TUGAS AKHIR – KI1502

Implementasi Prinsip Object-Oriented Design Pada Elemen-Elemen dalam Permainan RPG 'The Bloodline'

Aliya Fathma Najihati
NRP. 5113 100 012

Dosen Pembimbing 1
Imam Kuswardayan, S.Kom., MT.

Dosen Pembimbing 2
Sarwosri, S.Kom., MT.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KI1502

Implementasi Prinsip Object-Oriented Design (OOD) Pada Elemen-Elemen dalam Permainan RPG 'The Bloodline'

Aliya Fathma Najihati
NRP. 5113 100 012

Dosen Pembimbing 1
Imam Kuswardayan, S.Kom., MT.

Dosen Pembimbing 2
Sarwosri, S.Kom., MT.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – KI1502

IMPLEMENTATION OF OBJECT-ORIENTED DESIGN (OOD) PRINCIPLE ON ELEMENTS INSIDE OF RPG 'THE BLOODLINE'

Aliya Fathma Najihati
NRP. 5113 100 012

Supervisor 1
Imam Kuswardayan, S.Kom., MT.

Supervisor 2
Sarwosri, S.Kom., MT.

DEPARTMENT OF INFORMATICS
Faculty of Information and Communication Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

Implementasi Prinsip Object-Oriented Design (OOD) pada Elemen-Elemen dalam Permainan RPG 'The Bloodline'

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Interaksi Grafika dan Seni
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
Aliya Fathma Najihati
NRP. 5113100012

Disetujui oleh Pembimbing Tugas Akhir:

1. Imam Kuswardayan, S.Kom., M.T. ..... (Pembimbing 1)
(NIP. *197612152003121001*)

2. Sarwosri., S.Kom., MT. ..... (Pembimbing 2)
(NIP. *197608092001122001*)

SURABAYA
Maret, 2018

(Halaman ini sengaja dikosongkan)

Implementasi Prinsip Object-Oriented Design pada Elemen-Elemen dalam Permainan RPG 'The Bloodline'

Nama : **Aliya Fathma Najihati**
NRP : **5113100012**
Departemen : **Informatika**
Fakultas Teknologi Informasi dan Komunikasi ITS
Dosen Pembimbing I : **Imam Kuswardayan, S.Kom., MT.**
Dosen Pembimbing II : **Sarwosri, S.Kom., MT.**

ABSTRAK

Bermain game adalah suatu aktivitas yang sekarang ini banyak dilakukan oleh kalangan remaja dan dewasa. Salah satunya adalah permainan RPG. Permainan RPG merupakan jenis permainan yang tergolong memiliki elemen game dalam jumlah banyak yang saling berkaitan. Hal ini sering membuat developer melakukan kesalahan dalam proses pengembangannya dikarenakan banyaknya fungsi pada elemen-elemen tersebut yang bersifat sama atau mirip dengan satu sama lain yang berada di beberapa tempat yang berbeda.

Prinsip Object-Oriented Design menjadi salah satu solusi tepat untuk mengatasi masalah tersebut. Penggunaan prinsip Object-Oriented Design efektif dalam menghilangkan dan meminimalisir pengulangan fungsi mirip atau sama yang sering terjadi pada perancangan secara konvensional.

Tujuan dari pengerjaan Tugas Akhir ini adalah dapat menghasilkan permainan RPG dimana elemen-elemennya didesain menggunakan prinsip Object-Oriented Design untuk mengefektifkan kode sumber dan sistem secara keseluruhan.

Berdasarkan hasil uji coba, penggunaan prinsip Object-Oriented Design lebih efektif dan efisien karena sifatnya yang tidak repetitif.

Kata kunci: Game, RPG, GameMaker, Object-Oriented Design.

IMPLEMENTATION OF OBJECT-ORIENTED DESIGN PRINCIPLE ON ELEMENTS INSIDE OF RPG 'THE BLOODLINE'

Name	**: Aliya Fathma Najihati**
NRP	**: 5113100012**
Department	**: Department of Informatics Faculty of Information and Communication Technology ITS**
Supervisor I	**: Imam Kuswardayan, S.Kom., MT.**
Supervisor II	**: Sarwosri, S.Kom., MT.**

ABSTRACT

Playing game is an activity that is commonly done by teenagers and young adults nowadays. One of the game oftenly played is RPG. Role-Playing Game is a game which has quite a huge number of elements related to one another. That often becomes a reason for developers to make a mistake in developing the game if they're not careful, for there may be similar functions or behaviors which are often written in different places in the game.

Object-Oriented Design (OOD) principle is one of the best methods to solve that issue. The using of Object-Oriented Design (OOD) is deemed to be effective in omitting or minimalizing the abundance and repetition of similar or even same functions in source codes in different places, something that often happens in objects designed in conventional way.

The purpose of making this Final Project is to succesfully produce an RPG of which its elements are designed using the principle of Object-Oriented Design. According to testing result,

implementing Object-Oriented Design principle to the elements does make the system and source codes more effective and efficient due to the fact that Object-Oriented Design makes things non-repetitive.

Key words: Game, RPG, GameMaker, Object-Oriented Design.

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT karena atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul "Implementasi Prinsip Object-Oriented Design pada Elemen-Elemen dalam Permainan RPG 'The Bloodline'".

