SKRIPSI

PENERAPAN METODE *FINITE STATE MACHINE* DALAM PERANCANGAN PERILAKU MUSUH PADA GAME 2D *HORROR GIRL'S NIGHTMARE*

Disusun Oleh :

Niniek Mardiyani

14.18.153

PROGRAM STUDI TEKNIK INFORMATIKA S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

2018

Abstrak

Permainan atau sering disebut game merupakan suatu sarana hiburan yang diminati dan dimainkan oleh banyak orang. Seiring perkembangan, musuh-musuh dalam dunia game juga semakin cerdas dan unik, hal ini dimaksud agar membuat musuh lebih menarik. Oleh karena itu dengan perkembangan teknologi yang ada maka kemampuan musuh tersebut dapat diubah dengan kecerdasan buatan.

Game Girl's Nightmare ini menawarkan permainan dengan dengan tema horror yang dikemas dengan kontrol adventure menggunakan tools Unity 3D dan bahasa pemrograman c#. Karakter musuh yang ada pada game ini memiliki tingkat serangan yang berbeda. Oleh karena itu diterapkannya metode Finite State Machine, dimana metode berguna dalam perancangan perilaku musuh dalam setiap level. Dalam game ini terdapat 3 level dan beberapa musuh yaitu musuh kecil dan musuh utama. Perilaku yang dilakukan musuh yaitu melihat, mengejar, dan menyerang pemain dengan menyentuhnya.

Untuk pengujian pada game ini yaitu pengujian fungsional. Dari beberapa controller yaitu space, double space, control, dan shift serta pengujian metode pada game ini mendapatkan hasil yang sesuai dan berjalan 100%.

Keyword : Finite State Machine, Game Girl's Nightmare, Game 2D Horror

KATA PENGANTAR

Puji syukur Alhamdulillah penulis panjatkan kehadirat allah swt atas berkat, rahmat, taufik dan hidayah-Nya, penyusunan skripsi yang berjudul "(Penerapan Metode *Finite State Machine* dalam Perancangan Perilaku Musuh pada Game 2D *Horror Girl's Nightmare*)" dapat diselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan Nabi besar Muhammad SAW beserta keluarga, sahabat, dan pengikut beliau hingga akhir zaman.

Penulis menyadari bahwa dalam proses penulisan skripsi ini banyak mengalami kendala, namun berkat bantuan, bimbingan, kerjasama dari berbagai pihak dan berkah dari Allah SWT sehingga kendala-kendala yang dihadapi tersebut dapat diatasi. Untuk itu penulis menyampaikan ucapan terima kasih dan penghargaan kepada Bapak dan Ibu yang senantiasa mendoakan, memberikan bantuan moril, materi dan nasehat selama penulis menjalani pendidikan. Selanjutnya ucapan terimakasin penulis sampaikan pula kepada :

1. Dr. Ir. Lalu Mulyadi, MTA. selaku Rektor Institut Teknologi Nasional Malang.
2. Dr. Ir. F. Yudi Limpraptono, MT,selaku Dekan Fakultas Teknologi Industrim Insitut Teknologi Nasional Malang.
3. Joseph Dedy Irawan, ST, MT, selaku Ketua Program Studi Teknik Informatika, Institut Teknologi Nasional Malang.
4. Suryo Adi Wibowo, ST, MT, selaku Sekertaris Program Studi Teknik Informatika, Institut Teknologi Nasional Malang.
5. Ali Mahmudi, B.Eng, PhD, selaku Dosen Pembimbing I yang selalu memberikan bimbingan dan masukan.
6. Hani Zulfia Zahro' S.Kom, M.Kom, selaku Dosen Pembimbing II yang selalu memberikan bimbingan dan Masukan.
7. Semua dosen Program Studi Teknik Informatika yang telah membantu dalam penulisan dan masukan.
8. Semua teman teman berbagai angkatan yang telah memebrikan doa dan dukungan dalam penyelesaian skripsi.

Dengan segala kerendahan hati, penulis menyadari masih banyak terdapat kekurangan-kekurangan, sehingga penulis mengharapkan adanya saran dan kritik yang bersifat membangun demi kesempurnaan skripsi ini.

Malang, Januari 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Game merupakan aplikasi yang paling banyak dinikmati para pengguna saat ini. Perkembangan game dipengaruhi oleh munculnya pengembang *(developer) software game* yang berlomba-lomba mempercantik tampilan animasi dan gamenya. Hal yang paling mendasar game 2D dengan game 3D adalah tingkat detail sederhana sehingga dalam pembuatan membutuhkan waktu yang lebih singkat dengan performa tinggi pada spesifikasi *hardware*. Karena hal inilah maka peneliti mengambil game dengan model 2D (Sibero, 2013).

Game bergenre horror memberikan efek ketakutan pasif atau tidak langsung sehingga pemain game tersebut memberikan interaktivitas yang merangsang respon rasa takut (Madsen, 2016). Untuk menciptakan suasana horror tentunya harus ada dukungan suara, music dan lain sebagainya. Terdapat pengaruh positif dari efek utama atau interaksi suara dan musik dalam bermain game sehingga memberikan kesan tegang (Nacke dkk, 2010). Karena sebagian besar masyarakat Indonesia merasa takut dengan hantu (Suwardi, 2007). Maka Game 2D horror *Girl's Nightmare* ini dibuat dengan tampilan yang lebih menyenangkan tanpa perlu takut saat melawan musuh yang berupa hantu. Hantunya berupa pocong, kuntilanak, dan genderuwo yang merupakan suatu ikon hantu paling dikenal di Indonesia, sehingga lebih mudah meningkatkan keinginan dan ketertarikan pemain.

Terdapat banyak metode dalam pembuatan game, namun yang biasanya digunakan adalah *Finite State Machine*. yaitu sebuah metodologi yang menggambarkan tingkah laku. Sistem kerja dalam metodologi *Finite State Machine* (FSM) dapat dengan mudah diterapkan karena pengendalian sebuah karakter musuh dalam setiap level lebih mudah. Game biasanya terdiri dari tiga level dan adanya perancangan sistem kontrol yang menggambarkan prinsip kerja

sistem yaitu keadaan, kejadian, dan aksi. Penerapan FSM pada game terletak pada perilaku musuh dalam setiap level yang dibuat memiliki peningkatan bertarung yang meningkat dalam setiap penaikan level sehingga kecerdasan buatan pada pembuatan musuh lebih cerdas (Hendriano dkk, 2015).

Metode FSM adalah sebuah metodologi yang memiliki sistem *reward* dinamis dengan menggunakan agen cerdas berbasis *Finite State Machine* (FSM) yang diimplementasikan dalam game bergenre *Adventure-Horror*. Selain untuk sistem kontrol, FSM adalah model yang umum digunakan untuk merancang perilaku agen cerdas di game 2D *Horror Girl's Nightmare* yang mempunyai kelebihan pada kesederhanaan komputasinya dan kemudahan dalam pemahaman dan implementasinya (Haryanto, 2016). *Prototipe* atau *tools* yang digunakan dalam pengembangan game yaitu *Unity3D* dan perangkat lunak pendukung yaitu *Corel Draw*.

Oleh karena itu, game 2D dengan metode *Finite State Machine* memiliki grafik yang lebih sederhana dan pengalaman dalam memainkan game horror lebih menyenangkan dibandingkan game horror yang lain serta hal interaksi pemain terhadap NPC menggunakan FSM sehingga menentukan aksi atau respon musuh terhadap pemain lebih adaptif karena umpan balik ditentukan dari pilihan pemain (Haryanto, 2016).

1.2 Rumusan Masalah

Rumusan masalah dari penulisan skripsi ini adalah sebagai berikut :

1. Bagaimana membuat game *Horror* 2D *Girl's Nightmare* dengan *Unity?*
2. Bagaimana menerapkan metode FSM (*Finite State Machine*) pada *game* 2D *Girl's Nightmare*?

1.4 Batasan Masalah

Batasan masalah dari penulisan proposal ini adalah sebagai berikut :

1. Pada penelitian ini hanya mengembangkan algoritma pemrograman sesuai dengan metode *Finite State Machine*.

2. Bahasa pemrograman yang digunakan untuk mengembangkan sistem adalah bahasa C# pada *tools Unity3D*.
3. Desain karakter yang digunakan pada pembuatan game ini menggunakan *Corel Draw* X7 dan grafik 2D
4. Game ini hanya dimainkan pada PC, berbasis *desktop*, dan sistem operasi *Windows.*
5. Level pada game ini terbagi atas 3 tahapan level.
6. Karakter pada game ini terdiri dari si "*Addis*" sebagai karakter utama dan hantu-hantu.
7. Penerapan metode FSM pada game ini terletak pada perilaku musuh dimana perilaku musuh berpatroli mencari pemain dan menyerang pemain pada jarak tertentu.
8. Target pemain pada game ini adalah semua kalangan usia dan berupa *single-player* (satu pemain).

1.5 Tujuan

Tujuan dari penulisan proposal ini adalah sebagai berikut :

1. Menerapkan metode *Finite State Machine* untuk menghasilkan suatu kecerdasan buatan pada perilaku musuh dalam game *Girl's Nightmare.*
2. Mengimplementasikan game *Girl's Nightmare* kedalam bentuk game *desktop*.

1.6 Metode Penelitian

Untuk dapat mencapai keinginan dalam pembuatan *game horror Girl's Nightmare*, dengan menggunakan metode *FSM (Finite State Machines)* ini, maka perlu dilakukan dengan langkah-langkah sebagai berikut:

1. Studi Literatur

   Penelitian ini dimulai dengan studi literatur yaitu pengumpulan data yang berhubungan dengan permasalahan yang dibahas sehingga dapat membantu

penyelesaian masalah dalam perancangan *game* dari sumber-sumber bacaan seperti, buku, jurnal, referensi, *web page*, dan karya tulis ilmiah.

2. Analisa kebutuhan

   Data dan informasi yang telah diperoleh akan dianalisa agar didapatkan suatu kerangka yang digunakan untuk acuan perancangan perangkat lunak.

3. Perancangan Sistem

   Data-data yang telah terkumpul diimplementasikan kedalam program bersama dengan pembuatan metode *FSM (Finite State Machines)*.

4. Pengujian Program

   Pengujian coba ini bertujuan untuk memastikan bahwa masing-masing bagian dari sistem ini dapat bekerja sesuai yang diharapkan.

1.7 Sistematika Penulisan

Dalam penyusunan skripsi ini agar lebih mudah dipahami maka dibuatlah suatu sistematika dalam penulisan sebagai berikut :

BAB I : PENDAHULUAN

Pada Bab ini menjelaskan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metode penelitian, dan sistematika penulisan.

BAB II : LANDASAN TEORI

Pada bab ini membahas tentang Landasan Teori yang menguraikan teori-teori yang mendukung judul, dan pembahasan secara detail. Landasan tori dapat berupa definisi-definisi atau model yang langsung berkaitan dengan ilmu atau masalah yang diteliti. Pada bab ini juga dituliskan tentang *software* (komponen) yang digunakan dalam pembuatan program atau keperluan saat penelitian.

BAB III : ANALISA DAN PERANCANGAN SISTEM

Bab ini membahas tentang analisa pada sistem dan perancangan sistem yang dibuat.

BAB IV : IMPLEMENTASI DAN PENGUJIAN PROGRAM

Pada bab ini membahas paparan implementasi dan analisa hasil pengujian program yang dibuat.

BAB V : PENUTUP

Pada bab ini berisi kesimpulan dan saran yang didapat dari ulasan data-data penelitihan, menyimpulkan bukti-bukti yang diperoleh dari hasil analisa.

BAB II
TINJAUAN PUSTAKA

2.1. Tinjauan pustaka

2.1.1 Penelitian tentang *Game Horror*

Perkenalan budaya dari sebuah sistem kedalam sebuah permainan atau game. Dengan permainan ini, dapat mengetahui fungsi-fungsi dari benda pusaka yang menjadi benda kebudayaan Indonesia. Permainan ini mengajarkan bagaimana menjaga budaya yang diwarisi nenek moyang, dengan mengetahui mitos-mitos misteri yang ada dan budaya mitos tersebut dapat dikenalkan kembali melalu pembuatan game ini. Metode atau cara penyelesaiannya adalah tahap studi literatur dimana dengan mencari konsep dasar menggunakan game engine yaitu Unity serta mempelajari bahasa pemrograman C#, kemudian proses awal dengan mengumpulkan asset yang dimasukkan kedalam game, seperti pembuatan karakter dalam game, implementasi sistem yaitu pemasukan karakter kedalam game engine dan memasukkan script sesuai dengan yang dikehendaki, pengujian dan testing kembali pada game yang telah dibuat, dan penyusunan laporan. Hasil yang diperoleh oleh jurnal ini yaitu banyaknya peminat game tersebut. Permainan ini meningkatkan pengetahuan tentang benda pusaka yang merupakan budaya asli Indonesia. Dapat dilihat dari hasil kuisioner pemain setelah memainkan permainan ini (Hendriano dkk, 2015).

