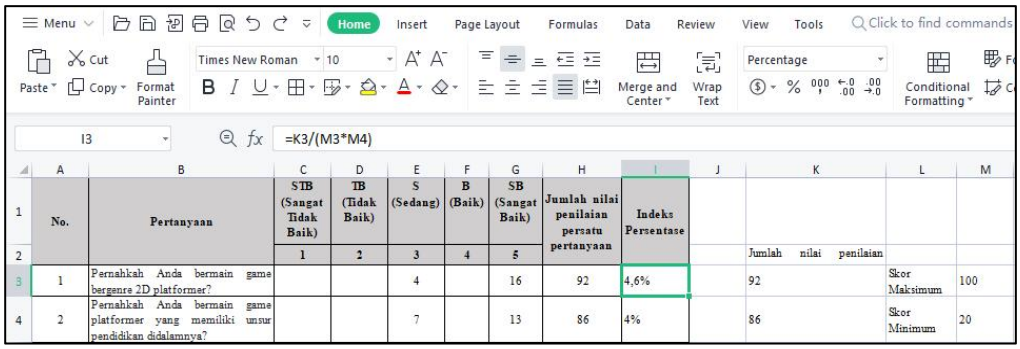
Misalkan, untuk S (Sedang) dengan nilai bobot 3 berjumlah 4 responden dan untuk SB (Sangat Baik) dengan nilai bobot berjumlah 16 responden. Kemudian dilakukan nya perhitungan menggunakan rumus T x Pn untuk mendapatkan nilai jumlah data responden, dimana jumlah jawaban yang dipilih responden dikali dengan bobot skor yang ditunjukan pada Gambar 43.



Gambar 43. Rumus Perhitungan Jumlah Nilai Persatu Pertanyaan

Setelah mendapatkan jumlah data responden nya, selanjutkan mencari indeks persentase setiap pertanyaan dengan rumus Index % = Total Skor / (Y * X). Total skor didapatkan dari jumlah data responden yang didapat sebelumnya, untuk nilai Y serta X didapat dengan menggunakan rumus "Y= skor tertinggi likert x jumlah responden" dan rumus untuk nilai X adalah "X= skor tertinggi likert x jumlah responden" dengan keterangan untuk skor tertinggi nya yaitu 5 dan skor terendeh nya ialah 1. Untuk responden nya berjumlah 20 berdasarkan yang sudah mengisi kuesioner.

Jadi, jumlah data responden yang telah didapat dibagi dengan hasil dari perkalian antara skor tertinggi likert (Y) dan skor terendah likert (X) yang ditunjukan pada Gambar 44.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	No.	Pertanyaan	STB (Sangat Tidak Baik)	TB (Tidak Baik)	S (Sedang)	B (Baik)	SB (Sangat Baik)	Jumlah nilai penilaian persatu pertanyaan	Indeks Persentase				
2			1	2	3	4	5				Jumlah nilai penilaian		
3	1	Pernahkah Anda bermain game bergenre 2D platformer?			4		16	92	4,6%		92	Skor Maksimum	100
4	2	Pernahkah Anda bermain game platformer yang memiliki unsur pendidikan didalamnya?			7		13	86	4%		86	Skor Minimum	20

Gambar 44. Rumus Menghitung Indeks Persentase

Jika semua hasil telah didapat, jumlah semua nilai indeks persentase yang didapat, kemudian dilihat nilai yang sudah berhasil didapat untuk menyimpulkan hasilnya dengan kriteria interpretasi skornya berdasarkan interval yang telah dibuat sebelumnya yang ditunjukan pada Gambar 45.

Angka 0% – 19,99% = Sangat (tidak setuju/buruk/kurang sekali)
Angka 20% – 39,99% = Tidak setuju / Kurang baik)
Angka 40% – 59,99% = Cukup / Netral
Angka 60% – 79,99% = (Setuju/Baik/suka)
Angka 80% – 100% = Sangat (setuju/Baik/Suka)

Gambar 45. Kriteria Interpretasi

Dapat disimpulkan, didapatkan 20 responden untuk mengisi kuesioner dengan total index presentase yang didapat adalah 82 % dan kriteria interpretasi skor nya “Sangat Baik”.

4.3 **Pengujian *Black Box***

Pengujian *black box* dilakukan dengan memperhatikan masukan ke sistem dan keluaran dari sistem. Pengujian *black box* yang dilakukan adalah sebagai berikut :

a. Pengujian buka aplikasi

Tabel 4. Tabel Pengujian Buka Aplikasi/Game

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Membuka dan menjalankan aplikasi.	Sentuh logo aplikasi pada *smartphone* yang telah di *install* aplikasi *game 2D platformer Virus Must Die berbasis android*	Aplikasi akan terbuka dan menampilkan *splash screen* dan masuk ke menu utama.	Sesuai

b. Pengujian Halaman Menu Utama

Tabel 5. Pengujian Halaman Menu Utama

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Button Play	Sentuh *button play* pada halaman menu utama aplikasi	Aplikasi akan berpindah ke *scene* menu level	Sesuai
Button Tutorial	Sentuh *button turorial* pada halaman menu utama aplikasi	Aplikasi akan membukatampilan panel *tutorial*	Sesuai
Button Options	Sentuh *button options* pada halaman menu utama aplikasi	Aplikasi akan membuka tampilan panel *options menu*	Sesuai

