**Rancang Bangun Game 2D Shooter Menggunakan**

**Metode Finite State Machine**

**Agung Nopriansyah1, Indra Kanedi2, Prahasti3**

1 Jurusan Informatika, Fakultas Ilmu Komputer, Universitas Dehasen Bengkulu

Perum Grand Korpri Blok E No.10 Rt.12 Rw.3 Bentiring, Muara Bangkahulu, Bengkulu 38126

Email: [meagung62@gmail.com](mailto:meagung62@gmail.com)1

2,3 Dosen Program Studi Teknik Komputer Fakultas Ilmu Komputer Universitas Dehasen Bengkulu

Jl. Meranti Raya No. 32 Kota Bengkulu 38228

Email: [indrakanedi12@gmail.com](mailto:indrakanedi12@gmail.com)2 [prahasti.mona82@gmail.com](mailto:prahasti.mona82@gmail.com)3

**ABSTRAK**

*Game 2D Shooter* ini merupakan permainan video petualangan dimana pemain memainkan karakter seorang *Survivor*. *Game* ini berfokus pada petualangan *Survivor* membunuh semua monster yang ada agar bisa melanjutkan ke *level* selanjutnya hingga tamat. *Game* ini menggunakan *Game Engine Unity*, dengan menerapkan *Finite State Machine*. *Finite State Machine* digunakan pada *NPC (Non-Playable Character)* yaitu karakter yang muncul pada permainana sebagai monster dalam *game* tersebut seperti karater *Enemy* dan *Boss*. Dalam pengujian yang telah dilakukan hasil dari *FSM (Finite State Machine)* pada *game 2D Shooter* ini sebagian responden dengan rata-rata 32% menjawab Sangat Bagus, 41% menjawab Bagus, 24% menjawab Biasa Saja, 3% menjawab Buruk.

Kata Kunci: Game, Unity, Game Engine, 2D Shooter, Finite State Machine

*ABSTRACT*

*This 2D Shooter game is an adventure video game where the player plays the character of a Survivor. This game focuses on the Survivor's adventure to kill all the monsters in order to continue to the next level until the end. This game uses the Unity game engine, by implementing the Finite State Machine. Finite State Machine is used on NPCs (Non-Playable Characters), namely characters that appear in the game as monsters in the game such as Enemy and Boss characters. In the tests that have been carried out the results of the FSM (Finite State Machine) in this 2D Shooter game, most of the respondents with an average of 32% answered Very Good, 41% answered Good, 24% answered Moderate, 3% answered Poor.*

Keywords: *Game, Unity, Game Engine, 2D Shooter, Finite State Machine*

1. Pendahuluan

Permainan Video atau lebih dikenal dengan *Video Game* biasanya identik dengan permainan yang dimainkan khalayak ramai dikomputer atau di perangkat elektronik lainnya. Sejak dulu dunia Game salah satu yang paling diminati, di jaman sekarang Game sudah diminati dari banyak kalangan pria dan wanita dari berbagai umur sudah banyak bermunculan. Game yang dimainkan pun sudah banyak jenisnya mulai dari Game Portable, Game PC ataupun Game Mobile. Seiring dengan perkembangan waktu dan teknologi, Industri Game pun ikut berkembang dengan pesat, mulai dari berkembangnya Hardware yang cukup berat untuk mengangkat game yang berat pula, Tampilan grafis yang memukau yang hampir menyerupai dunia nyata.

Perkembangan yang pesat ini diikuti pula dengan banyaknya orang yang membuat game atau Game Developer secara independen bermunculan. Seiring dengan perkembangan teknologi dan zaman semakin banyak pula bermunculan insan kreatif. Di masa sekarang Game telah menjadi salah satu lahan bisnis yang menjanjikan karena dizaman sekarang perkembangannya juga pesat dan banyak diminati. Teknologi di bidang Game development juga berkembang pesat dengan munculnya berbagai engine yang mempermudah para developer untuk membuat game yang sesuai keinginannya. Salah satunya Unity, Unity adalah game editor berbasis C# yang dikembangkan oleh Unity Technologies yang memberikan kebebasan kepada user untuk membuat game berbasis 2D atau 3D. Karena Unity berbasis bahasa pemrograman C# maka diperlukan pemahaman tentang Coding.