Perkembangan game yang ada di Indonesia dan kontribusi dalam dunia game Indonesia. Dengan kontribusi ini diharapkan perkembangan game di Indonesia semakin maju terutama game yang dilatar belakangi dengan unsur budaya Indonesia. Game bertema petualangan-misteri ini bersifat hiburan yang diharapkan dapat meningkatkan sreatifitas serta inovasi. Metode pengembangan sistem yang digunakan adalah *Multimedia Development Life Cycle (MDLC)* versi Luther-Sutopo. Tahapnya berupa pengonsepan dimana tahap ini yaitu penentuan tujuan dan siapa pengguna program, perancangan yaitu tahap

pembuatan spesifikasi, pengumpulan material yaitu pengumpulan bahan sesuai dengan yang dibutuhkan, pembuatan yaitu tahap pembuatan semua objek, dan pengujian yaitu menyelesaikan tahapan pembuatan dan menjalankan aplikasi apakah ada terdapat kesalahan atau tidak, dan yang terakhir adalah pendistribusian. Hasil dari sistem yang dibuat adalah kesesuaian dengan perancangan yang telah dikonsep, dan layak sebagai hiburan dan salah satu bentuk ikut memajukan dunia game Indonesia dan rata-rata pengguna game cukup pro terhadap game ini (Pratama dkk, 2011).

Dalam meningkatkan efisiensi kepribadian baik diperlukan berbagai alternatif pembentukan karakter terutama pada game yang beredar di masyarakat. Karena game adalah bagian dari pembentukan kepribadian pada jiwa muda, maka metode pembelajaran ini bisa pula dijadikan tolak ukur yang baik. Metode yang digunakan yaitu metode pembelajaran, dimana metode pembelajaran ini memberi pengalaman belajar anak, metode demonstrasi merupakan suatu cara untuk menunjukkan cara mengerjakan sesuatu yang berupa ilustrasi suatu kejadian, dan metode bermain yaitu suatu sarana untuk berlatih, mengeksploitasi yang dilakukan secara berulang-ulang. Hasil dari penelitian ini yaitu dengan action berupa pemain dapat membunuh Musuh berjenis binatang dengan tombak, serta membunuh musuh berjenis hantu dengan amalan doa. Hasil pengujian yang ada pada game ini yaitu menunjukkan bahwa aplikasi ini dapat digunakan dengan baik, dan sebagian besar responden menyukai game ini (Ghazali, 2017).

2.1.2 Penelitian tentang Game 2D menggunakan metode Finite State Machine

Game dikembangkan dan dijadikan sarana pendidikan yang bersifat adaptif, dimana ini dapat disesuaikan dengan penggunanya. Dengan adanya game ini dapat merancang agen cerdas yang mempunyai kelebihan pada kesederhanaan komputasi dan kemudahan dalam pemahaman. Metode yang terdapat pada penelitian ini yaitu perancangan model skenario adaptif yang menggabungkan antara konsep elemen yang disajikan berdasarkan pengalaman. Pengalaman

menyenangkan dalam bermain game dihasilkan dari elemen audio vidual, fisik, tantangan, kognitif, dan fantasi. Metode yang digunakan menggunakan finite state machine, dimana fsm ini digunakan untuk menggambarkan perilaku agen cerdas.Tahapnya yaitu pendefinisian perilaku yang ada, penggambaran sebagai state, dan penentuan transisi antar state dan kondisi serta penggabungan smua state. Hasil dari pencapaian penilitian ini yaitu komponen adiktif menjadi pembelajaran yang dapat diapresiasikan dari pengalaman dan kemudian aktivitas pembelajaran ini menjadi skenario game yang apik. Kelebihan dari FSM adalah perancangan dan implementasinya yang mudah dan tidak membebani sistem (Haryanto dkk, 2014).

Adanya ketidakseimbangan antara pengguna dengan tingkat kesulitan game sehingga timbul kebosanan dalam memainkan game. Berkaitan dengan permasalahan tersebut, bisa diketahui bahwa adanya kemampuan pengguna yang terlalu bervariasi sehingga dibuatnya game yang mampu menyesuaikan kemampuan dan karakteristik pengguna. Metode penelitian terdiri dari beberapa tahapan yaitu tahapan penelitian dimana terdiri dari lima langkah : pendalaman literatur, perancangan sistem, implementasi, pengujian, dan analisa hasil pengujian. Pendalaman literatur yaitu mempelajari berbagai literatur penelitian yang sudah ada, perancangan sistem dengan menggunakan rancangan reward dinamis dengan FSM, implementasi yaitu reward dinamis diimplementasikan dalam game berjenis Role Playing Game, pengujian, dan analisa hasil pengujian dilakukan uji coba dan evaluasi. Dalam reward dinamis berbasis FSM, setiap pilihan yang diambil oleh pemain menentukan reward yang didapatkan. Hasil yang ditemukan dalam penelitian ini yaitu pada penelitian ini, telah dirancang reward dinamis yang merupakan bagian dari skenario adaptif dimana reward memberikan umpan balik dari pilihan yang ditentukan pemain namun perilaku reward ini masih dapat diprediksi dengan mudah karena keterbatasan FSM dengan memodelkan perilaku (Haryanto, 2016).

Perkembangan game yang dapat dijadikan alat pembelajaran bagi pendidikan, perkembangan game sebagai alat pembelajaran bertujuan agar proses belajar lebih mudah dan menyenangkan. Penelitian ini dapat memberikan kontribusi pada pembuatan agen cerdas dalam game yang diterapkan dalam pengembangan pembelajaran di sekolah. Metode yang digunakan yaitu Finite State Machine yang dikatan sabagai mesin abstrak yang dapat berada dalam satu dari beberapa state. Pada game ditunjukkan bahwa setiap kondisi yang ada pada lingkungan game yang dihasilkan oleh pemain akan menyebabkan perpindahan state.Secara garis besar, hasil yang diperoleh oleh game ini ialah berisi materi pembelajaran, level yang ditentukan sebarapa sulit misi yang diselesaikan. Misi/cerita yang ada menuntut pemahaman pemainn dalam materi pembelajaran sehingga pemain secara tidak sadar belajar untuk menang melawan agen cerdas (Haryanto dkk, 2010).

Dalam suatu permainan terdapat beberapa tipe animasi didalamnya, salah satunya yaitu role playing game, dimana RPG yaitu permainan dimana player atau pengguna memerankan tokoh tertentu dan berkolaborasi dengan tokoh lain untuk membentuk suatu cerita, sehingga pengguna dapat melakukan improvisasi membentuk arah dan hasil akhir permainan. Hal ini membentuk kreatifitas dan imajinasi pengguna, sehingga menjadi tujuan utama penelitian ini.Metode penelitian yang digunakan yaitu FSM dimana adanya perancangan FSM NCP parter yang bertugas membantu player dalam battle system. Perancangan NPC musuh dengan state awal dari NPC musuh adalah state homing dengan dua kondisi yang transisi yang mempengaruhi yaitu kondisi player approaching dan no player. Dari tabel terjadinya *state "Heal"* dan tidak mengalami *state "Heal"*. Tidak terjadinya *state "Heal"* ini kemungkinan dikarenakan *player* mengalami *state "HP player full"* dimana akibat dari penggunaan *item* atau *recover HP* yang terjadi pada *quest* sebelumnya. Lalu dilakukan engujian lagi dengan hasil 19 responden atau 76% mengalami terjadinya *state-transition "HP Player* $\leq$ 50% *initial HP"*. Dari keseluruhan pengujian, responden lebih memilih untuk

menyelesaikan jalan cerita daripada mengoptimalkan pembangunan karakter (Nendya dkk, 2015).

Berkembangnya *video game* di Indonesia mendorong berdirinya berbagai perusahaan pengembangan *game* atau biasa disebut *game developer*. Jika dibandingkan dengan *game developer* asing, pangsa pasar *game* di Indonesia masih didominasi oleh *developer* asing. Dengan kata lain, penjualan keseluruhan *platform game* yang dikembangkan oleh *game developer* asing lebih banyak dibandingkan dengan game *developer* lokal. Berdasarkan permasalahan yang ada, maka ia mengembangkan game lokal menjadi hiburan. Game ini dirancang khusus dengan tujuan sebagai salah satu media hiburan yang menjadi pilihan untuk menghilangkan kejenuhan. Game ini mendukung platform android, dan menggunakan tablet berbasis mobile game. Jumlah pemain yang ada adalah single player. Environtment yang digunakan dalam pembuatan game yaitu tileset dimana tileset adalah sebuah file gambar yang berisikan blok-blok gambar yang menyusun sebuah map dari setiap levelgame. Player dalam game ini dapat berjalanan ke kanan dan ke kiri, melompat dan menggunakan senjata untuk mengalahkan musuh dan musuh tersebut dapat mendekati dan menyerang player pada jarak yang sudah ditentukan (Mahardika dkk, 2016).

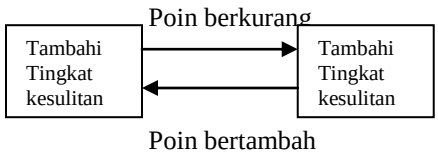
Anak-anak masa kini lebih menyukai permainan modern sehingga hal-hal yang bersifat tradisional seperti cerita rakyat sudah mulai ditinggalkan, maka akan dibuat sebuah game bertemakan cerita rakyat yang disajikan dalam konsep adventure. Berdasarkan pengujian dapat disimpulkan bahwa *game* yang pada *game* ini sehingga lebih menarik dan menantang bagi *user*. Perluditambahkan sehingga lebih menarik.*Game* ini perlu pengembangan desain dibuat dalam penelitian ini masih perlu perbaikan dalam hal gambar *interface* agar lebih dipahami oleh *user*. *AI* Selain itu terdapat beberapakesulitan dalam implementasi cerita rakyat menjadi *adventure game.*Misalnyapada cerita Buaya Ajaib Sungai Tami yang terlalu sederhana isi ceritanya sehingga sulit untuk membuat permainan dari cerita ini (Tjahjono dkk, 2017).

Musik bambu merupakan kebudayaan daerah yang sudah mulai hilang ditelan zaman, dengan perkembangan teknologi seperti sekarang ini, pengetahuan dan keinginan untuk mendengar alat musik bambu sangat kurang. Karena itu penulis membuat prototipe game dengan genre edukasi dengan menggunakan *Unity 3D* dengan tujuan untuk melestarikan kebudayaan daerah yaitu alat musik bambu dan menjadikan game ini sebagai sarana pengenalan musik bambu. Metode rapid game prototyping adalah metode hasil modifiasi dari metode pengembangan perangkat lunak cepat pemrograman ekstrim.Tahap analisa kebutuhan bertujuan mengidentifikasi kebutuhan user, data, dan user stories.Tahap perancangan berupa storyboard, 3D modeling, dan prototyping. Untuk menyempurnakan prototipe game musik bambu yang telah dibuat agar dapat lebih baik lagi dan dapat dipublikasikan secara global maka perlu dilakukan pengembangan lebih lanjut, yaitu mengembangkan game agar memiliki lebih banyak level, agar permainan tidak terkesan terlalu mudah juga perlu ditambahkannya objek alat musik bambu dalam game agar pemain dapat mengetahui lebih banyak jenis alat musik bambu (Tjahyadi dkk, 2014).

2.2. Landasan Teori

2.2.1 Metode *Finite State Machine*

Perilaku adaptif dari skenario game bertujuan supaya pengalaman pengguna dijaga berada di dalam zona *flow* atau ZPD, sehingga memperhatikan tingkat kesulitan dan kemampuan pengguna. Adaptasi yang dilakukan berdasarkan *flow experience* ditunjukkan pada FSM di Gambar 2.1 berikut.