Button About	Sentuh *button about* pada halaman menu utama aplikasi	Aplikasi akan membuka tampilan panel *about*	Sesuai
Button X (exit)	Sentuh *Button X* pada tampilan menu utama	Aplikasi *game Virus Must Die* akan tertutup	Sesuai

c. Pengujian Tampilan Panel *Tutorial*

Tabel 6. Pengujian Tampilan Panel Tutorial

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Button next (panah)	Sentuh *button next* pada tampilan panel *tutorial*	Tampilan panel akan berubah dan menampilkan penjelasan tentang game lainnya	Sesuai
Button X (exit)	Sentuh *Button X* pada tampilan panel *tutorial*	Tampilan panel *tutorial* tertutup dan menampilkan menu utama kembali	Sesuai

d. Pengujian Tampilan Panel *Options Menu*

Tabel 7. Pengujian Tampilan Panel Options Menu

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Slider Volume	Geser *Slider Volume* pada panel *options menu*	Suara pada *game* akan mengecil jika digeser kekiri dan membesar jika digeser kekanan	Sesuai

Button X (exit)	Sentuh *Button X* pada tampilan panel *options menu*	Tampilan panel *options menu* tertutup dan menampilkan menu utama kembali.	Sesuai

e. Pengujian Tampilan Panel *About*

Tabel 8. Pengujian Tampilan Panel About

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Button X (exit)	Sentuh *Button X* pada tampilan panel *about*	Tampilan panel *about* tertutup dan menampilkan menu utama kembali.	Sesuai

f. Pengujian Halaman Menu level

Tabel 9. Pengujian Halaman Menu Level

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Button Level 1 (Prologue)	Sentuh *button level 1* pada halaman menu level	Aplikasi akan berpindah *scene* dari menu level ke *scene level 1*	Sesuai
Button Level 2	Sentuh *button level 2* pada halaman menu level	Aplikasi akan berpindah *scene* dari menu level ke *scene level 2*	Sesuai
Button Level 3	Sentuh *button level 3* pada halaman menu level	Aplikasi akan berpindah *scene* dari menu level ke *scene level 3*	Sesuai
Button Level 4	Sentuh *button level 4* pada halaman menu level	Aplikasi akan berpindah *scene* dari menu level ke *scene level 4*	Sesuai

Button Kembali (Back)	Sentuh *button Kembali* pada halaman menu level	Aplikasi akan berpindah *scene* kembali ke menu utama	Sesuai

g. Pengujian *Gameplay Level 1 (Prologue)*

Tabel 10. Pengujian Gameplay Level 1 (Prologue)

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Joystick	Gerakan *Joystick* kekiri dan kekanan pada tampilan *gameplay level 1*	*Player* akan bergerak sesuai arah gerakan *joystick*	Sesuai
Dialog NPC	Gerakan *Player* mendekati *NPC* pada tampilan *gameplay level 1*	*Dialog Box* akan muncul setelah mendekati *NPC*	Sesuai
Button Next Dialog	Sentuh *button next* pada *dialog* pada tampilan *gameplay level 1*	*Dialog* berganti setelah menyentuh *button next* pada *dialog*	Sesuai
Button Reload	Sentuh *button reload* pada tampilan *gameplay level 1*	*Button shoot akan muncul* pada tampilan layar	Sesuai
Button Shoot	Sentuh *button shoot* pada tampilan *gameplay level 1*	Peluru akan *respawn* sebanyak 3 kali kemudian *button reload* aktif kembali	Sesuai
Button Jump	Sentuh *button jump* pada tampilan *gameplay level 1*	*Player* melompat ketika *button jump* disentuh	Sesuai
Button Pause	Sentuh *button pause* pada tampilan *gameplay level 1*	Tampilan panel *pause menu* akan muncul	Sesuai

Health (nyawa)	Gerakan *player* mendekati musuh dan biarkan *health player* berkurang	*Health* dari *player* akan berkurangdan jika *health* habis maka *scene* berpindah ke tampilan *lose scene*	Sesuai
Enemy	Kalah kan *enemy* menggunakan tembakan *button shoot*	*Enemy* yang dikalahkan akan menghilang	Sesuai
Clear level	Gerakan *player* sampai ke finish (ujung kanan level)	*Scene* akan berpindah ke *scene* tampilan *Kuis matematika dasar*	Sesuai

h. Pengujian *Gameplay Level 2*

Tabel 11. Pengujian Gameplay Level 2

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Joystick	Gerakan *Joystick* kekiri dan kekanan pada tampilan *gameplay level 2*	*Player* akan bergerak sesuai arah gerakan *joystick*	Sesuai
Button Next Dialog	Sentuh *button next* pada *dialog* pada tampilan *gameplay level 2*	*Dialog* berganti setelah menyentuh *button next* pada *dialog*	Sesuai
Button Reload	Sentuh *button reload* pada tampilan *gameplay level 2*	*Button shoot akan muncul* pada tampilan layar	Sesuai
Button Shoot	Sentuh *button shoot* pada tampilan *gameplay level 2*	Peluru akan *respawn* sebanyak 3 kali dan *button shoot* menghilang kemudian *button reload* muncul	Sesuai
Button Jump	Sentuh *button jump* pada tampilan *gameplay level 2*	*Player* melompat ketika *button jump* disentuh	Sesuai