**1.1 *Game***

*Game* merupakan kalimat bahasa inggris yang memiliki arti dasar “Permainan”. Permainan dalam hal ini merujuk pada pengertian “Kelincahan Intelektual” *(intellectual playability). Game* bisa diartikan sebagai arena keputusan dan aksi pemainnya. Ada target-target dan misi untuk dicapai pemainnya. Kelincahan intelektual pada tingkat tertentu merupakan ukuran sejauh mana *game* itu menarik untuk dimainkan secara maksimal.

**1.2 Game Engine**

*Game Engine* adalah sebuah perangkat lunak *(software)* yang dirancang untuk pembuatan dan pengembangan suatu video *game*. *Game engine* memberikan kemudahan dalam menciptakan konsep sebuah *game* yang akan di buat. Mulai dari sistem *rendering*, *physics*, arsitektur suara *scripting, Artificial Intelligence,* dan bahkan *sistem networking.* *Game engine* bisa disebut sebagai jiwa aspek dari sebuah *game*

**1.3 Unity**

*Unity Technologies* berdiri tahun 2004 oleh David Helgason, Nicholas Francis dan Joachim Ante. *Game Engine* ini dibangun atas dasar kepedulian mereka terhadap *indie developer* yang tidak bisa membeli *game engine* karena terlalu mahal. Perusahaan ini berfokus untuk membuat sebuah perangkat lunak yang bisa digunakan oleh semua orang, khususnya untuk membangun sebuah *game.* Di tahun 2009, *Unity* diluncurkan secara gratis dan pada bulan Agustus 2012, *Unity* mencapai popularitas tertinggi dengan lebih dari 1 juta *developer* terdaftar di seluruh dunia.

Menurut Roedavan (2018:4) *Unity* memungkinkan seseorang untuk membuat sebuah *game 3D* dengan mudah dan cepat. Secara *default,* *Unity* telah diatur untuk pembuatan *game bergenre* *First Person Shooter (FPS)*, namun *Unity* juga bisa digunakan untuk membuat game bergenre *Role Playing Game (RPG)*, dan ­­*Real-Time Strategy (RTS)*. Selain itu, *Unity* merupakan sebuah *engine multiplatform* yang memungkinkan *game* yang anda bangun dipublish untuk berbagai platform seperti *Windows, Mac, Android, iOS, PS3* dan juga *Wii.*

*Unity* merupakan suatu aplikasi yang digunakan

untuk mengembangkan game multi platform yang

didesain untuk mudah digunakan. *Unity* itu bagus

dan penuh perpaduan dengan aplikasi yang

profesional. Editor pada *Unity* dibuat degan user interface yang sederhana. (Toni & Anton, 2017)

**1.4 Finite State Machine**

Menurut Asmiatun dan Putri (2017:74) *Finite State Machine* adalah sebuah perangkat atau model yang memiliki beberapa jumlah kondisi yang terbatas pada waktu tertentu dan dapat beroperasi pada masukan baik melalui transisi dari salah satu kondisi ke kondisi yang lain sehingga menyebabkan output atau tindakan untuk memutuskan sesuatu. *Finite State Machine* hanya dapat digunakan untuk satu kondisi pada setiap peristiwa.

**1.5 C# (C-Sharp)**

Menurut Nugroho *C#* atau *C-Sharp* adalah bahasa pemrograman yang diciptakan oleh *Microsoft* (dikembangkan oleh Anders Hjelsberg yang telah meciptakan berbagai macam bahasa pemrograman termasuk *Borland Turbo C++* dan *Borland Delphi*). Bahasa C# juta telah distandarisasi secara internasional oleh ECMA. (Arif, Dini, dan Marwanto, 2013)