Gambar 2.1 Finite State Machine

Appreciative Learning mengarahkan pada aktivitas pembelajaran yang terdiri dari *Design, Dream, Destiny* dan *Discovery*. Dalam *Appreciative* Learning dikemukakan bahwa banyak alternatif yang disediakan untuk pembelajar. Alternatif-alternatif tersebut akan disediakan melalui level dan interaktivitas dalam game yang akan memicu salah satu aktivitas pembelajaran supaya aktif. (Eow dkk, 2010)

2.2.2 Pemrograman C#

C# adalah bahasa pemrograman baru yang diciptakan oleh Microsoft yang dikembangkan dibawah pimpinan Anders Hejlsberg yang telah menciptakan berbagai macam bahasa pemrograman termasuk Borland Turbo C++ dan orland Delphi.Bahasa C# juga telah di standarisasi secara internasional oleh ECMA. Seperti halnya bahasa pemrograman yang lain, C# bisa digunakan untuk membangun berbagai macam jenis aplikasi, seperti aplikasi berbasis windows (desktop) dan aplikasi berbasis web serta aplikasi berbasis web services.

Bahasa C# memiliki keunggulan- keunggulan, antara lain :

1. C# bersifat sederhana.

   C# dikatakan sederhana karena bahasa ini didasarkan kepada bahasa C dan C++. Jika kita telah familiar dengan C dan C++ atau bahkan java, kita akan menemukan aspek-aspek yang begitu familiar, seperti ***statements, expression, operators,*** dan beberapa fungsi yang diadopsi langsung dari C dan C++, tetap dengan berbagai perbaikan yang membuat bahasanya menjadi lebih sederhana.
2. C# adalah bahasa pemrograman yang memiliki level aplikasi yang tinggi. Program C# merupakan sebuah solusi dari permasalahan masa depan dan masa kini. Karena C# merupakan bahasa ***High Level Interporability.*** Dan jika membahas C# secara tidak langsung juga membicarakan teknologi.
3. C# adalah bahasa pemrograman dengan kata kunci atau ***keyword*** sedikit dan lebih mudah. Kata kunci disini adalah merupakan fungsi ataupun kata dasar yang disediakan oleh ***compiler*** suatu bahasa pemrograman. Hal ini membawa

pengaruh semakin mudahnya kita menulis program dengan C#. Pengaruh lain dari sedikitnya kata kunci ini adalah proses eksekusi program C# yang sangat cepat.

4. C# adalah bahasa pemrograman dengan produktivitas yang tinggi.Konsep OOP yang tertanam kuat pada C# memungkinkan pembuatan program yang dapat dengan mudah dikembangakan dengan kekayaan ***class library***.
5. C# adalah bahasa pemrograman yang kuat dan fleksibel. Dengan menguasai bahasa C# sehingga bias menulis dan dikembangkan berbagai jenis program mulai dari ***operating system, word processor, grapic processor, spreadsheets.*** Ataupun ***compiler*** untuk suatu bahasa pemrograman.
6. C# adalah bahasa pemrograman yang bersifat ***moduler***. Program C# ditulis dalam ***routine*** yang biasa dipanggil dengan fungsi. Fungsi-fungsi yang telah kita buat, bisa kita gunakan kembali dalam program ataupun aplikasi lain (Jamie, 2011).

BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1. Deskripsi Game

Di Game *Girl's Nightmare* ini merupakan game dengan gabungan genre advanture dan horror. Game ini terdapat berbagai musuh yang sama tetapi memiliki bos musuh yang berbeda tiap level. Fitur-fitur yang digunakan pada game ini yaitu:

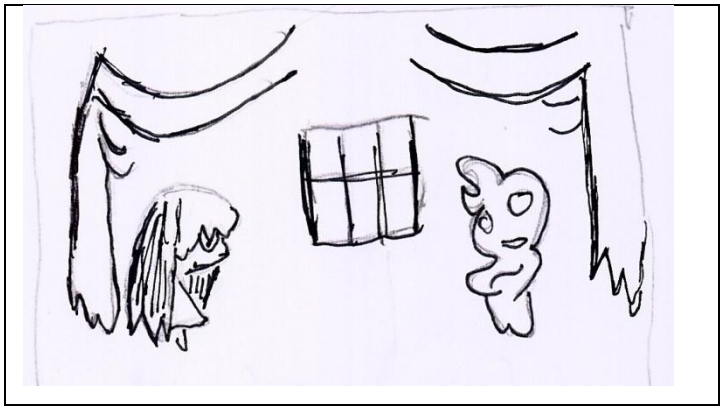
1. Game ini menggunakan grafik 2D
2. Metode yang digunakan adalah Finite State Machine
3. Metode ini digunakan pada perilaku musuh, jika karakter terdeteksi pada area yang telah ditetapkan maka musuh akan mungejar lalu mempengaruhi karakter.

3.2. *Storyline*

Dalam game *Girl's Nightmare* ini terdapat 3 level yang akan dilewati player. Pada level 1 terdapat stage *Haunted House*, player akan terdampar disebuah rumah dan terdiri dari lorong panjang yang terdapat banyak hantu. Di level ini, player harus menghindari dan melawan hantu yang ada. Setelah melewati level 1 maka masuk ke level 2 yaitu *Graveyard*, player diharuskan menghindari dan melawan hantu yang ada. Kemudian untuk level terakhir player akan menghadapi lebih banyak hantu dan menghadapi hantu utama di stage *Forrest*. Pada setiap level player harus bisa mempertahankan nyawanya dan mengumpulkan coin yang ada.

3.3. *Storyboard*

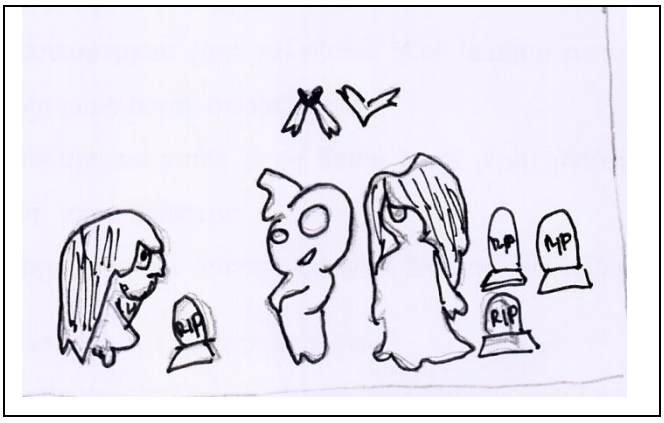
Rancangan *Storyboard* Level 1 pada game Girl's Nightmare dengan stage bernama Haunted House terdapat pada gambar 3.2 dibawah ini.



Gambar 3.1 Haunted House

Pada gambar 3.1 diatas tampak haunted house yang menjadi level 1 pada game ini. Didalam level ini terdapat pocong yang akan menyerang pemain.

Level 2 pada game Girl's Nightmare dengan stage bernama Graveyard terdapat pada gambar 3.2 dibawah ini.



Gambar 3.2 Haunted House

Pada gambar 3.2 diatas tampak graveyard berupa pekuburan yang menjadi level 2 pada game ini. Didalam level ini terdapat pocong dan kuntilanak yang akan menyerang pemain.

Level 3 pada game Girl’s Nightmare dengan stage bernama Forrest terdapat pada gambar 3.3 dibawah ini.

Gambar 3.3 Forrest

Pada gambar 3.3 diatas tampak Forrest berupa hutan yang menjadi level 3 pada game ini. Didalam level ini terdapat pocong, kuntilanak, dan genderuwo yang merupakan musuh terakhir,

3.4. Desain Karakter

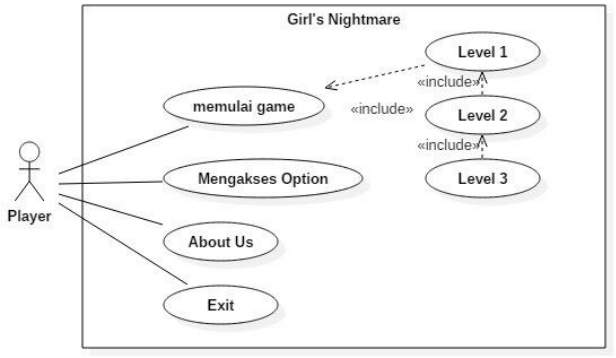
Karakter pada game lady lag terdapat pada Tabel 3.1 dibawah ini. Karakter setiap game memilliki peran masing-masing dan peran tersebut dijelaskan pada keterangan.

Tabel 3.1 desain karakter

NO	Nama Gambar	Keterangan
1.		Tampak samping player
4.		Hantu level 1.
5.		Hantu level 2
6.		Hantu level 3

3.5 *Use Case*

Use Case pembuatan *game horror* "Girl's Nightmare" ditunjukkan pada Gambar 3.4 dibawah ini



Gambar 3.4 *Use case Game*

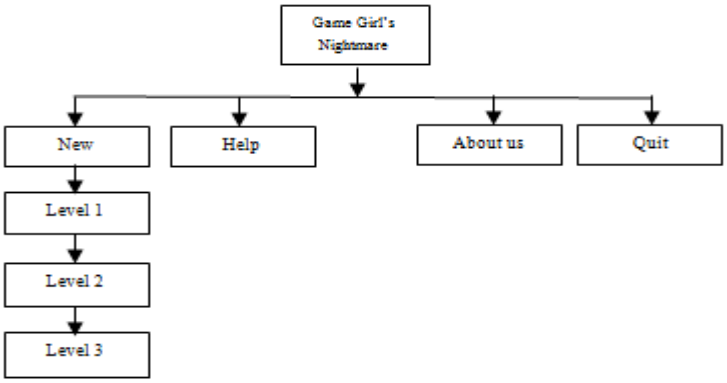
Pada gambar 3.4 diatas dijelaskan bahwa Actor1 merupakan player atau pemain yang bisa memainkan permainan atau *play*, pergantian option menu atau *change option menu*, dan keluar dari game tersebut. Pada *play*, *player* dapat menginilisiasikan permainan.

3.5. Desain Menu

Game ini akan memberikan beberapa level yang harus diselesaikan oleh pemain seperti pada Haunted House, Graveyard, dan Forrest. Beberapa menu yang terdapat pada game ini adalah menu New Game, How to play, dan Quit. Berikut penjelasaan menu yang ada pada game ini :

1. Menu New Game, digunakan pemain jika akan memulai permainan dari awal.
2. Menu How To Play, berfungsi sebagai menu yang menjelaskan tentang kontrol permainan yang nantinya akan digunakan oleh pemain.
3. Menu Quit, digunakan pemain jika akan mengakhiri permainan.

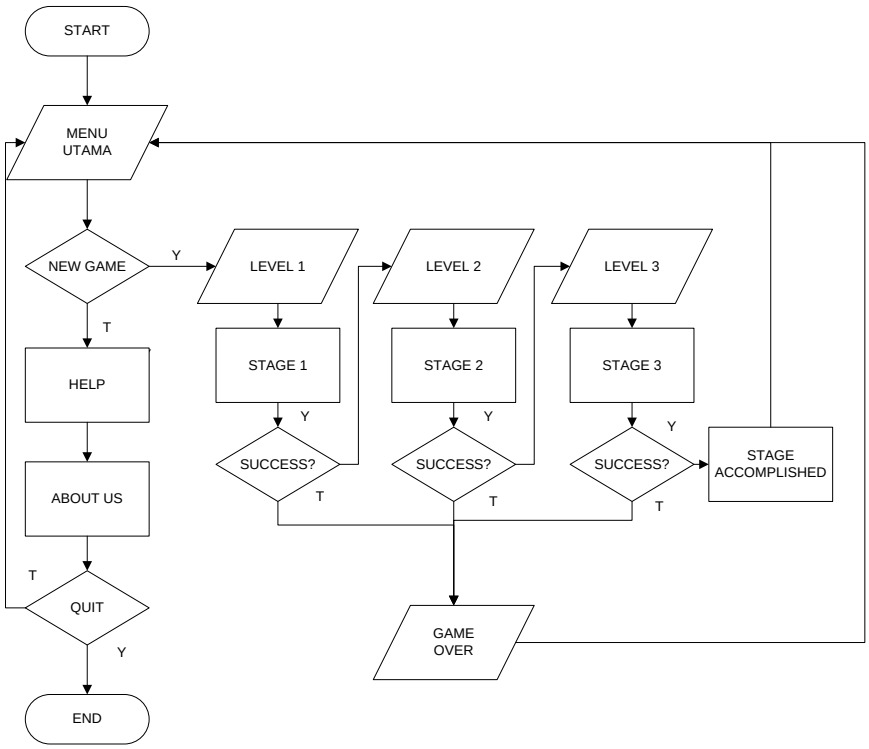
Untuk melihat lebih jelas, di bawah ini merupakan gambar perancangan struktur menu *Girl's Nightmare* terlihat pada Gambar 3.1 dibawah ini.