Button Pause	Sentuh *button pause* pada tampilan *gameplay level 2*	Tampilan panel *pause menu* akan muncul	Sesuai
Health (nyawa)	Gerakan *player* mendekati musuh dan biarkan *health player* berkurang	*Health* dari *player* akan berkurangdan jika *health* habis maka *scene* berpindah ke tampilan *lose scene*	Sesuai
Jebakan Duri	Gerakan *player* mendekati jebakan duri	*Health* dari *player* akan berkurang dan jika *health* habis maka *scene* berpindah ke tampilan *lose scene*	Sesuai
Enemy	Kalah kan *enemy* menggunakan tembakan *button shoot*	*Enemy* yang dikalahkan akan menghilang	Sesuai
Clear level	Gerakan *player* sampai ke finish (ujung kanan level)	*Scene* akan berpindah ke *scene* tampilan *Kuis matematika dasar*	Sesuai

i. Pengujian *Gameplay Level 3*

Tabel 12. Pengujian Gameplay Level 3

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Joystick	Gerakan *Joystick* kekiri dan kekanan pada tampilan M*gameplay level 3*	*Player* akan bergerak sesuai arah gerakan *joystick*	Sesuai
Button Next Dialog	Sentuh *button next* pada *dialog* pada tampilan *gameplay level 3*	*Dialog* berganti setelah menyentuh *button next* pada *dialog*	Sesuai
Button Reload	Sentuh *button reload* pada tampilan *gameplay level 3*	*Button shoot akan muncul* pada tampilan layar	Sesuai

Button Shoot	Sentuh *button shoot* pada tampilan *gameplay level 3*	Peluru akan *respawn* sebanyak 3 kali dan *button shoot* menghilang kemudian *button reload* muncul	Sesuai
Button Jump	Sentuh *button jump* pada tampilan *gameplay level 3*	*Player* melompat ketika *button jump* disentuh	Sesuai
Button Pause	Sentuh *button pause* pada tampilan *gameplay level 3*	Tampilan panel *pause menu* akan muncul dan *gameplay* terjeda	Sesuai
Health (nyawa)	Gerakan *player* mendekati musuh dan biarkan *health player* berkurang	*Health* dari *player* akan berkurangdan jika *health* habis maka *scene* berpindah ke tampilan *lose scene*	Sesuai
Jebakan Duri	Gerakan *player* mendekati jebakan duri	*Health* dari *player* akan berkurang dan jika *health* habis maka *scene* berpindah ke tampilan *lose scene*	Sesuai
Enemy	Kalah kan *enemy* menggunakan tembakan *button shoot*	*Enemy* yang dikalahkan akan menghilang	Sesuai
Clear level	Gerakan *player* sampai ke finish (ujung kanan level)	*Scene* akan berpindah ke *scene* tampilan *Kuis matematika dasar*	Sesuai

j. Pengujian *Gameplay Level 4*

Tabel 13. Pengujian Gameplay Level 4

Pengujian	Skenario Uji	Hasil yang diharapkan	Hasil Pengujian
Joystick	Gerakan *Joystick* kekiri dan kekanan pada tampilan *gameplay level 4*	*Player* akan bergerak sesuai arah gerakan *joystick*	Sesuai
Dialog NPC	Gerakan *Player* mendekati *NPC* pada tampilan *gameplay level 4*	*Dialog Box* akan muncul setelah mendekati *NPC*	Sesuai
Button Next Dialog	Sentuh *button next* pada *dialog* pada tampilan *gameplay level 4*	*Dialog* berganti setelah menyentuh *button next* pada *dialog*	Sesuai
Button Pause	Sentuh *button pause* pada tampilan *gameplay level 4*	Tampilan panel *pause menu* akan muncul	Sesuai
Health (nyawa)	Gerakan *player* mendekati musuh dan biarkan *health player* berkurang	*Health* dari *player* akan berkurangdan jika *health* habis maka *scene* berpindah ke tampilan *lose scene*	Sesuai
Clear level	Habiskan percakapan dengan *NPC*	*Scene* akan berpindah ke *scene* tampilan *Scene Win*	Sesuai

k. Pengujian Enemy

Tabel 14. Pengujian Enemy

Pengujian	Skenario Uji	Hasil yang diharapkan	Hasil Pengujian
Mendekati Musuh Bewarna merah	Dekati musuh warna merah	Musuh akan bergerak mengikuti kita dan animasi dari musuh berubah	Sesuai