*C#* merupakan salah satu aplikasi yang memili kemampuan dalam penguatan *Framework.NET. C#* dibuat sejalan dengan perkembangan *Framework.* *NET, C#* sendiri dikembangkan oleh *Microsoft.* Dalam penerapannya *C-Sharp (C#)* menjanjikan produktifitas, fleksibilitas serta kemudahan yang ada dari aplikasi sebelumnya yaitu *Visual Basic*, *Java* dan *C++. C#* mengadopsi kemampuan dari peggabungan aplikasi sebelumnya (mempelajari-c) *Microsoft* membuat *C#* seiring dengan pembuatan *Framework.NET.* *Chief Architect* dalam pembuatan *C#* adalah Anders Hejlsberg yang sebelumnya berperan dalam pembuatan *Borland Delphi* dan *Turbo Pascal. C#* menjanjikan produktifitas dan kemudahan yang ada di *Visual Basic* dengan kemampuan dan fleksibilitas yang ada di *C/C++.* Menurut spesifikasi bahasanya, *“C# (pronounced “C Sharp”) is a simple, modern, object oriented, and type-safe programming language. It will immediately be familiar to C and C++ programmers. C# combines the high productivity of Rapid Application Development (RAD) languages and the raw power of C++.”.* Untuk mencapai produktifitas tinggi ini konsepkonsep sulit C++ disederhanakan dan fitur- fitur baru ditambahkan. Hal ini mungkin terasa mirip dengan Java, karena itulah C# bisa dianggap sebagai sepupu Java [5]. (Yahya & Amri, 2018)

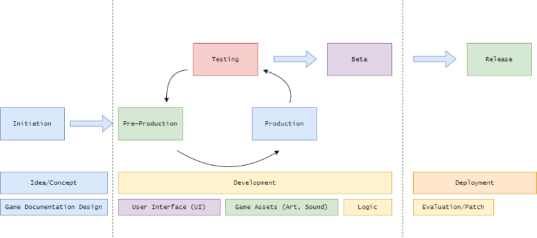
1. **Metode Penelitian**

Pada penelitian ini metode yg digunakan adalah metode *Game Development Life Cycle (GDLC)*. Menurut Roedavan (2018) *Game Development Life Cycle (GDLC)* merupakan tahapan yang dilaksanakan secara sistematis untuk membangun game (*video game*) yang terdiri dari 6 fase yaitu *Initiation, Pre-Production, Production, Testing* (pengujian), *Beta Release* dan *Full Release*. Rangkaian *GDLC* biasanya disederhanakan menjadi 3 fase yaitu *Idea/Concept, Development* dan *Deployment.*

***Idea/Concept***, Pengembang game akan membuat ide dasar terkait latar belakang pembuatan sebuah *game*, *main story*, spesifikasi karakter, genre *game*, jenis *art* yang digunakan, *platform* yang akan dituju, dimensi *game* yang akan dibuat sampai monetisasi game jika sasarannya untuk diperjual belikan. Biasanya, *Output* dari fase ini *adalah Game Document Design (GDD)* yang akan menjadi acuan utama dalam proses pengembangan selanjutnya.

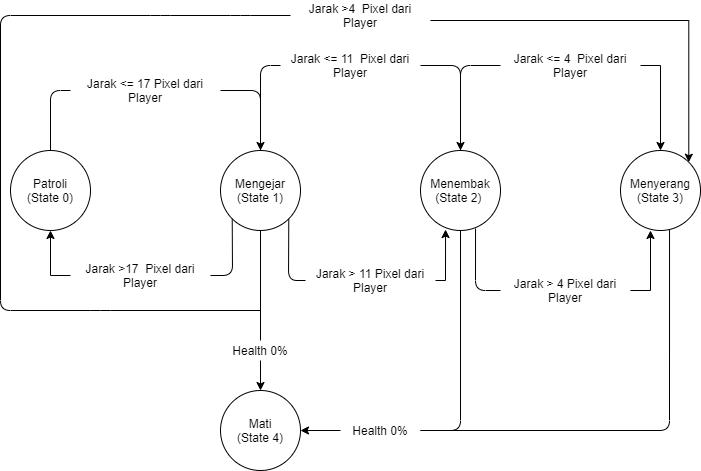
***Development*,** Fase ini berfokus pada penetapan jadwal pengembangan, penentuan tim yang terlibat serta tenggat waktu yang akan digunakan dalam fase pembangunan yang sudah memasukkan unsur pengujian dan penetapan tanggal rilis beta.