Gambar 3.5 Desain Menu

3.4 Desain alur

Perancangan alur game Girl's Nightmare agar proses game diketahui ditunjukkan pada Gambar 3.2 dibawah ini.

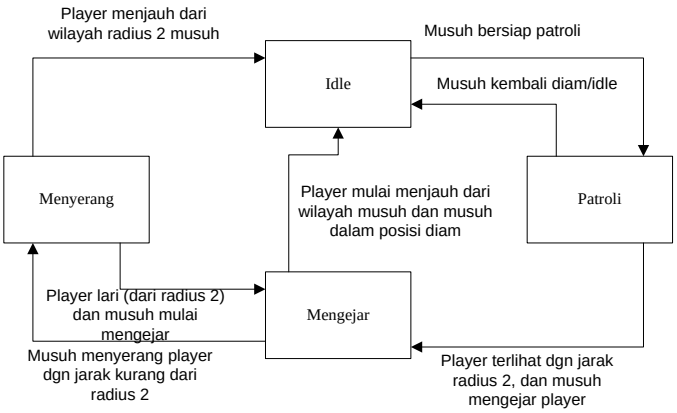


Gambar 3.6 Desain Alur

Pada gambar diatas, program dimulai dari start kemudian masuk ke 3 menu utama yaitu *new game, help, about us dan Quit*. Jika pemain memulai game maka akan masuk ke stage 1 dimana stage 1 merupakan level 1 permainan, jika gagal maka akan *game over* dan jika berhasil maka melanjutkan ke stage 2 sampai stage terakhir. Jika sampai terakhir dan game telah selesai maka akan diarahkan kembali ke *new game.*

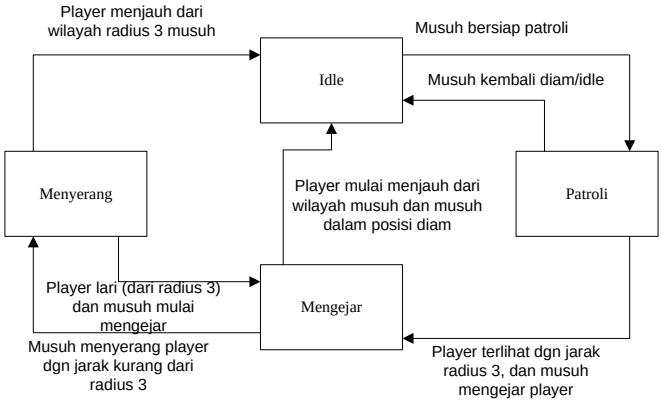
3.7 Diagram Finite State Machine

Finite State Machine yang umum digunakan untuk memodelkan perilaku di dalam game. Berikut konsep dan model dari Finite State Machine.



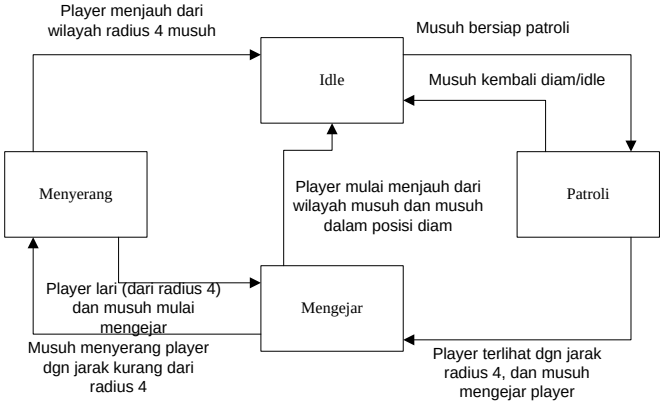
Gambar 3.7 Diagram FSM pocong

Pada gambar 3.7 diatas, state/kondisi diilustrasikan dengan gambar kotak yang mewakili perpindahan musuh dalam state. Perpindahan antar perilaku musuh diilustrasikan sebagai tanda panah. Idle adalah state pertama, jika state pertama terjadi maka musuh bersiap patroli. Patroli adalah state kedua, jika state kedua terjadi maka musuh bisa kembali diam ke state pertama dan player mulai terlihat jika pada stage ketiga. Mengejar adalah stage ketiga dimana musuh dapat menyerang dengan jarak kurang dari radius 2. Setelah itu musuh akan menyerang dengan menyentuh player, jika player berlari dan tidak dalam jangkauan musuh maka musuh akan kembali ke state pertama.



Gambar 3.8 Diagram FSM Kuntilanak

Pada gambar 3.8 diatas, state/kondisi diilustrasikan dengan gambar kotak yang mewakili perpindahan musuh dalam state. Perpindahan antar perilaku musuh diilustrasikan sebagai tanda panah. Idle adalah state pertama, jika state pertama terjadi maka musuh bersiap patroli. Patroli adalah state kedua, jika state kedua terjadi maka musuh bisa kembali diam ke state pertama dan player mulai terlihat jika pada stage ketiga. Mengejar adalah stage ketiga dimana musuh dapat menyerang dengan jarak kurang dari radius 3. Setelah itu musuh akan menyerang dengan menyentuh player, jika player berlari dan tidak dalam jangkauan musuh maka musuh akan kembali ke state pertama.



Gambar 3.9 Diagram FSM Genderuwo

Pada gambar 3.9 diatas, state/kondisi diilustrasikan dengan gambar kotak yang mewakili perpindahan musuh dalam state. Perpindahan antar perilaku musuh diilustrasikan sebagai tanda panah. Idle adalah state pertama, jika state pertama terjadi maka musuh bersiap patroli. Patroli adalah state kedua, jika state kedua terjadi maka musuh bisa kembali diam ke state pertama dan player mulai terlihat jika pada stage ketiga. Mengejar adalah stage ketiga dimana musuh dapat menyerang dengan jarak kurang dari radius 4. Setelah itu musuh akan menyerang dengan menyentuh player, jika player berlari dan tidak dalam jangkauan musuh maka musuh akan kembali ke state pertama.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Hasil Implementasi

Hasil Implementasi adalah proses penerapan rancangan sistem yang telah dibuat menjadi suatu aplikasi yang bisa dijalankan pada kenyataanya. Disamping itu, implementasi ini juga berfungsi untuk mengetahui tingkat keberhasilan dari rancangan yang telah dibuat. Implementasi aplikasi ini dibagi menjadi beberapa bagian diantaranya :

4.1.1 Tampilan Menu Utama

Tampilan menu utama pada game Girl's Nightmare ini memiliki 3 menu utama yaitu, new game, how to play, dan exit. New game untuk memulai game, how to play untuk mengetahui kontrol game, dan exit untuk keluar dari game. Seperti terliht pada Gambar 4.1 dibawah ini

Gambar 4. 1 Tampilan menu utama

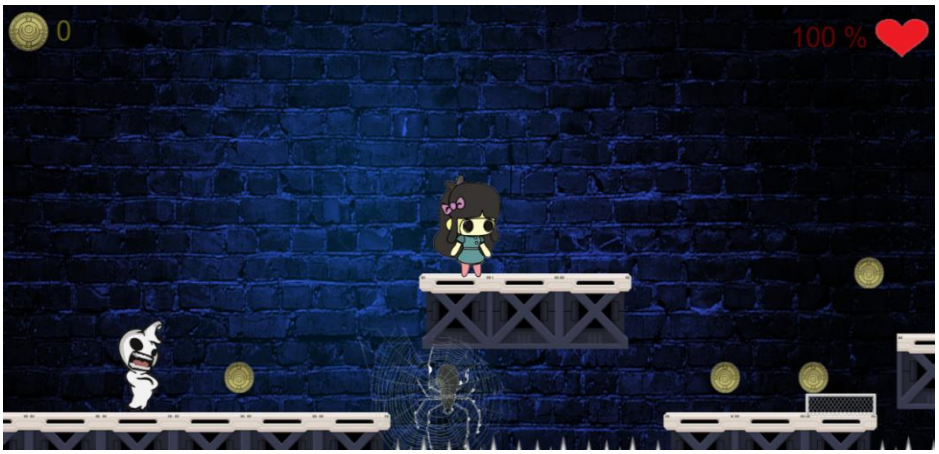
4.1.4 Tampilan Menu Help

Pada awal game Girl's Nightmare, pemain akan diarahkan ke permainan baru yaitu *Haunted House,* seperti ditunjukkan pada Gambar 4.2 dibawah ini

Gambar 4. 2.Tampilan menu *Help*

4.1.5 Tampilan Game Stage 1, 2 dan 3

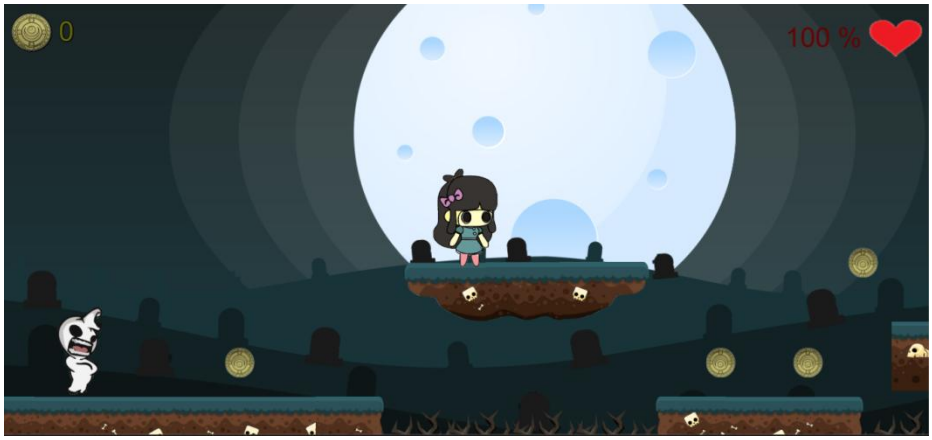
Tampilan *Game* adalah tampilan pada saat *gameplay* atau permainan sedang berjalan. Adapun desain *game* seperti berikut:



Gambar 4. 3.Tampilan *Game Stage 1*

Pada gambar 4.3 adalah tampilan *gambar stage* 1, di stage 1 ini ada satu *character player* dan musuh, untuk *character player* nya seorang gadis,

dan untuk *character* musuh yaitu *pocong*. Untuk tingkat kesulitan di *sta*ge 1 masih pada tingkat mudah



Gambar 4. 4.Tampilan *Game Stage 2*

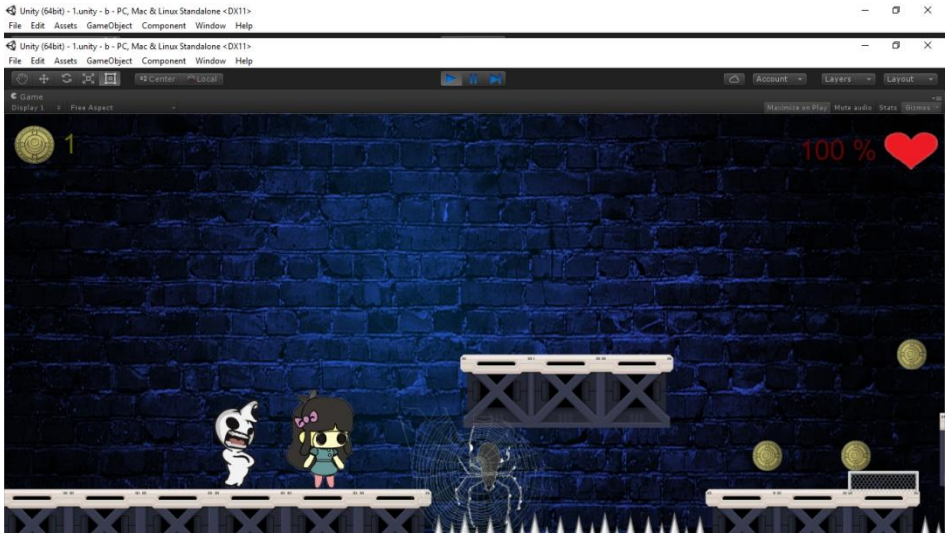
Pada gambar 4.4 yaitu tampilan *gambar stage* 2, di stage 2 ini ada dua *character* yaitu *player* dan *character* musuh. Untuk tingkat kesulitan di *sta*ge 2 masih pada tingkat sedang.

Gambar 4. 5.Tampilan *Game Stage 3*

Pada gambar 4.5 yaitu tampilan *gambar stage* 3, di stage 3 ini ada dua *character* yaitu *player* dan ada *character* musuh. Untuk tingkat kesulitan di *sta*ge 3 sudah mencapai tingkat sulit.