Mendekati Musuh Bewarna merah atau hijau	Dekati musuh warna merah atau merah	*Efek* knockback aktif, dimana *player* akan terpental sesuai berlawanan arah dengan datang nya player	Sesuai

l. Pengujian Halaman Kuis

Tabel 15. Pengujian Halaman Kuis

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
InputField "Masukan Jawaban"	Sentuh *inputfield* pada tampilan halaman kuis	*Keyboard android* muncul dan dapat menginput jawaban sesuai pertanyaan	Sesuai
Button Jawab	Setelah menginput jawaban, sentuh *button jawab* pada tampilan halaman kuis	Soal akan berganti ke soal berikutnya	Sesuai
Button Jawab	Sentuh *button jawab* tanpa menginput jawaban	*Scene* akan mengulang kembali level sebelumnya	Sesuai
Button Jawab	Menginput jawaban yang salah, sentuh *button jawab* pada tampilan halaman kuis	*Scene* akan mengulang kembali level sebelumnya	Sesuai

m. Pengujian Tampilan Panel *Pause Menu*

Tabel 16. Pengujian Tampilan Panel Pause Menu

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Button Resume	Sentuh *button resume* pada tampilan panel *pause menu*	*Pause menu* tertutup dan *gameplay* kembali berjalan seperti semula	Sesuai
Button Restart Level	Sentuh *button restart level* pada tampilan panel *pause menu*	*Scene* akan mengulang dari awal sesuai dengan level yang dimainkan	Sesuai
Button Menu Level	Sentuh *button menu level* pada tampilan panel *pause menu*	*Scene* berpindah ke tampilan *scene menu level*	Sesuai
Button Quit	Sentuh *button quit* pada tampilan panel *pause menu*	*Scene* berpindah ke tampilan menu utama	Sesuai

n. Pengujian Halaman *Win Scene*

Tabel 17. Pengujian Halaman Win Scene

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Button Main Menu	Sentuh *button main menu* pada tampilan halaman *win scene*	*Scene* berpindah ke tampilan menu utama	Sesuai

o. Pengujian Halaman *Lose Scene*

Tabel 18. Pengujian Halaman Lose Scene

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Button Menu Level	Sentuh *button menu level* pada tampilan halaman *lose scene*	*Scene* berpindah ke tampilan *scene menu level*	Sesuai
Button Main Menu	Sentuh *button main menu* pada tampilanhalaman *lose scene*	*Scene* berpindah ke tampilan menu utama	Sesuai

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dalam Pembuatan *Game 2D Platformer Virus Must Die* Berbasis *Android* Menggunakan *Unity* ini di harapkan dapat memberikan kemudahan dan hiburan yang lebih menarik dan berdaya guna sebagai akhir laporan, penulis dapat menarik kesimpulan sebagai berikut:

a. Pembaca dapat mengetahui proses pembuatan *Game 2D Platformer Virus Must Die* Berbasis *Android* dari awal hingga akhir, dari me-*import assets* 2D ke dalam *software Unity,* pembuatan *scene* Unity untuk tempat *assets* diletakkan, dan *build,* yaitu mengolah setiap *scene* yang dibuat menjadi satu- kesatuan dalam bentuk aplikasi.

b. Aplikasi *game* ini memiliki unsur edukasi degan menambahkan pertanyaan matematika dasar yang membuat *User/*pengguna bisa bermain sambil belajar.

c. Aplikasi *game* ini dapat berjalan di *smartphone* berbasis *android* dan dapat dijadikan media belajar sambil bermain.

d. Dilakukan nya kuesioner untuk melihat respon pengguna terhadap *game* dan berhasil mendapatkan 20 responden dengan total index % sebesar 82% maka aplikasi *game* ini mendapatkan hasil sangat baik.

5.2 Saran

Setelah dilakukan nya penelitian ini, disarankan:

a. Penambahan *sound effect* dan menambahkan *background* musik yang lebih menarik

b. Menambahkan fitur *level unlocked*

c. Memperbaiki Fitur lompat (*jump)* yang optimal.

DAFTAR PUSTAKA

Adani, Muhammad Robith. "Pengertian Storyboard dan Cara Membuatnya untuk Video Marketing." *Pengertian Storyboard*, Sekawan Media, Selasa Desember 2020, https://www.sekawanmedia.co.id/pengertian-storyboard/.

Andrea, R., Akbar, R. I., & Fitroni, M. 2014. Developing battle of Etam earth game agent with finite state machine (FSM) and sugeno fuzzy. ICCS Proceeding, 1(1), 184-187.

Bracey, Kezz. "Alternatif Adobe: Aplikasi Seni Pixel." *adobe-alternatives-pixel-art--cms-28911*, envato tuts+, 17 Juli 2017, https://webdesign.tutsplus.com/id/articles/adobe-alternatives-pixel-art--cms-28911.

Dewanta, Raven Anugra. "Apa Itu Visual Studio Code." *https://www.griyawebsite.com/mengenal-apa-itu-visual-studio-code-dan-bahasa-pemrograman/*, Griya Media Nusantara, 14 Agustus 2021, https://www.griyawebsite.com/mengenal-apa-itu-visual-studio-code-dan-bahasa-pemrograman/.

Fadjar Efendy Rasjid, S. 2010. Android: Sistem Operasi Pada Smartphone. Retrieved Desember 12, 2016, from http://www.ubaya.ac.id/ubaya/articles_detail/7/android--sistem operasipada-smartphone.html

Lathif, Razka. "Apa Itu Draw.io ? Ini Penjelasan Lengkapnya." *apa-itu-draw-io*, Surga Tekno, 5 September 2019, https://surgatekno.com/tech-news/apa-itu-draw-io/.

.P, Agung. "Skala Likert." *serviceacjogja.pro/skala-likert/*, serviceacjogja.pro, 2020, https://serviceacjogja.pro/skala-likert/

Rifai, Wafda Adita. 2015. Pengembangan Game Edukasi Lingkungan Berbasis Android. Yogyakarta: 2015.