***Deployment*** lebih diarahkan pada teknis publikasi *game* pada *platform* yang dituju, termasuk skema persiapan patch jika *game* yang dihasilkan memiliki bug atau mengalami gangguan teknis terkait perbedaan *environment platform* yang dituju.



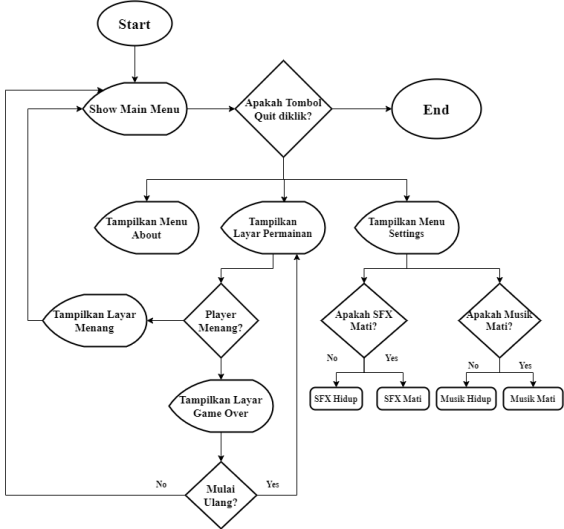
Gambar 1. Game Development Life Cycle (GDLC)

**2.1 Rancangan Diagram Finite State**



Gambar 2. Rancangan *Finite State Machine*

**2.2 Flowchart Game**

****

Gambar 3. Tampilan *Flowchart*

1. **Hasil dan Pembahasan**

*2D Shooter Platformer* adalah *game* yang bergenre 2D Action Shooter yang dirancang menggunakan aplikasi Unity serta menggunakan metodologi Finite State Machine (FSM).

**3.1 Tampilan Menu Utama**

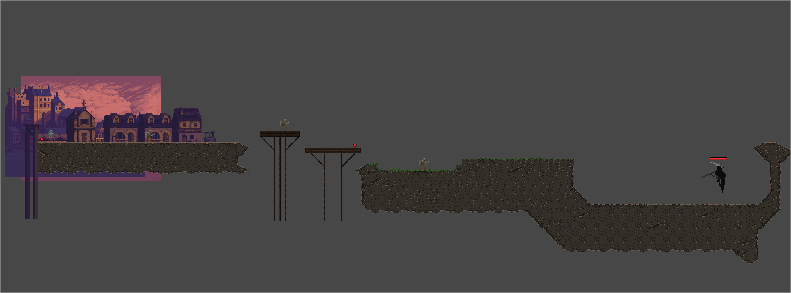
Bagian ini merupakan bagian pertama yang muncul saat game baru dimulai. Pada tampilan in terdapat tombol seperti Mulai, Pengaturan, Tentang dan Keluar.



Gambar 4. Tampilan Menu Utama

**3.2 Tampilan Level 1**

Tampilan *level 1* ini terdapat musuh tengkorak yang mana jika kita berada didekat nya maka ia akan langsung mengejar player dan jika berada terlalu dekat monster tengkorak akan menyerang player, gambar berikut adalah tampilan keseluruhan map pada *game engine Unity* bukan tampilan dalam game.



Gambar 5. Tampilan *map* *Level 1*

**3.3 Tampilan *Boss* *Level 1***

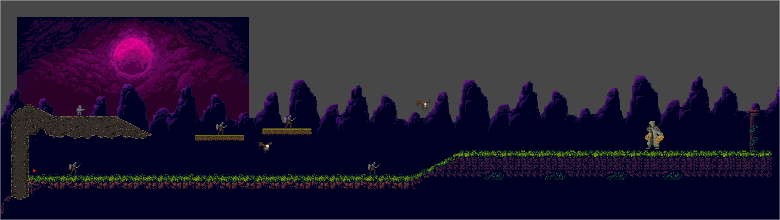
Tampilan *boss level* *1* ini *boss* bisa mengejar *player* pada jangkauan tertentu, *boss* ini hanya bisa menembak ke arah *player.*