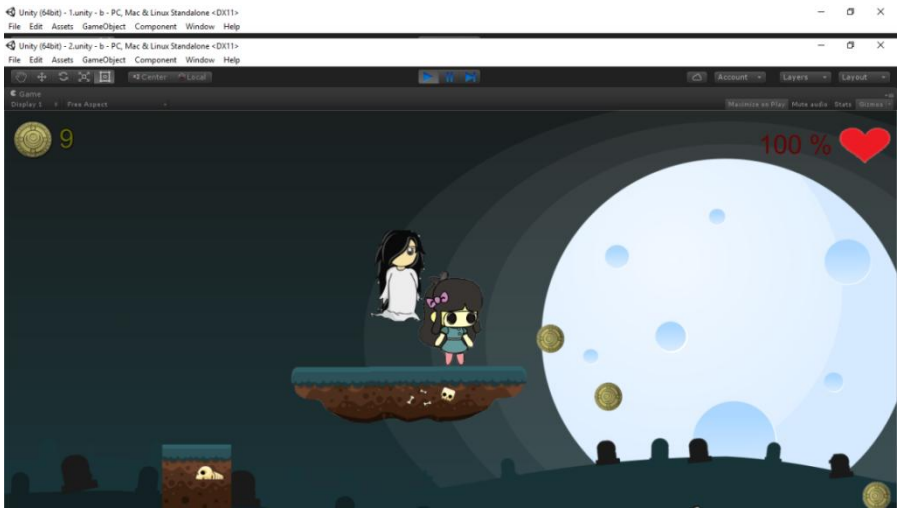
4.2. Tampilan *GameplayStage* 1, 2, dan 3

Pengujian *gameplay* adalah pengujian bagaimana *game* tersebut berjalan sesuai dengan rancangan sistem yang telah dibuat sebagai berikut:



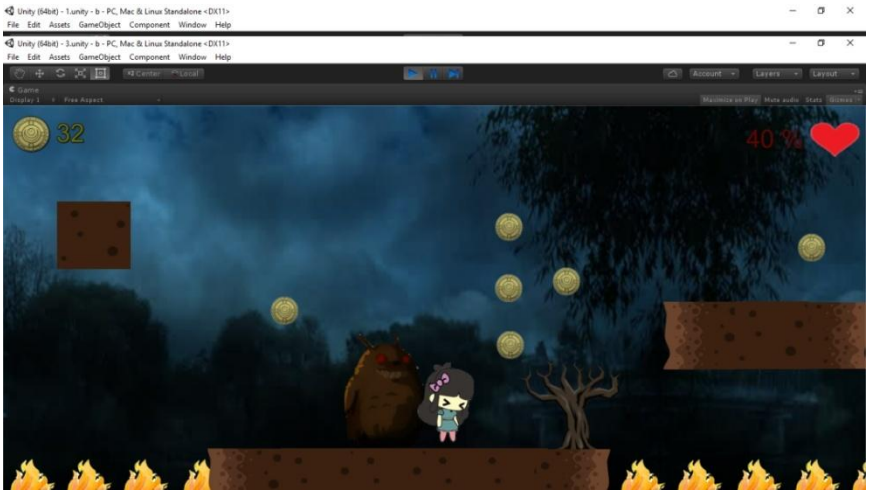
Gambar 4. 6.Tampilan *Gameplay Stage* 1

Pada Gambar 4.6 *stage* 1, menggambarkan *pocong* yang sedang mengejar *player*. Hal ini dikarenakan *player* sudah memasuki wilayah jangkauan musuh sehingga, musuh akan mengejar dan berusaha menyerang *player*. Pada *stage* 1 dengan tingkat yang masih mudah, musuh sangat lambat ketika mengejar *player*.



Gambar 4. 7.Tampilan *Gameplay Stage* 2

Pada Gambar 4.7 *stage* 2, menggambarkan *kuntilanak* yang sedang mengejar *player*. Hal ini dikarenakan *player* sudah memasuki wilayah jangkauan musuh sehingga, musuh akan mengejar dan berusaha menyerang *player*. Pada *stage* 2 dengan tingkat yang sedang, musuh mulai cepat ketika mengejar *player*.

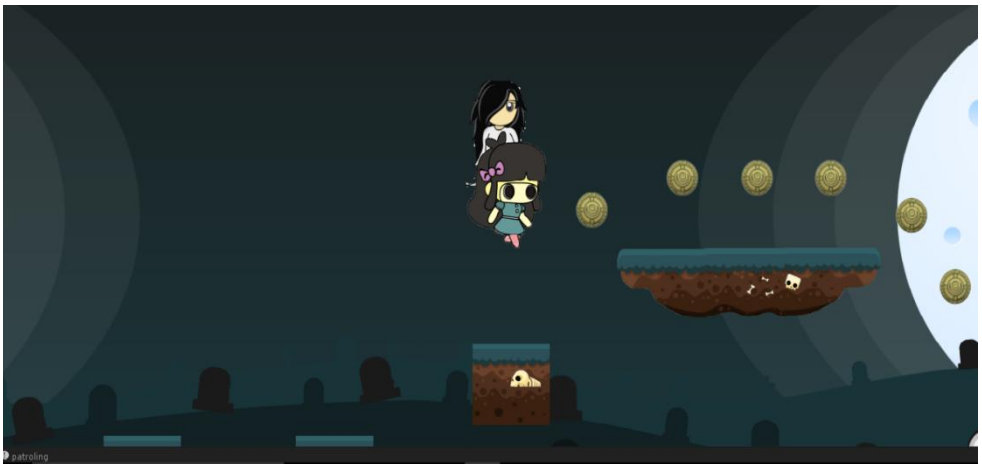


Gambar 4. 8.Tampilan *Gameplay Stage* 3

Pada Gambar 4.8 *stage* 3, menggambarkan gondoruwo yang sedang mengejar *player*. Hal ini dikarenakan *player* sudah memasuki wilayah jangkauan musuh sehingga, musuh akan mengejar dan berusaha menyerang *player*. Pada *stage* 3 dengan tingkat yang sulit, kecepatan musuh saat mengejar mulai bervariasi. Ada yang mengejar player dengan lambat, dan nada juga yang mengejar player dengan cepat.

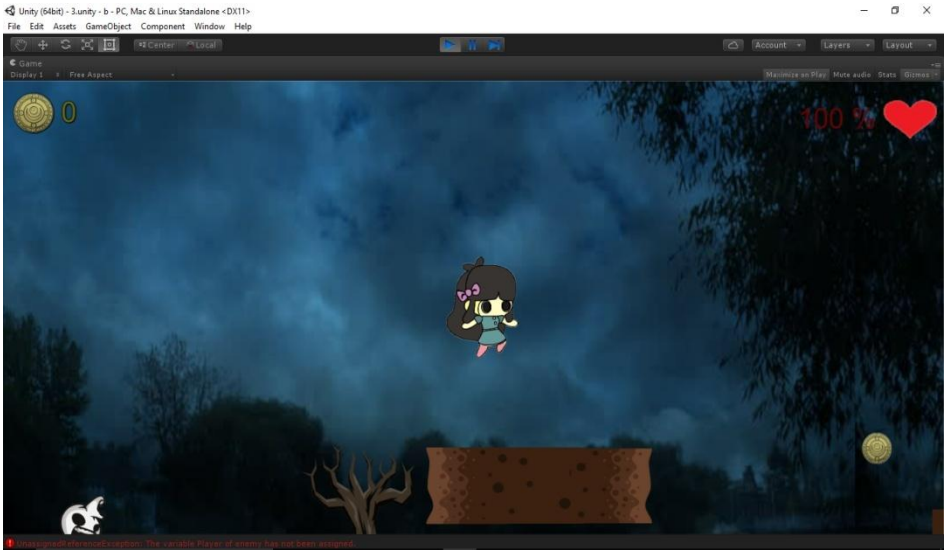
Gambar 4. 9.Tampilan *Gameplay Stage* 1 (*Jumping*)

Pada Gambar 4.9, menggambarkan player melompat dari satu platform ke satu platform yang lain. Hal ini dikarenakan *player* bisa segera keluar dari stage 1 ini menuju stage 2.



Gambar 4. 10 Tampilan *Gameplay Stage* 2 (*Jumping*)

Pada Gambar 4.10, menggambarkan player melompat dari satu platform ke satu platform yang lain. Hal ini dikarenakan *player* bisa segera keluar dari stage 2 ini menuju stage 3.

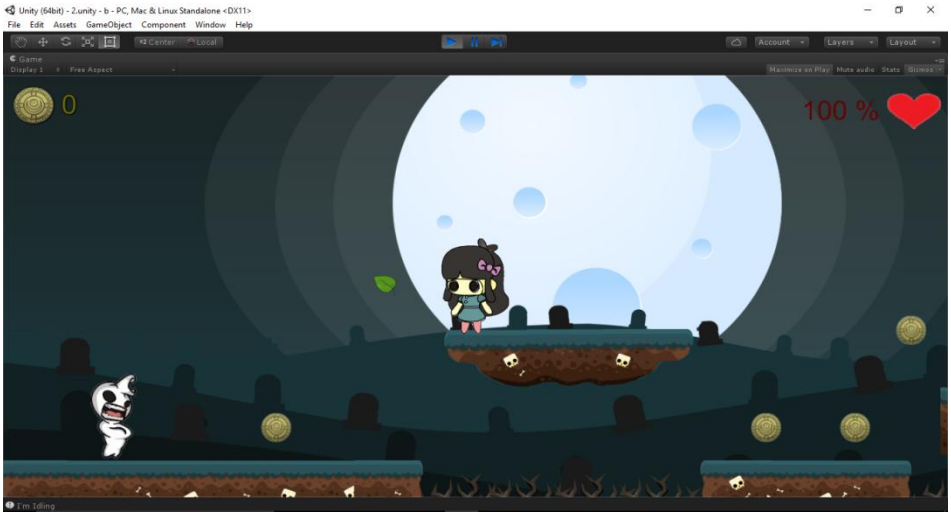


Gambar 4. 11 Tampilan *Gameplay Stage* 3 (*Jumping*)

Pada Gambar 4.11, menggambarkan player melompat dari satu platform ke satu platform yang lain. Hal ini dikarenakan *player* bisa segera keluar dari stage 3 dan berhasil menyelamatkan diri.

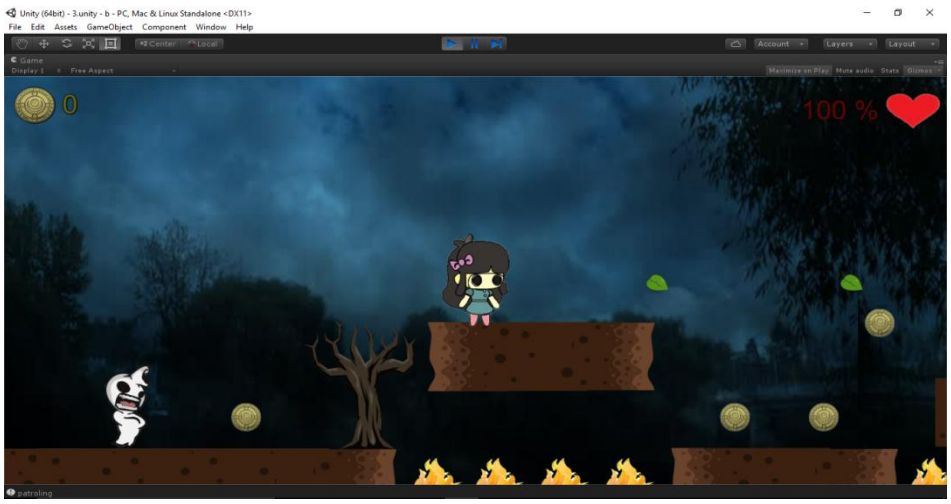
Gambar 4. 11 Tampilan *Gameplay Stage* 1 (*Attack*)

Pada Gambar 4.11, menggambarkan player menyerang musuh dengan tembakan kelornya. Hal ini menyebabkan musuh mati dan *player* bisa segera keluar dari stage 1 ini menuju stage 2.



Gambar 4. 12 Tampilan *Gameplay Stage* 2 (*Attack*)

Pada Gambar 4.12, menggambarkan player menyerang musuh dengan tembakan kelornya. Hal ini menyebabkan musuh mati dan *player* bisa segera keluar dari stage 2 ini menuju stage 3.