Setiawan, I. (2006). Perancangan Software Eembedded System Berbasis FSM. Retrieved from http://www.elektro.undip.ac.id/iwan/Perancangan Software Embedded System Berbasis FSM.pdf

Singkoh, Robert Theophani, Lumenta, Arie S.M., & Tulenan, Virginia. 2016. Perancangan Game FPS (First Person Shooter) Police Personal Training. ISSN: 2301-8402.

Tanzil, S.KOM., M.T.I, Fidelson. "Waterfall Model." *Waterfall Model*, Bina Nusantara, https://socs.binus.ac.id/2018/12/21/waterfall-model/.

Unity. "Unity Manual." *Unity2D*, Unity, https://docs.unity3d.com/Manual/Unity2D.html.

Wibowo, Dimas Catur. “Apa itu Android Studio dan Android SDK?” *apa-itu-android-studio-dan-android-sdk*, dicoding, 30 Mei 2019, https://www.dicoding.com/blog/apa-itu-android-studio-dan-android-sdk/.

Wahyupjl. “Apa itu Unity 3D.” *apa-itu-unity-3d*, Event Kampus, 12 Juli 2018, https://eventkampus.com/blog/detail/1474/apa-itu-unity-3d.

Yulsilviana, Ekawati, and Hanifah Ekawati. “Penerapan Metode Finite State Machine (FSM).” *Penerapan Metode Finite State Machin*e *(FSM) Pada Game Agent Legenda Anak Borneo*, 2019, p. 116.

Zamroni, M.Rosidi, Suryaman, Nizar, & Jalaluddin, Ahmad. 2013. Rancang Bangun Aplikasi Permainan Untuk Pembelajaran Anak Menggunakan HTML 5. Jurnal Teknika (Volume 5 Nomor 2). 489.

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
(STMIK) PALANGKARAYA
Jl. G. Obos No.114 Telp.0536-3224593, 3225515 Fax.0536-3225515 Palangka Raya
email : humas@stmikplk.ac.id - website : www.stmikplk.ac.id

SURAT TUGAS

No.268/STMIK-3.C.2/KP/VIII/2021

Ketua Program Studi Teknik Informatika Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) Palangkaraya menugaskan nama- nama tersebut di bawah ini :

1. Nama : Lili Rusdiana, M.Kom
   NIK : 198707282011007
   Sebagai Pembimbing I Dalam Pembuatan Program
2. Nama : Fenroy Yedithia, S.Kom, M.TI
   NIK : 199208112019102
   Sebagai Pembimbing II Dalam Penulisan Tugas Akhir

Untuk membimbing Tugas Akhir mahasiswa :

Nama	:	Dimas Prayoga
NIM	:	C1855201055
Program Studi	:	TEKNIK INFORMATIKA (55201)
Tanggal Daftar	:	08 Agustus 2021
Judul Tugas Akhir	:	Pengembangan Game 2D Platformer "Virus Must Die" Berbasis Android menggunakan Software Unity

Demikian surat ini dibuat agar dapat dipergunakan sebagaimana mestinya dan dilaksanakan dengan penuh tanggung jawab.

Palangka Raya, 30 Agustus 2021
Ketua Program Studi Teknik Informatika,

Lili Rusdiana, M.Kom.
NIK. 198707282011007

Tembusan :
1. Pembimbing I dan II
2. Mahasiswa yang bersangkutan
3. Arsip

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
(STMIK) PALANGKARAYA
Jl. G. Obos No.114 Telp.0536-3224593, 3225515 Fax.0536-3225515 Palangkaraya
email : humas@stmikplk.ac.id - website : www.stmikplk.ac.id

KARTU KEGIATAN KONSULTASI TUGAS AKHIR

Nama Mahasiswa : Dimas Anggara
NIM : C1855201055
Tanggal Persetujuan Judul : 30 Agustus 2021
Judul Tugas Akhir : Pengembangan Game 2D Platformer "Virus Must Die" Berbasis Android Menggunakan Software Unity

No	Tanggal Konsultasi		Uraian	Tanda Tangan
	Terima	Kembali		
1	7/9 2021	7/9 2021	Pengarahan konsep edukasi untuk topik	
2	24/8 2021	24/8 2021	Pengarahan Pengembangan game, Perbaikan Penulisan	
3	21/9 2021	21/9 2021	Pengarahan metode pengembangan game yang digunakan dan Perbaikan penulisan	
4.	1/10 2021	1/10 2021	Konsul Via whatsapp (online), Perbaikan penulisan	
5	6/10 2021	6/10 2021	Lengkapi penulisan sesuai arahan konsul sebelumnya. Bab 3 fokuskan ke penjelasan terkait penelitian	
6	7/10 2021	7/10 2021	lengkapi penulisan & proposal silakan lanjut ke dorbing 2	
7.	11/10 2021	11/10 2021	Perbaikan bab 2-3	
8.	15/10 2021	15/10 2021	lanjut ke seminar	
9.	08/01 2021	08/01 2021	Konsultasi Bab 4	
10	10/01 2021	10/01 2021	Konsultasi lanjutan Bab 4	
11.	14/01 2021	14/01 2021	Mendemokan rancangan Game dan Review untuk perbaikan.	
12.	14/01 2021	17/01 2021	Demontrasi Program Berserta dgn Gitekan untuk perancangan berserta dgn program	
13	18/01 2021	18/01 2021	Perbaikan penulisan, penyempurnaan dgn program	