Gambar 6. Tampilan *Boss Level 1*

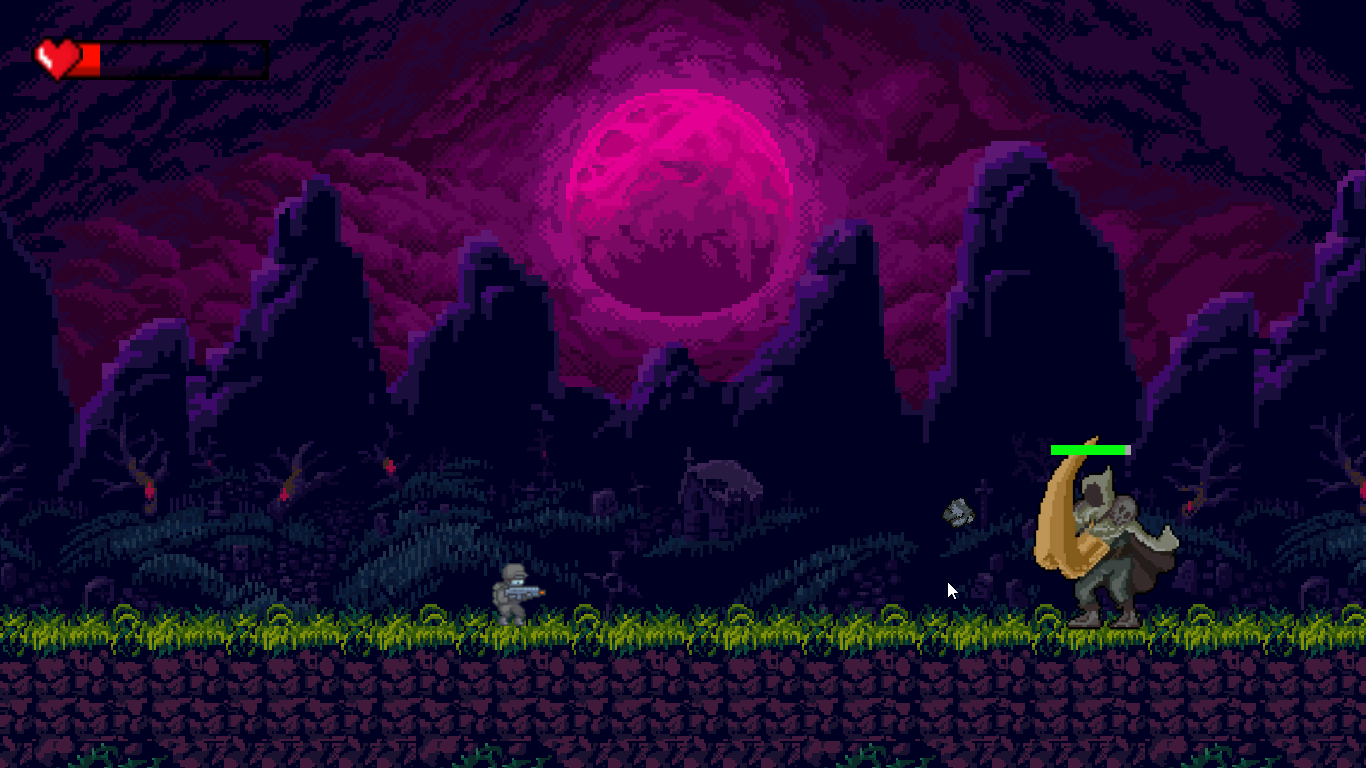
**3.4 Tampilan *Level 2***

Di *level 2* ini ada monster yang berbeda yaitu monster yang bisa terbang mengikuti *player,* jika berada dekat dengan player maka monster ini akan menembak ke arah *player*. Selain monster yang terbang ada monster tengkorak yang sama seperti di *level 1.*