Gambar 4. 13 Tampilan *Gameplay Stage* 3 (*Attack*)

Pada Gambar 4.13, menggambarkan player menyerang musuh dengan tembakan kelornya. Hal ini menyebabkan musuh mati dan *player* bisa segera keluar dari stage 3 dan menyelamatkan diri.

4.3. Pengujian *Artificial Intelligence* (AI)

Pengujian AI adalah pengujian mengenai fungsi yang berkaitan dengan AI (Artificial Intelligence) yang ada dalam *Game Girl's Nightmare*. Hasil pengujian dari AI *Finite State Machine* dapat dilihat pada Tabel 4.1.

Tabel 4.1 Pengujian *AI (*Artificial Intelligence*)* untuk stage 1, 2, dan 3

No	Fungsi	Output	Hasil
1	*Range player* dengan musuh jarak sama dengan 2	Player terlihatdan musuh mengejar player	Berhasil (Gambar 4.6)
2	*Range player* dengan musuh >2	Player mulai menjauh jarak lebih dari radius 2 dan musuh kembali dalam posisi diam	Berhasil (Gambar 4.6)
3	*Range player* dengan musuh <2	Musuh menyerang player jarak kurang dari radius 2	Berhasil (Gambar 4.6)
4	*Range player* dengan musuh jarak sama dengan 3	Player terlihatdan musuh mengejar player	Berhasil (Gambar 4.7)
5	*Range player* dengan musuh >3	Player mulai menjauh jarak lebih dari radius 3 dan musuh kembali dalam posisi diam	Berhasil (Gambar 4.7)
6	*Range player* dengan musuh <3	Musuh menyerang player jarak kurang dari radius 3	Berhasil (Gambar 4.7)
7	*Range player* dengan musuh jarak sama dengan 4	Player terlihatdan musuh mengejar player	Berhasil (Gambar 4.8)
8	*Range player* dengan musuh >4	Player mulai menjauh jarak lebih dari radius 4 dan musuh kembali dalam posisi diam	Berhasil (Gambar 4.8)
9	*Range player* dengan musuh <4	Musuh menyerang player jarak kurang dari radius 4	Berhasil (Gambar 4.8)

Berdasarkan Tabel 4.1, AI yang ada pada musuh sudah berfungsi dengan baik. Ketika pemain mendekati musuh, musuh bisa mengejar dan bila pemain semakin dekat, musuh bisa menyerang pemain. *Health bar* dari pemain juga sudah bisa berkurang ketika pemain menerima serangan dari musuh. Dengan demikian dapat disimpulkan bahwa semua AI (*Artificial Intelligence*) yang ada dalam *Game Girl's Nightmare* seluruhnya berjalan dengan baik.

4.4. Pengujian *Stage* 1, 2, dan 3

Pengujian *stage* 1, 2, dan 3 adalah pengujian mengenai proses pengujian *stage* yang terjadi dalam *game*. Hasil dari pengujian dapat dilihat pada Tabel 4.2.

Tabel 4. 2.Pengujian Stage 1, 2, dan 3

No.	Keterangan	Hasil		
		Stage 1	Stage 2	Stage 3
1.	Player dapat berjalan dan meloncat	Berhasil	Berhasil	Berhasil
2.	Musuh dapat menyerang karakter player	Berhasil	Berhasil	Berhasil
3.	Musuh dapat mengejar karakter player	Berhasil	Berhasil	Berhasil
4.	Skor bertambah ketika koin di ambil oleh player	Berhasil	Berhasil	Berhasil
5.	Health poin player berkurang ketika di serang	Berhasil	Berhasil	Berhasil
6.	Player dapat mati ketika health poin	Berhasil	Berhasil	Berhasil
7.	Tombol pergerakan player berjalan sesuai keinginan	Berhasil	Berhasil	Berhasil
10.	Player mati ketika jatuh di jurang	Berhasil	Berhasil	Berhasil

Berdasarkan Tabel 4.2, pengujian yang dilakukan pada karakter utama dan juga musuh. Pemain sudah dapat menggerakkan karakater utama menggunakan tombol yang telah diatur. *Health poin* dari pemain juga sudah berkurang sesuai dengan *damage* yang diterima. Ketika *health poin* sudah habis maka pemain akan mati. Skor di *game* ini juga sudah berfungsi dengan baik. Oleh karena itu, dapat disimpulkan bahwa semua fungsi berjalan dengan baik.

4.5. Pengujian *Control Player*

Pengujian *control player* adalah pengujian fungsi dari setiap tombol yang sudah diterapkan untuk menggerakkan karakter utama. Hasil pengujian *control player* dapat dilihat pada Tabel 4.4.

Tabel 4. 4.Pengujian *Control Player*

No	Tombol	Fungsi	Hasil
1.	Space	Melompat	Sesuai
2.	Control	Melempar kelor	Sesuai
3	Shift	Menendang	Sesuai
4	Panah kiri	Bergerak ke kiri	Sesuai
5.	Panah kanan	Bergerak ke kanan	Sesuai

Dari Tabel 4.3 ketika pemain menekan tombol arah kanan dan tombol arah kiri, karakter utama dapat bergerak sesuai dengan arah tombol tersebut. Ketika pemain menekan tombol spasi karakter juga dapat meloncat. Dan ketika pemain menekan tombol shift maka pemain akan menendang. Dari penjelasan di atas menunjukkan bahwa semua fungsi dari *control player* berjalan dengan baik

4.6. Pengujian *User*

Pengujian dilakukan untuk mengetahui apakah sistem sudah berjalan dengan baik atau belum. Pengujian dilakukan terhadap 10 responden yang terdiri dari orang pecinta *game* yang berasal dari mahasiswa. Kuisioner berisi lima pertanyaan tentang *game Girl's Nightmare.* hasil dari pertanyaan terhadap responden dapat dilihat pada tabel 4.5.

Tabel 4.5. Pengujian *User*

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	8 orang	2 orang	0 orang
2	Desain Animasi pada Game	5 orang	5 orang	0 orang
3	Kontrol pada game	8 orang	2 orang	0 orang
4	Fitur game	6 orang	4 orang	0 orang
5	(Cerita, Narasi Game)	8 orang	2 orang	0 orang
6	Game sudah menarik	6 orang	4 orang	0 orang
7	Fungsi game berjalan	6 orang	4 orang	0 orang
8	Button pada menu menarik	7 orang	3 orang	0 orang
9	Tingkat kesulitan	5 orang	5 orang)	0 orang
10	Game menyenangkan	6 orang	4 orang	0 orang
TOTAL		8	2	0
Persentase		80%	20%	0%

Dari Tabel 4.5 diatas, diketahui mayoritas user menilai baik.

4.7. Pengujian OS/*Performance*

Pengujian *performance* adalah pengujian yang dilakukan pada kinerja atau respon perangkat keras. Pengujian *performance* dimaksudkan untuk mengetahui apakah aplikasi berjalan pada suatu perangkat keras tertentu dengan spesifikasi perangkat keras yang berbeda-beda. Pengujian dilakukan menggunakan 4 komputer dengan spesifikasi berbeda dan menggunakan monitor beresolusi 1366 x 768 pixel. Hasil dari pengujian performance dari *Game Girl's Nightmare* terdapat pada Tabel 4.6.

Tabel 4. 6.Pengujian *Performance*

No	*Processor*	RAM	VGA	OS	Keterangan
1	Core 2 Duo	1GB	512MB	Win XP	Berjalan lancar
2	Intel Corei3	2GB	1GB	Win 8	Berjalan lancar
3	Intel Corei3	4GB	1GB	Win 7	Berjalan lancar
4	Intel Core i5	8GB	2GB	Win 10	Berjalan lancar

Dari Tabel 4.6, dapat disimpulkan bahwa *Game Girl's Nightmare* dapat dijalankan pada computer dengan minimal menggunakan OS Windows XP.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dari perancangan dan implementasi pada *game horror Girl's Nightmare* maka diambil beberapa kesimpulan sebagai berikut:

1. Seluruh AI yang ada pada *game Girl's Nightmare* berjalan baik.
2. Semua fungsi kontrol *game Girl's Nightmare* berjalan baik.
3. *Game Girl's Nightmare* dapat dijalankan pada komputer dengan minimal OS Windows XP.
4. Hasil pengujian user berupa 80% baik dan 20% kurang baik, sehingga Game *Girl's Nightmare* berjalan lancar.

5.2 Saran

Saran ini sebagai acuan terhadap penelitian atau pengembangan selanjutnya dari *game horror Girl's Nightmare*:

1. Grafik karakter musuh, karakter pemain dan objeknya dibuat lebih bagus agar lebih menarik.
2. Penambahan *stage* pada game yang lebih banyak agar permainan bisa lebih lama untuk diselesaikan.
3. Menambahkan fitur *savegame* agar pemain bisa melanjutkan *game* nya lagi ketika *game* dikeluarkan.
4. Serangan dari musuh berupa pocong, kuntilanak, dan genderuwo sebaiknya dibuat lebih menarik.

DAFTAR PUSTAKA

Drachen, A., Nacke, L.E., Yannakakis, G. and Pedersen, A.L., 2010, July. Correlation between heart rate, electrodermal activity and player experience in first-person shooter games. In *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games* (pp. 49-54). ACM.

Eow YL, dkk. 2010. Appreciative Learning Approach : A New Pedagogical Option. In International Conference On Computers In Education.

Haryanto, H., 2016. Reward Dinamis dalam Skenario Adaptif Menggunakan Metode Finite State Machine pada Game Edukasi. *Journal of Applied Intelligent System*, *1*(2), pp.144-153.

Hendriano, K.B. and Safitri, P.C., ASTRAL, Petualangan Game 2D Tentang Mitos Indonesia ASTRAL, 2D Indonesia's Myth Adventure Game Andre.

Pratama, W., 2014. Game Adventure Misteri Kotak Pandora. *Telematika, 7*(2).

Norton, T., 2013. *Learning C# by Developing Games with Unity 3D*. Packt Publishing Ltd, 123-128.

Sibero, I.C., 2010. *Membuat Game 2D menggunakan game maker*. Penerbit Mediakom.

Soon, G.K., Guan, T.T., On, C.K., Alfred, R. and Anthony, P., 2013, November. *A comparison on the performance of crossover techniques in video game. InControl System, Computing and Engineering (ICCSCE), 2013 IEEE International Conference on* (pp. 493-498). IEEE.

L A M P I R A N

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
PROGRAM STUDI TEKNIK INFORMATIKA S-1
Jl. Karanglo, Km. 2 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA : NINIEK MARDIYANI
NIM : 14.18.153
JURUSAN : TEKNIK INFORMATIKA S-1
JUDUL : PENERAPAN METODE *FINITE STATE MACHINE* DALAM PERANCANGAN PERILAKU MUSUH PADA GAME 2D *HORROR GIRL'S NIGHTMARE*

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :
Hari : Senin
Tanggal : 15 Januari 2018
Nilai : 84 (A)

Panitia Ujian Skripsi:

Ketua Majelis Penguji

Joseph Dedy Irawan, ST, MT
NIP. 197404162005011002

Anggota Penguji:

Dosen Penguji I

Sandy Natali Mantja, S. Kom
NIP.P. 1030800418

Dosen Penguji II

Agung Panji Sasmito, S.Pd, M.Pd
NIP. P. 1031500499

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
PROGRAM STUDI TEKNIK INFORMATIKA S-1
Jl. Karanglo, Km.2 Malang

FORMULIR BIMBINGAN SKRIPSI

Nama : Niniek Mardiyani
NIM : 1418153
Masa Bimbingan : 25 September 2017 s/d 25 Maret 2018
Judul Skripsi : Penerapan Metode *Finite State Machine* dalam Perancangan Perilaku Musuh pada Game 2d *Horror Girl's Nightmare*

No	Tanggal	Uraian	Paraf Pembimbing
1.	6 Oktober 2017	Konsul skenario game	
2.	11 Oktober 2017	Demo karakter	
3.	30 Oktober 2017	Demo game	
4.	31 Oktober 2017	Demo game	
5.	23 November 2017	Level 1 rumah tua, level 2 makam dan level 3 hutan	
6.	30 November 2017	Tambahkan halaman pengujian user	
7.	21 Desember 2017	Demo : poin dan coin	
8.	9 Januari 2018	Demo game	
9.	10 Januari 2018	Draft skripsi	
10	12 Januari 2018	Acc skripsi	

Malang, 24 Januari 2018
Dosen Pembimbing

(Ali Mahmudi, B.Eng.PhD)
NIP. P. 1031000429

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
PROGRAM STUDI TEKNIK INFORMATIKA S-1
Jl. Karanglo, Km.2 Malang