No.	Tanggal Konsultasi		Uraian	Tanda Tangan
	Terima	Kembali		
14.	19/1 2021	19/1 2021	ACC Sidang lengkapi berkas	
15.	19/1 2021	19/1 2021	ACC Sidang TA	

Menyetujui :

Dosen Pembimbing I,

Lili [illegible], M.Kom

Dosen Pembimbing II,

Fenroy Yedithia, M.Kom

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
(STMIK) PALANGKARAYA
Jl. G. Obos No.114 Telp.0536-3224593, 3225515 Fax.0536-3225515 Palangka Raya
email : humas@stmikplk.ac.id - website : www.stmikplk.ac.id

SURAT TUGAS PENGUJI TUGAS AKHIR

No. 17/STMIK-3.C.2/KP/I/2022

Ketua Program Studi Teknik Informatika Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) Palangkaraya menugaskan kepada nama- nama berikut :

1. Nama : Veny Cahya Hardita, M.Kom
   NIK : 199504302020002
   Sebagai Ketua
2. Nama : Elok Faiqotul Himmah, S.Si., M.Sc.
   NIK : 198503092009003
   Sebagai Sekretaris
3. Nama : Sulistyowati, S.Kom., M.Cs.
   NIK : 198212162007002
   Sebagai Anggota
4. Nama : Lili Rusdiana, M.Kom
   NIK : 198707282011007
   Sebagai Anggota
5. Nama : Fenroy Yedithia, S.Kom, M.TI
   NIK : 199208112019102
   Sebagai Anggota

Tim Penguji Tugas Akhir Mahasiswa :

Nama	:	Dimas Prayoga
NIM	:	C1855201055
Hari/ Tanggal Ujian	:	Sabtu, 22 Januari 2022
Waktu	:	10.00 WIB
Judul Tugas Akhir	:	Pengembangan Game 2D Platformer "Virus Must Die" Berbasis Android menggunakan Software Unity

Demikian surat ini dibuat agar dapat dipergunakan sebagaimana mestinya dan dilaksanakan dengan penuh tanggung jawab.

Palangka Raya, 18 Januari 2022
Ketua Program Studi Teknik Informatika,

Lili Rusdiana, M.Kom.
NIK. 198707282011007

Tembusan :
1. Dosen Penguji
2. Mahasiswa yang Bersangkutan
3. Arsip

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
(STMIK) PALANGKARAYA
Jl. G. Obos No.114 Telp.0536-3224593, 3225515 Fax.0536-3225515 Palangkaraya
email : humas@stmikplk.ac.id - website : www.stmikplk.ac.id

BERITA ACARA
UJIAN TUGAS AKHIR

Periode (Bulan) : Januari Tahun 2022

1. Hari/Tanggal Ujian : Sabtu / 22
2. Waktu (Jam) : 10.00 WIB sampai dengan 12.00 WIB
3. Nama Mahasiswa : Dimas Prayoga
4. Nomor Induk Mahasiswa : C1855201055
5. Program Studi : Teknik Informatika
6. Tahun Angkatan : 2018
7. Judul Tugas Akhir : Pengembangan Game 2D Platformer "Virus Must Die" Berbasis Android Menggunakan Unity
8. Dosen Penguji :

	Nama	Nilai	Tanda Tangan
1.	Veny Cahya Hardita, M.Kom		( )
2.	Elok Faiqotul Himmah, S.Si., M.Cs		( )
3.	Sulistyaningati, S.Kom., M.Cs.		( )
4.	Lili Rusdiana, M.Kom		( )
5.	Fenroy Yedithia, S.Kom., M.T.		( )

9. Hasil Ujian : **LULUS / TIDAK LULUS** *) NILAI = 81,3
   Dengan Perbaikan/ Tanpa Perbaikan *)
10. Catatan Penting : 1. Lama Perbaikan : ..... hari
    2. Jika lebih dari 1 (satu) bulan dikenakan sanksi berupa denda sebesar Rp. 600.000,- (Enam ratus ribu rupiah) per bulan dari tanggal ujian
    3. Jika lebih dari 3 (tiga) bulan dari tanggal ujian maka hasil ujian dibatalkan dan wajib mengajukan judul dan pembimbing baru

Palangka Raya, 22 Januari 2022

Mengetahui :
Ketua Program Studi Teknik Informatika,

Lili Rusdiana, M.Kom.
NIK. 198707282011007

Ketua Penguji,

Veny Cahya Hardita, M.Kom
NIK. 1955043020200002

Tembusan:
1. Arsip Prodi Teknik Informatika
2. Mahasiswa yang bersangkutan

Dibawa saat konsultasi perbaikan dengan dosen penguji
**) Coret yang tidak perlu*

Lampiran 5. Listing Program Game 2D Platformer " Virus Must Die"