Gambar 7. Tampilan *Level 2*

**3.5 Tampilan *Boss Level 2***

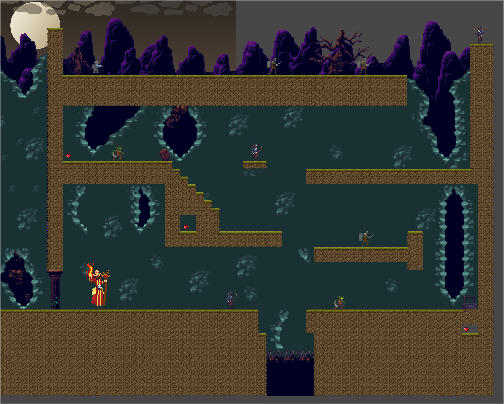
*Boss* pada *level 2* ini tidak mengejar *player* tapi jika berada jarak tertentu *boss* ini akan melempar batu kearah *player* dengan kencang, jadi *player* harus menghindari setiap batu yg dilempar.



Gambar 8. Tampilan *Boss level 2*

**3.6 Tampilan *Level 3***

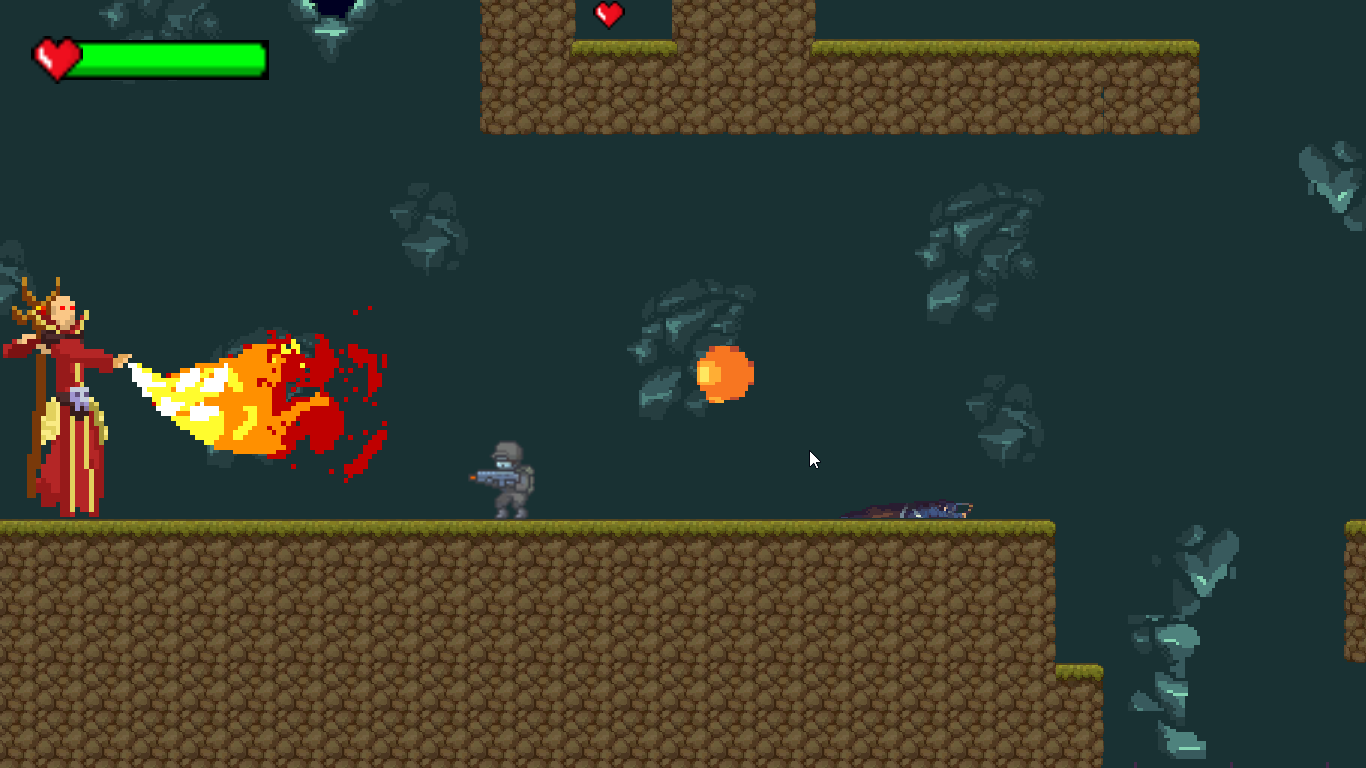
Pada *level 3* ini terdapat 2 monster baru pemanah dan *goblin*, *Goblin* prinsipnya sama seperti dengan tengkorak hanya gerakannya lebih cepat. Pemanah tidak bisa mengejar pemain tapi jika berada pada jarak tertentu pemanah akan menembak busur ke arah *player.*