FORMULIR BIMBINGAN SKRIPSI

Nama : Niniek Mardiyani
NIM : 1418153
Masa Bimbingan : 25 September 2017 s/d 25 Maret 2018
Judul Skripsi : Penerapan Metode *Finite State Machine* dalam Perancangan Perilaku Musuh pada Game 2d *Horror Girl's Nightmare*

No	Tanggal	Uraian	Paraf Pembimbing
1.	6 Oktober 2017	Konsep game, perbaikan desain karakter dan desain *background.*	
2.	11 Oktober 2017	Laporan BAB I dan BAB II	
3.	30 Oktober 2017	Penambahan pada game	
4.	1 November 2017	Proposal progress, progress pada game	
5.	23 November 2017	Menu awal, prolog, dan BAB III	
6.	30 November 2017	Paper Semhas	
7.	21 Desember 2017	BAB IV dan BAB V	
8.	8 Januari 2018	Demo game	
9.	9 Januari 2018	Pengujian	
10	10 Januari 2018	Laporan skripsi	

Malang, 24 Januari 2018
Dosen Pembimbing

(Hani Zulfia Zahro, S.Kom.M.Kom)
NIP. P. 1031500480

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
PROGRAM STUDI TEKNIK INFORMATIKA S-1
Jl. Karanglo, Km. 2 Malang

FORMULIR PERBAIKAN SKRIPSI

Dalam pelaksanaan ujian skripsi jenjang Strata 1 Program Studi Teknik Informatika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : NINIEK MARDIYANI
NIM : 14.18.153
JURUSAN : TEKNIK INFORMATIKA S-1
JUDUL : PENERAPAN METODE *FINITE STATE MACHINE* DALAM PERANCANGAN PERILAKU MUSUH PADA GAME 2D *HORROR GIRL 'S NIGHTMARE*

No	Penguji	Tanggal	Uraian	Paraf
1.	Penguji I	15 Januari 2018	1. Lihat Laporan untuk Revisi	
2.	Penguji II	15 Januari 2018	1. Kata pengantar 2. Kebutuhan Fungsional dan Non Fungsional 3. *Story Board* 4. Menambahkan *Use Case* 5. Diagram FSM	

Dosen Penguji I

Sandy Natali Mantja, S. Kom
NIP.P. 1030800418

Dosen Penguji II

Agung Panji Sasmito, S.Pd, M.Pd
NIP. P. 1031500499

Dosen Pembimbing I

Ali Mahmudi, B.Eng.PhD
NIP. P. 1031000429

Dosen Pembimbing II

Hani Zulfia Zahro', S.Kom, M.Kom
NIP.P. 1031500480

ANGKET PENGUJIAN USER

PENERAPAN METODE *FINITE STATE MACHINE*
DALAM PERANCANGAN PERILAKU MUSUH
PADA GAME 2D HORROR *GIRL'S NIGHTMARE*

Nama : J Koylana Erceva
NIM : 14.18.001
Jurusan : T. Informatika S1

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	✓		
2	Desain Animasi pada Game		✓	
3	Kontrol pada game		✓	
4	Fitur game		✓	
5	Informasi game (Cerita, Narasi Game)	✓		
6	Game sudah menarik		✓	
7	Fungsi game berjalan	✓		
8	Button pada menu menarik		✓	
9	Kesulitan pada game	✓		
10	Game menyenangkan		✓	

Malang, November 2017

(J Koylana E.)

ANGKET PENGUJIAN USER

PENERAPAN METODE *FINITE STATE MACHINE*
DALAM PERANCANGAN PERILAKU MUSUH
PADA GAME 2D HORROR *GIRL'S NIGHTMARE*

Nama : IKHSAN MA'RIF
NIM : 161811g
Jurusan : Teknik Informatika

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	✓		
2	Desain Animasi pada Game		✓	
3	Kontrol pada game	✓		
4	Fitur game	✓		
5	Informasi game (Cerita, Narasi Game)	✓		
6	Game sudah menarik	✓		
7	Fungsi game berjalan	✓		
8	Button pada menu menarik		✓	
9	Kesulitan pada game	✓		
10	Game menyenangkan		✓	

Malang, November 2017

(IKHSAN MA'RIF)

ANGKET PENGUJIAN USER

PENERAPAN METODE *FINITE STATE MACHINE*
DALAM PERANCANGAN PERILAKU MUSUH
PADA GAME 2D HORROR *GIRL'S NIGHTMARE*

Nama : Kiky Andriani Putri
NIM : 161811
Jurusan : T. Informatika S1

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	✓		
2	Desain Animasi pada Game		✓	
3	Kontrol pada game	✓		
4	Fitur game	✓		
5	Informasi game (Cerita, Narasi Game)	✓		
6	Game sudah menarik	✓		
7	Fungsi game berjalan	✓		
8	Button pada menu menarik		✓	
9	Kesulitan pada game	✓		
10	Game menyenangkan		✓	

Malang, November 2017

(kiky andriani)

ANGKET PENGUJIAN USER

PENERAPAN METODE *FINITE STATE MACHINE*
DALAM PERANCANGAN PERILAKU MUSUH
PADA GAME 2D HORROR *GIRL'S NIGHTMARE*

Nama : YOLANDA APRILIA PUTRI KARTIKASARI
NIM : 1718024
Jurusan : TEKNIK INFORMATIKA S-1

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game		✓	
2	Desain Animasi pada Game	✓		
3	Kontrol pada game	✓		
4	Fitur game	✓		
5	Informasi game (Cerita, Narasi Game)	✓		
6	Game sudah menarik	✓		
7	Fungsi game berjalan	✓		
8	Button pada menu menarik	✓		
9	Kesulitan pada game	✓		
10	Game menyenangkan		✓	

Malang, November 2017

(YOLANDA)

ANGKET PENGUJIAN USER

PENERAPAN METODE *FINITE STATE MACHINE*
DALAM PERANCANGAN PERILAKU MUSUH
PADA GAME 2D HORROR *GIRL'S NIGHTMARE*

Nama : Ismira Drani Putri
NIM : 1513078
Jurusan : Teknik Industri S1

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	✓		
2	Desain Animasi pada Game	✓		
3	Kontrol pada game		✓	
4	Fitur game		✓	
5	Informasi game (Cerita, Narasi Game)	✓		
6	Game sudah menarik		✓	
7	Fungsi game berjalan	✓		
8	Button pada menu menarik	✓		
9	Kesulitan pada game		✓	
10	Game menyenangkan	✓		

Malang, November 2017

(Ismira drani)

ANGKET PENGUJIAN USER

PENERAPAN METODE *FINITE STATE MACHINE*
DALAM PERANCANGAN PERILAKU MUSUH
PADA GAME 2D HORROR *GIRL'S NIGHTMARE*

Nama : Rima Ngndia
NIM : 1113036
Jurusan : T. Industri

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	✓		
2	Desain Animasi pada Game	✓		
3	Kontrol pada game	✓		
4	Fitur game		✓	
5	Informasi game (Cerita, Narasi Game)	✓		
6	Game sudah menarik	✓		
7	Fungsi game berjalan	✓		
8	Button pada menu menarik	✓		
9	Kesulitan pada game	✓		
10	Game menyenangkan	✓		

Malang, November 2017

( )

ANGKET PENGUJIAN USER

PENERAPAN METODE *FINITE STATE MACHINE*
DALAM PERANCANGAN PERILAKU MUSUH
PADA GAME 2D HORROR *GIRL'S NIGHTMARE*

Nama : Lily Karuniasari
NIM : 1418164
Jurusan : Teknik Informatika S1

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	✓		
2	Desain Animasi pada Game	✓		
3	Kontrol pada game	✓		
4	Fitur game	✓		
5	Informasi game (Cerita, Narasi Game)		✓	
6	Game sudah menarik	✓		
7	Fungsi game berjalan		✓	
8	Button pada menu menarik	✓		
9	Kesulitan pada game		✓	
10	Game menyenangkan	✓		

Malang, November 2017

(Lily K.)

ANGKET PENGUJIAN USER

PENERAPAN METODE *FINITE STATE MACHINE*
DALAM PERANCANGAN PERILAKU MUSUH
PADA GAME 2D HORROR *GIRL'S NIGHTMARE*

Nama : Sri Devi Rukmana
NIM : 1513092
Jurusan : Teknik Industri S-1

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	✓		
2	Desain Animasi pada Game		✓	
3	Kontrol pada game	✓		
4	Fitur game	✓		
5	Informasi game (Cerita, Narasi Game)	✓		
6	Game sudah menarik		✓	
7	Fungsi game berjalan		✓	
8	Button pada menu menarik	✓		
9	Kesulitan pada game		✓	
10	Game menyenangkan	✓		

Malang, November 2017

(SRI DEVI)

ANGKET PENGUJIAN USER

PENERAPAN METODE *FINITE STATE MACHINE*
DALAM PERANCANGAN PERILAKU MUSUH
PADA GAME 2D HORROR *GIRL'S NIGHTMARE*

Nama : Rachmatia Bestariani
NIM : 15 13 039
Jurusan : Teknik Industri S-1

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game		✓	
2	Desain Animasi pada Game		✓	
3	Kontrol pada game	✓		
4	Fitur game	✓		
5	Informasi game (Cerita, Narasi Game)		✓	
6	Game sudah menarik		✓	
7	Fungsi game berjalan		✓	
8	Button pada menu menarik	✓		
9	Kesulitan pada game		✓	
10	Game menyenangkan	✓		

Malang, November 2017

(Rachmatia B.)

ANGKET PENGUJIAN USER

PENERAPAN METODE *FINITE STATE MACHINE*
DALAM PERANCANGAN PERILAKU MUSUH
PADA GAME 2D HORROR *GIRL'S NIGHTMARE*

Nama : Jenike Gracelya Noke
NIM : 1918081
Jurusan : T. Informatika

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	✓		
2	Desain Animasi pada Game	✓		
3	Kontrol pada game	✓		
4	Fitur game		✓	
5	Informasi game (Cerita, Narasi Game)	✓		
6	Game sudah menarik	✓		
7	Fungsi game berjalan		✓	
8	Button pada menu menarik	✓		
9	Kesulitan pada game		✓	
10	Game menyenangkan	✓		

Malang, November 2017

(Jenike G.Noke)

Lampiran 1. Source Code Enemy.cs

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

public class enemy : character
{

	private IenemyStates currentState;

	public GameObject Player;
	private float distanceBetween;

	public GameObject target { get; set;}
	// coba ini public Transform _player;

	[SerializeField]
	private float meleeRange;

	[SerializeField]
	private float throwRange;

	[SerializeField]
	private Transform Leftedge;

	[SerializeField]
	private Transform Rightedge;


	public bool InMeleeRange
	{
		get
		{
			if (target != null)
			{
				return  Vector2.Distance  (transform.position,
target.transform.position)<= meleeRange;
			}
			return false;
		}
	}

	public bool InThrowRange
	{
		get
		{
			if (target != null)
			{
				return  Vector2.Distance  (transform.position,
target.transform.position) <= throwRange;
			}
			return false;
		}
	}

	public override bool IsDead
	{
		get
		{
			return health <= 0;
```

```
		}
	}



	// Use this for initialization
	public override void Start ()
	{
		base.Start ();

		ChangeState (new idleState ());
	}



	// Update is called once per frame
	void Update ()
	{

		if (!IsDead)
		{
			if (!TakingDamage)
			{
				currentState.Execute ();
			}

			LookAtTarget ();
		}

		distanceBetween 		= 		Vector2.Distance
(Player.transform.position, transform.position);
	}

	private void LookAtTarget()
	{
		if (target != null)
		{
			float	xDir	=	target.transform.position.x	-
transform.position.x;

			if	(xDir	<	0	&&	facingRight	||	xDir	>	0	&&
!facingRight)
			{
				ChangeDirection ();
			}
		}
	}
	public void ChangeState(IenemyStates newState)
	{
		if (currentState != null)
		{
			currentState.Exit ();
		}
		currentState = newState;

		currentState.Enter (this);
	}
	public void move()
	{
		if (distanceBetween >= 2) {
```

```
                    if (!Attack) {
                          myanimator.SetFloat ("speed", 1);