Coding HealthBar Player

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class HealthBar : MonoBehaviour
{
    public Health playerHealth;
    public Image totalHealthBar;
    public Image currentHealthBar;
    // Start is called before the first frame update
    void Start()
    {
        totalHealthBar.fillAmount = playerHealth.currentHealth /10;
    }
    void Update()
    {
        currentHealthBar.fillAmount = playerHealth.currentHealth / 10;
    }
}
```

Coding Health Player

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Health : MonoBehaviour
{
    [SerializeField] private float startingHealth;
    public float currentHealth {get; private set;
    private void Awake()
    {      currentHealth = startingHealth;  }

    public void TakeDamage(float damage) {
        currentHealth = Mathf.Clamp(currentHealth - damage, 0, startingHealth);

        if (currentHealth > 0)
        { }
        else
        {
            Destroy(gameObject);
            SceneManager.LoadScene("Lose_scene");
        } }}
```

Coding Enemy Patrol

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Patrol : MonoBehaviour
{
  [HideInInspector]
  public bool mustPatrol;
  public bool mustTurn;
  public Transform sightStart, sightEnd;
  public float walkSpeed;
  public Rigidbody2D rb;
  public Transform groundCheckPos;
  public LayerMask groundLayer;
  private bool collision = false;

  void Start()
  {
    mustPatrol = true;
```

```
 }
 void Update()
 {
      collision =
Physics2D.Linecast(sightStart.position,
sightEnd.position, 1 <<
LayerMask.NameToLayer("Wall"));


      Debug.DrawLine (sightStart.position,
sightEnd.position, Color.green);


     if (collision)
     {
      flip();
     }
     if (mustPatrol)
     { patrol();}
 }
 private void FixedUpdate() {
   if (mustPatrol){
     mustTurn
= !Physics2D.OverlapCircle(groundCheckPo
s.position, 0.1f, groundLayer);
    } }
 void patrol(){
   if (mustTurn)
   {
     flip();
   }
   rb.velocity = new Vector2(walkSpeed *
Time.fixedDeltaTime, rb.velocity.y);
 }
 void flip(){
   mustPatrol = false;
   transform.localScale = new
Vector3((transform.localScale.x == 0.643f) ?
-0.643f : 0.643f,0.643f,0.643f);
   walkSpeed *= -1;
   mustPatrol = true;
 }
}
```

Coding EnemyFollowPlayer

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class EnemyFollowPlayer :
MonoBehaviour
{
   public float speed;
   public float lineOfSite;
   private Transform player = null;
   public Animator animator;
   void Start()
   {
      player =
GameObject.FindGameObjectWithTag("Pla
yer").transform;
      animator.SetBool("Agro", false);
   }
   void Update()
   {
      float distanceFormPlayer =
Vector2.Distance(player.position,
transform.position);
      if (distanceFormPlayer < lineOfSite)
      {
         transform.position =
Vector2.MoveTowards(this.transform.positio
n, player.position, speed * Time.deltaTime);
         animator.SetBool("Agro", true);
      }else{
         animator.SetBool("Agro", false);
      }
   }
   private void OnDrawGizmosSelected()
   {
      Gizmos.color = Color.green;
      Gizmos.DrawWireSphere(transform.po
sition, lineOfSite);
   }
```

```
}
```

Coding Enemy

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Enemy : MonoBehaviour
{
    public float damage;
    public int health = 100;


    public void TakeDamage(int damage){
        health -= damage;


        if (health <= 0)
        {
            Die();
        }
    }
    void Die()
    {
        Destroy(gameObject);
}


    private void OnTriggerEnter2D(Collider2D
collision) {
        if (collision.tag == "Player")
        {
            collision.GetComponent<Health>().T
akeDamage(damage);
        }
        if(collision.tag == "Player")
        {
            var player =
collision.GetComponent<PLAYERMOVEME
NT>();
            player.knockbackCount =
player.knockbackLength;
            if(collision.transform.position.x <
transform.position.x){
                player.knockFromRight = true;
            }else {
                player.knockFromRight = false;
            }}}}
```

Coding PLAYERMOVEMENT

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;


public class PLAYERMOVEMENT :
MonoBehaviour
{
    Rigidbody2D rb;


    public float Speed;
    public Joystick mj;
    public float jumpspeed;
    public CharacterController2D controller;
    public Animator animator;
    public Transform joystick;


    public float knockback;
    public float knockbackLength;
    public float knockbackCount;
    public bool knockFromRight;


    protected Player_JumpButton jumpButton;
    bool jumped;


    bool jump = false;
    float horizontalmove;
    float x;
    bool crouch = false;


    private void Start()
```

```
    {
        rb = GetComponent<Rigidbody2D>();
        jumpButton = FindObjectOfType<Player_JumpButton>();
        joystick.gameObject.SetActive(true);
    }

    private void Update()
    {
        if(Dialog_Manager.isActive == true)
        {
            joystick.gameObject.SetActive(false);
            Speed = 0f;
        }

        if (knockbackCount <= 0)
        {
            if (mj.Horizontal >= .2f)
            {
                horizontalmove = Speed;
            }else if (mj.Horizontal <= -.2f)
            {
                horizontalmove = -Speed;
            }else
            {
                horizontalmove =0f;
            }

        } else {
            if(knockFromRight)
            {
                rb.velocity = new Vector2(-knockback, knockback);
            }

            if (!knockFromRight)
            {
                rb.velocity = new Vector2(knockback, knockback);
            }
            knockbackCount -= Time.deltaTime;
        }
        animator.SetFloat("Speed", Mathf.Abs(horizontalmove));
        if (!jumped && jumpButton.Pressed)
        {
            jump = true;
            animator.SetBool("Lompat", true);
        }
        if (jumped && jumpButton.Pressed)
        {
            jump = false;
        }
    }
    public void Onlanding()
    {
        animator.SetBool("Lompat", false);
    }
     private void OnLevelWasLoasded(int level)
    {
        startPos();
    }
     void startPos()
     {
        transform.position = GameObject.FindWithTag("StartPos").transform.position;
    }
    private void FixedUpdate()
    {
        controller.Move(horizontalmove * Time.fixedDeltaTime, crouch, jump);
        jump = false;
}}
```