Gambar 9. Tampilan *Level 3*

**3.7 Tampilan *Boss Level 3***

*Boss level* terakhir ini prinsipnya sama dengan *boss* sebelumnya tapi *boss level* terakhir ini jika player berada pada jarak tertentu dengan *boss* maka *boss* ini akan menembakkan bola api kearah *player.*



Gambar 10. Tampilan *Boss Level 3*

**3.8 Tampilan Jeda**

Tampilan ini muncul jika pemain menekan tombol *ESC* pada *keyboard,* ketika muncul permainan akan berhenti dan akan muncul pilihan untuk kembali ke permainan, pengaturan dan keluar ke menu. Jika kembali ke permainan maka game akan lanjut bergerak lagi seperti biasa, jika tombol pengaturan dipilih akan muncul pilihan mematikan suara, dan jika tombol keluar ke menu maka akan kembali ke tampilan menu awal *game.*



Gambar 11. Tampilan Jeda

**3.9 Tampilan Jika *Player* Mati**

Berikut adalah tampilan yang akan muncul jika *player* mati diserang oleh musuh atau jatuh ke lubang, ketika tampilan ini muncul pemain diberi dua pilihan yaitu memulai ulang permainan atau kembali ke menu.



Gambar 12. Tampilan Jika *Player* Mati

**3.10 Tampilan Menu Pengaturan**

Pada tampilan tombol pengaturan ini berisi tentang pengaturan suara pada *game*, menu ini mengatur ada atau tidaknya suara dalam *game* seperti musik dan efek suara tembakan.



Gambar 13. Tampilan menu Pengaturan

**3.11 Tampilan Menu Tentang**

Pada tampilan ini berisi tentang petunjuk kontrol permainan dan data diri pembuat *game.*



Gambar 14. Tampilan Menu Tentang

**3.12 Pengujian Menggunakan Black Box Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Percobaan yang dilakukan** | **Hasil** | **Keterangan** |
| 1 |  | Memulai permainan | Valid |
| 2 |  | Menampilkan keterangan tentang *game* dan biodata pembuat | Valid |
| 3 |  | Menampilkan pengaturan suara | Valid |
| 4 |  | Efek suara dan musik akan mati | Valid |
| 5 |  | Keluar dari permainan | Valid |
| 6 | Jika tombol spasi atau panah atas ditekan | Karakter akan melompat | Valid |
| 7 | Jika tombol A atau panah kiri ditekan | Karakter akan bergerak ke kiri | Valid |
| 8 | Jika tombol D atau panah kanan ditekan | Karakter akan bergerak ke kanan | Valid |
| 9 | Jika mengklik kanan pada mouse atau *ctrl* kiri | Karakter akan menembakkan peluru | Valid |
| 10 | Jika tombol *ESC* ditekan | Memunculkan menu berhenti | Valid |
| 11 | Jika musuh menyerang karakter | Darah player akan berkurang | Valid |
| 12 | Jka karakter menyerang musuh | Darah lawan berkurang atau mati | Valid |
| 13 | Jika player mati | Mengulang ke awal *level* | Valid |
| 14 | Jika player berhasil mengalahkan *boss* disetiap *level* | Menuju ke *level* berikutnya | Valid |