                          transform.Translate    (GetDirectioon    ()    *
(movementSpeed * Time.deltaTime));
                    }
              }
       }

       public Vector2 GetDirectioon()
       {
              return facingRight ? Vector2.right : Vector2.left;
       }

       public override void OnTriggerEnter2D (Collider2D other)
       {
              base.OnTriggerEnter2D (other);
              currentState.OnTriggerEnter (other);

              if (other.tag == "Kelor") {
                    StartCoroutine(TakeDamage());
                    Destroy (other.gameObject);
              }
       }

       public override IEnumerator TakeDamage()
       {
              health -= 10;

              if (!IsDead) {
                    myanimator.SetTrigger ("damage");
                    yield return null;
              }
              else
              {
                    myanimator.SetTrigger ("die");
                    yield return null;
              }
              Debug.Log ("Damaged");
       }

}
```

Lampiran 2. Source Code Player.cs

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

public delegate void DeadEventHandler();

public class player : character

{
       private static player instance;
       //private player currentState;

       public event DeadEventHandler Dead;
```

```
	public static player Instance
	{
		get
		{
			if (instance == null)
			{
				instance = GameObject.FindObjectOfType<player>
();
			}
			return instance;

		}

	}


	[SerializeField]
	private Transform[] groundPoints;

	[SerializeField]
	private float groundRadius;

	[SerializeField]
	private LayerMask whatIsGround;

	[SerializeField]
	private bool airControl;

	[SerializeField]
	private float jumpForce;

	private bool immortal = false;

	private SpriteRenderer spriteRenderer;

	[SerializeField]
	private float immortaltime;

	public Rigidbody2D MyRigidbody { get; set;}

	public bool Jump { get; set;}

	public bool OnGround { get; set;}

	public override bool IsDead
	{
		get
		{
			if (health <= 0)
			{
				onDead ();
			}

			return health <= 0;
		}
	}

	private Vector2 startPos;

	// Use this for initialization
```

```
	public override void Start ()
	{
		base.Start ();
		startPos = transform.position;
		spriteRenderer = GetComponent<SpriteRenderer> ();
		MyRigidbody = GetComponent<Rigidbody2D> ();
	}

	void Update()
	{
		if (!TakingDamage && !IsDead)
		{
			if (transform.position.y <= -14f) {
				MyRigidbody.velocity = Vector2.zero;
				transform.position = startPos;
			}
			HandleInput ();
		}
//		if (!IsDead)
//		{
//			if (!TakingDamage)
//			{
//				myanimator.SetTrigger ("damage");
//			}
//
//			myanimator.SetLayerWeight (1, 0);
//			myanimator.SetTrigger ("die");
//		}
//		yield return null;
	}

	// Update is called once per frame
	void FixedUpdate ()
	{
		float horizontal = Input.GetAxis ("Horizontal");

		OnGround = IsGrounded ();
		if (!TakingDamage && !IsDead)
		{
			HandleMovement (horizontal);

			Flip (horizontal);

			HandleLayers ();
		}

	}

	public void onDead()
	{
		if (Dead != null)
		{
			Dead ();
		}
	}

	private void HandleMovement(float horizontal)
	{
		if (MyRigidbody.velocity.y < 0)
		{
```

```
                        myanimator.SetBool ("land", true);
                }
                if (!Attack && (OnGround || airControl)) //kalau bisa ini
yangdiperbaiki nik! tambahkan ||aircontrol
                {
                        MyRigidbody.velocity = new Vector2 (horizontal *
movementSpeed, MyRigidbody.velocity.y);
                }
                if (Jump && MyRigidbody.velocity.y == 0)
                {
                        MyRigidbody.AddForce (new Vector2 (0, jumpForce));

                }
                myanimator.SetFloat ("speed", Mathf.Abs (horizontal));
        }

        private void HandleInput()
        {
                if (Input.GetKeyDown (KeyCode.Space))
                {
                        myanimator.SetTrigger ("jump");
//                      Debug.Log ("Jumped");
                }
                if (Input.GetKeyDown (KeyCode.LeftShift))
                {
                        myanimator.SetTrigger ("attack");
                }
                if (Input.GetKeyDown (KeyCode.LeftControl))
                {
                        myanimator.SetTrigger ("throw");
                }
        }

        private void Flip(float horizontal)
        {
                if (horizontal > 0 && !facingRight || horizontal < 0 &&
facingRight)
                {
                        ChangeDirection ();
                }
        }

        private bool IsGrounded()
        {
                if (MyRigidbody.velocity.y <= 0)
                {
                        foreach (Transform point in groundPoints)
                        {
                                Collider2D[] colliders =
Physics2D.OverlapCircleAll (point.position, groundRadius, whatIsGround);

                                for (int i = 0; i < colliders.Length; i++)
                                {
                                        if (colliders [i].gameObject !=
gameObject)
                                        {

                                                return true;
                                        }
                                }
```

```
                }
        }
        return false;
}

private void HandleLayers()
{
        if (!OnGround) {
                myanimator.SetLayerWeight (1, 1);
        }
        else
        {
                myanimator.SetLayerWeight (1, 0);
        }
}

public override void ThrowKelor (int value)
{
        if (!OnGround && value == 1 || OnGround && value == 0)
        {
                base.ThrowKelor (value);
        }

}

private IEnumerator indicateImmortal()
{
        while (immortal)
        {
                spriteRenderer.enabled = false;

                yield return new WaitForSeconds (.1f);

                spriteRenderer.enabled = true;

                yield return new WaitForSeconds (.1f);
        }
}
public override void OnTriggerEnter2D (Collider2D other)
{
        base.OnTriggerEnter2D (other);
        //currentState.OnTriggerEnter (other);

        if (other.tag == "enemyAttack") {
                StartCoroutine(TakeDamage());
                Destroy (other.gameObject);
        }
}
public override IEnumerator TakeDamage ()
{
        //if (!immortal) {
                health -= 10;

                if (!IsDead)
        {
                        myanimator.SetTrigger ("damage");
                        immortal = true;

                        StartCoroutine (indicateImmortal ());
                        yield      return      new      WaitForSeconds
```

```
(immortaltime);

                        immortal = false;
                }
            else
            {
                        myanimator.SetLayerWeight (1, 0);
                        myanimator.SetTrigger ("die");
                        yield return null;
                        StartCoroutine (respawn ());
                }

                PlayerPrefs.SetInt ("health", health);


            }
        IEnumerator respawn()
        {
            yield return new WaitForSeconds (1f);
            transform.position = startPos;
            health = 100;
            myanimator.SetTrigger ("damage");
            base.Start ();
        }



//      public override IEnumerator TakeDamage()
//      {
//          health -= 10;
//
//          if (!IsDead) {
//              myanimator.SetTrigger ("damage");
//              yield return null;
//          }
//          else
//          {
//              myanimator.SetTrigger ("die");
//              yield return null;
//          }
//          Debug.Log ("Damaged");
//      }

}
```

Lampiran 3. Source Character.cs

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

public abstract class character : MonoBehaviour {


    [SerializeField]
    protected Transform kelorPos;

    [SerializeField]
```

```
	protected float movementSpeed;

	protected bool facingRight;

	[SerializeField]
	private GameObject kelorprefab;

	[SerializeField]
	protected int health;

	[SerializeField]
	private EdgeCollider2D Enemyattack;

	[SerializeField]
	private List<string> damageSources;

	public abstract bool IsDead{ get;}

	public bool Attack { get; set;}

	public bool TakingDamage { get; set;}

	public Animator myanimator { get; private set;}

	public abstract IEnumerator TakeDamage ();

	// Use this for initialization
	public virtual void Start ()
	{
		facingRight = true;

		myanimator = GetComponent<Animator> ();

	}

	// Update is called once per frame
	void Update ()
	{

	}



	public void ChangeDirection()
		{
			facingRight = !facingRight;
			transform.localScale	=	new	Vector3
(transform.localScale.x * -1, 1, 1);
		}

	public virtual void ThrowKelor(int value)
	{
		if (facingRight) {
			GameObject	tmp	=	(GameObject)Instantiate
(kelorprefab, kelorPos.position, Quaternion.Euler (new Vector3 (0, 0, -
90)));
			tmp.GetComponent<kelor>		().Initialize
(Vector2.right);
		} else {
			GameObject	tmp	=	(GameObject)Instantiate
```

```
(kelorprefab, kelorPos.position, Quaternion.Euler (new Vector3 (0, 0,
90)));
                    tmp.GetComponent<kelor> ().Initialize (Vector2.left);
             }

      }

      //public void meleeAttack()
      //{
      //     Enemyattack.enabled = !Enemyattack.enabled;
      //}

      public virtual void OnTriggerEnter2D(Collider2D other)
      {
             if (damageSources.Contains(other.tag))
             {
                    StartCoroutine (TakeDamage ());
             }

      }
}
```

Lampiran 4. Source Collisiontrigger.cs

```
using UnityEngine;
using System.Collections;

public class collisiontrigger : MonoBehaviour {

      private BoxCollider2D playerCollider;

      [SerializeField]
      private BoxCollider2D platformCollider;

      [SerializeField]
      private BoxCollider2D platformTrigger;

      // Use this for initialization
      void Start () {

             playerCollider                                          =
GameObject.Find("1").GetComponent<BoxCollider2D> ();
             Physics2D.IgnoreCollision(platformCollider,
platformTrigger, true);
      }

      void OnTriggerEnter2D(Collider2D other)
      {
             if (other.gameObject.name == "1")
             {
                    Physics2D.IgnoreCollision     (platformCollider,
playerCollider, true);
             }
      }
      void OnTriggerExit2D(Collider2D other)
      {
```

```
            if (other.gameObject.name == "1")
            {
                  Physics2D.IgnoreCollision     (platformCollider,
playerCollider, false);
            }
      }
}
```

Lampiran 5. Source Code enemysight.cs

```
using UnityEngine;
using System.Collections;

public class enemysight : MonoBehaviour {

      [SerializeField]
      private enemy enemy;

      void OnTriggerEnter2D(Collider2D other)
      {
            if (other.tag == "Player")
            {
                  enemy.target = other.gameObject;
            }
      }

      void OnTriggerExit2D (Collider2D other)
      {
            if (other.tag == "Player")
            {
                  enemy.target = null;
            }

      }
}
```

Lampiran 6. Source Code Camerafollow.cs

```
using UnityEngine;
using System.Collections;

public class camerafollow : MonoBehaviour {

      [SerializeField]
      private float xMax;

      [SerializeField]
      private float yMax;

      [SerializeField]
```

```
	private float xMin;

	[SerializeField]
	private float yMin;

	private Transform target;

	// Use this for initialization
	void Start ()
	{
		target = GameObject.Find ("1").transform;

	}

	// Update is called once per frame
	void LateUpdate ()
	{
		transform.position = new Vector3 (Mathf.Clamp
(target.position.x, xMin, xMax), Mathf.Clamp (target.position.y, yMin,
yMax), transform.position.z);

	}
}
```

Lampiran 7. Source Coin.cs

```
using UnityEngine;
using System.Collections;

public class coin : MonoBehaviour {

	// Use this for initialization
	void Start () {
		PlayerPrefs.SetInt ("X", 0);
	}

	// Update is called once per frame
	void Update () {

	}

	void OnTriggerEnter2D (Collider2D other)
	{
		if (other.gameObject.tag == "Player")
		{
			gamemanager.Instance.CollectedCoins++;
			PlayerPrefs.SetInt ("X", PlayerPrefs.GetInt ("X") +
1);
			Destroy (this.gameObject);
		}
	}
}
```

Lampiran 8. Source healthbar.cs

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

public class healthbar : MonoBehaviour {

	// Use this for initialization
	void Start () {
		PlayerPrefs.SetInt ("health", 100);
	}

	// Update is called once per frame
	void Update ()
	{
		gameObject.GetComponent<Text>().text=
PlayerPrefs.GetInt("health").ToString()+" %";
		if (PlayerPrefs.GetInt ("health") <= 0)
		{
			Application.LoadLevel (4);
		}

	}
}
```

Lampiran 9. kelor.cs

```
using UnityEngine;
using System.Collections;

[RequireComponent(typeof(Rigidbody2D))]

public class kelor : MonoBehaviour
{
	[SerializeField]
	private float speed;

	private Rigidbody2D myRigidbody;

	private Vector2 direction;

	// Use this for initialization
	void Start () {

		myRigidbody = GetComponent<Rigidbody2D> ();
		//direction = Vector2.right;
	}

	void FixedUpdate()
	{
		myRigidbody.velocity = direction * speed;
	}


    public void Initialize(Vector2 direction)
	{
		this.direction = direction;
	}
```

```
        void OnBecameInvisible()
        {
                Destroy (gameObject);
        }

//
//              if (o.tag == "Enemy") {
//
        o.gameObject.GetComponentInParent<enemy>().TakeDamage();
//              }
//      }
}
```