Coding CharacterController2D

```
using UnityEngine;

using UnityEngine.Events;\

public class CharacterController2D :
MonoBehaviour

{

    [SerializeField] private float
m_JumpForce;                        // Amount of force
added when the player jumps.

    [Range(0, 1)] [SerializeField] private float
m_CrouchSpeed = .36f;          // Amount of
maxSpeed applied to crouching movement. 1 =
100%

    [Range(0, .3f)] [SerializeField] private float
m_MovementSmoothing = .05f;  // How much to
smooth out the movement

    [SerializeField] private bool m_AirControl =
false;                      // Whether or not a player can
steer while jumping;

    [SerializeField] private LayerMask
m_WhatIsGround;                     // A mask
determining what is ground to the character

    [SerializeField] private Transform
m_GroundCheck;                      // A position
marking where to check if the player is grounded.

    [SerializeField] private Transform
m_CeilingCheck;                     // A position
marking where to check for ceilings

    [SerializeField] private Collider2D
m_CrouchDisableCollider;                // A collider
that will be disabled when crouching



    public float k_GroundedRadius = .2f; // Radius
of the overlap circle to determine if grounded

    public bool m_Grounded = false;            //
Whether or not the player is grounded.

    const float k_CeilingRadius = .2f; // Radius of
the overlap circle to determine if the player can
stand up

    private Rigidbody2D m_Rigidbody2D;

    private bool m_FacingRight = true;  // For
determining which way the player is currently
facing.

    private Vector3 m_Velocity = Vector3.zero;



    bool wasGrounded;



    [Header("Events")]

    [Space]

    public UnityEvent OnLandEvent;



    [System.Serializable]

    public class BoolEvent : UnityEvent<bool> { }



    public BoolEvent OnCrouchEvent;

    private bool m_wasCrouching = false;



    private void Awake()

    {

        m_Rigidbody2D =
GetComponent<Rigidbody2D>();



        if (OnLandEvent == null)

            OnLandEvent = new UnityEvent();



        if (OnCrouchEvent == null)

            OnCrouchEvent = new BoolEvent();

    }



    private void FixedUpdate()

    {

        m_Grounded =
Physics2D.Raycast(transform.position,
Vector2.down, k_GroundedRadius,
m_WhatIsGround);



        if (m_Grounded)

            {

                m_Grounded = true;

                if (!wasGrounded)

                    OnLandEvent.Invoke();

            }



    }



    public void Move(float move, bool crouch, bool
jump)

    {
```

```
        // If crouching, check to see if the character
can stand up

        if (crouch)

        {

            // If the character has a ceiling preventing
them from standing up, keep them crouching

            if
(Physics2D.OverlapCircle(m_CeilingCheck.positio
n, k_CeilingRadius, m_WhatIsGround));

            {

                crouch = true;

            }

        }

        if (m_Grounded || m_AirControl)

        {

            if (crouch)

            {

                if (!m_wasCrouching)

                {

                    m_wasCrouching = true;

                    OnCrouchEvent.Invoke(true);

                }

                move *= m_CrouchSpeed;



                if (m_CrouchDisableCollider != null)

                    m_CrouchDisableCollider.enabled =
false;

            }

            else

            {

                if (m_CrouchDisableCollider != null)

                    m_CrouchDisableCollider.enabled =
true;



                if (m_wasCrouching)

                {

                    m_wasCrouching = false;

                    OnCrouchEvent.Invoke(false);

                }

            }

            Vector3 targetVelocity = new
Vector2(move * 10f, m_Rigidbody2D.velocity.y);

            m_Rigidbody2D.velocity =
Vector3.SmoothDamp(m_Rigidbody2D.velocity,
targetVelocity, ref m_Velocity,
                m_MovementSmoothing);

            // If the input is moving the player right and
the player is facing left...

            if (move > 0 && !m_FacingRight)

            {

                Flip();

            }

            // Otherwise if the input is moving the
player left and the player is facing right...

            else if (move < 0 && m_FacingRight)

            {

                Flip();

            }}

        if (m_Grounded && jump)

        {

            m_Grounded = false;

            m_Rigidbody2D.velocity = new
Vector2(m_Rigidbody2D.velocity.x, 0);

            m_Rigidbody2D.AddForce(Vector2.up *
m_JumpForce, ForceMode2D.Impulse);

        }}

    private void Flip()

    {

        // Switch the way the player is labelled as
facing.

        m_FacingRight = !m_FacingRight;

        transform.Rotate (0f, 180f, 0f);

    }

    private void OnDrawGizmos(){

        Gizmos.color = Color.red;

        Gizmos.DrawLine(transform.position,
transform.position + Vector3.down *
k_GroundedRadius);

    }

}
```