Tabel 1. Pengujian Sistem Menggunakan Black Box

**3.13 Hasil Pengujian Game Kepada User**

Pengujian ini dilaksanakan sebagai bentuk untuk mengetahui apakah *2D Game Shooter Platformer* sudah berjalan dengan baik. Pengujian ini dilakukan terhadap 15 orang responden dengan mengisi kuesioner pada *Google Form* setelah responden mencoba *game.*

Kuesioner yang sudah diisi dapat dilihat pada lampiran. Dari hasil kuesioner akan dilakukan perhitungan dengan interval nilai 0% - 24% (Tidak Layak), 25% - 49% (Cukup Layak), 50% - 74% (Layak), 75% - 100% (Sangat Layak).

Keterangan :

1 = Sangat Bagus

2 = Bagus

3 = Biasa Saja

4 = Buruk

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Pertanyaan** | **Penilaian** | | | |
| **1** | **2** | **3** | **4** |
| 1 | Bagaimana pendapat anda mengenai *game 2D Shooter Platformer?* | 5 | 8 | 2 | 0 |
| 2 | Menurut anda bagaimana tampilan menu dari *game 2D Shooter Platformer* | 5 | 8 | 2 | 0 |
| 3 | Bagaimana menurut anda konsep *gameplay* *game 2D Shooter Platformer* | 6 | 6 | 3 | 0 |
| 4 | Bagaimana menurut anda kontrol dari game ini? | 5 | 4 | 5 | 1 |
| 5 | Bagaimana menurut anda tentang efek suara dari *game 2D Shooter Platformer?* | 3 | 5 | 6 | 1 |
| Rata-Rata | | 32% | 41% | 24% | 3% |

Tabel 2. Hasil Kuesioner Pengujian Game

Berdasarkan tabel 2 dapat disimpulkan bahwa *game 2D Shooter Platform Shooter* ini layak dimainkan dinyatakan dengan nilai rata-rata 32% menjawab Sangat Bagus. 41% menjawab Bagus, 24% menjawab Biasa Saja, 3% menjawab Buruk, maka dari itu hasil pengujian terhadap 15 responden berada pada persenan tertinggi yaitu 41% yang artinya berada pada interval Layak.

**Kesimpulan**

Berdasarkan hasil perancangan serta pengujian game *2D Shooter Platformer* dapat disimpulkan bahwa *Finite State Machine (FSM)* berhasil diterapkan pada *game* yang bergenre *Action Shooter* ini untuk kecerdasan musuh pada *game* ini dan berdasarkan pengujian *black box* berjalan dengan lancar sesuai dengan rancangan dan berdasarkan kuesioner kepada 15 orang bahwa game ini layak untuk dimainkan dan musuh pada *game* ini berfungsi dengan baik.

**Daftar Pustaka**

1. Ardyanto, Toni & Respati Pamungkas, Anton (2017). Pembuatan Game 2D Petualangan Hanoman Berbasis Android. JURNAL ILMIAH GO INFOTECH Volume 23 No. 2, Desember 2017, 14-17.
2. Asmiatun, S., & Putri, A. N. (2017). *Belajar Membuat Game 2D dan 3D Menggunakan Unity*. Deepublish.
3. Jauhari, M. A., Hamidin, D., & Rahmatuloh, M. (2017). Komparasi Stabilitas Eksekusi Kode Bahas Pemrograman .Net C# Versi 4.0.3019 Dengan Google Golang Versi 1.4.2 Menggunakan Algoritma Bubble Sort dan Insertion Sort . Jurnal Teknik Informatika, 9(1), 13–20.
4. Kurniawan, R. (2012). Apa Itu Game Engine. <https://rickykurn.wordpress.com/2012/03/08/apa-itu-game-engine/>
5. Roedavan, R. (2018). *Unity - Tutorial Game Engine Edisi Revisi Kedua*. Penerbit Informatika.
6. Yahya & Muliawan Nur, Amri (2018). Pengaruh Aplikasi C# dalam Proses Perhitungan Numerik Terhadap Solusi Persamaan Non Linier. Infotek : Jurnal Informatika dan Teknologi Vol. 1 No. 2, Juli 2018, 79-87.