Pengerjaan tugas akhir ini penulis lakukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Program Studi S-1 Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama proses pengerjaan tugas akhir ini hingga selesai, antara lain:

1. Allah SWT atas segala karunia dan rahmat-Nya yang telah diberikan selama ini.
2. Keluarga penulis, Bapak Tubagus Atmiko, Ibu Hidayati, saudari kembar Aliya Rahma Najihati, dan juga keluarga yang tidak dapat penulis sebutkan satu per satu yang telah memberi dukungan moral dan material serta doa untuk penulis.
3. Bapak Imam Kuswardayan S.Kom., MT. selaku dosen pembimbing I yang telah memberikan bimbingan dan arahan dalam pengerjaan tugas akhir ini.
4. Ibu Sarwosri S.Kom., MT. selaku dosen pembimbing II yang telah memberikan bimbingan dan arahan dalam pengerjaan tugas akhir ini.
5. Teman-teman angkatan 2013 yang memiliki dosen pembimbing yang sama yang telah memberikan dukungan selama saya menyelesaikan Tugas Akhir ini.

6. Seluruh pihak yang tidak bisa saya sebutkan satu persatu yang telah memberikan dukungan selama saya menyelesaikan Tugas Akhir ini.

Penulis menyadarai bahwa masih terdapat banyak kekurangan dalam tugas akhir ini. Oleh karena itu, penulis menerima dengan rendah hati kritik dan saran untuk pembelajaran dan perbaikan ke depannya. Semoga tugas akhir ini dapat memberikan manfaat yang sebaik-baiknya.

Surabaya, Januari 2018

Penulis

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Bagian ini akan dijelaskan hal-hal yang menjadi latar belakang, permasalahan yang dihadapi, batasan masalah, tujuan dan manfaat, metodologi dan sistematika penulisan yang digunakan dalam pembuatan tugas akhir ini.

1.1 Latar Belakang

Aplikasi permainan RPG tergolong suatu pengembangan proyek yang membutuhkan berbagai *resource* yang saling berkaitan. Banyak komponen penting yang tak bisa lepas dari permainan itu sendiri; dimulai dari jenis persenjataan yang dimiliki pemain, skill apa yang dimiliki pemain, jenis-jenis musuh yang menghadang pemain, berbagai jenis item dengan segala macam kegunaannya, dan karakter-karakter Non-Playable Character (NPC) yang membantu pemain dalam menyelesaikan permainan; kesemuanya memiliki atribut masing-masing yang saling berkaitan. Tiap atribut dalam komponen memiliki kemiripan atau pola yang sama. Pada jenis persenjataan, misalnya. Sama-sama merupakan senjata, namun berdasarkan jenisnya, senjata-senjata tersebut memiliki kegunaan yang berbeda.

Implementasi pendekatan Object-Oriented Design pada atribut-atribut pada game dan fungsi-fungsi yang memiliki pola yang sama diyakini mampu mengefektifkan bentuk kodingan di dalam program. Hasil yang diharapkan dalam pengerjaan Tugas Akhir ini adalah terciptanya suatu permainan *single-player* RPG yang teroptimasi, dimana segala *asset* dan *resource* yang berhubungan dengan permainan yang memiliki pola yang sama didesain menggunakan pendekatan Object-Oriented Design.

1.2 Rumusan Masalah

Perumusan masalah yang terdapat pada tugas akhir ini, antara lain:

1. Bagaimana mendesain elemen-elemen dalam permainan RPG The Bloodline yang dibuat dengan menggunakan pendekatan Object-Oriented Design?
2. Bagaimana rancangan aturan main dan scenario dari permainan RPG The Bloodline?
3. Bagaimana mengimplementasikan rancangan ke dalam permainan RPG The Bloodline?

1.3 Batasan Masalah

Batasan masalah yang terdapat pada tugas akhir ini, yaitu sebagai berikut:

1. Elemen-elemen pada permainan RPG meliputi *weapon*, *item*, *armor*, karakter (*hero*, *monster,* dan *NPC*), dan *skill*.
2. Implementasi meggunakan GameMaker Studio: Professional.
3. Dungeon yang diimplementasikan hanya berjumlah 2 *dungeon*.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah membuat aplikasi permainan RPG yang berfokus pada implementasi pendekatan object-oriented design sebagai sarana untuk menciptakan relasi behaviour yang dinamis pada *weapon*, *enemy*, dan *asset* lain di dalamnya.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini, antara lain:

1. Mengimplementasikan prinsip Object-Oriented Design dalam beberapa aspek pada permainan RPG The Bloodline,
2. Memberikan media hiburan bagi pemain.

1.6 Metodologi

1. Penyusunan proposal tugas akhir
   Tahap pertama dalam proses pengerjaan tugas akhir ini adalah menyusun proposal tugas akhir. Pada proposal tugas akhir ini diajukan implementasi prinsip Object-Oriented Design (OOD) pada Elemen-Elemen dalam Permainan RPG 'The Bloodline'.

2. Studi literatur
   Pada tahap ini, akan dicari studi literatur yang relevan untuk dijadikan referensi dalam pengerjaan tugas akhir. Studi literatur ini didapatkan dari buku, internet, dan materi-materi kuliah yang berhubungan dengan metode yang akan digunakan.

3. Analisis dan desain perangkat lunak
   Perangkat lunak yang akan dibangun merupakan aplikasi untuk perangkat desktop. Akan ada dua macam tampilan dari aplikasi ini. Tampilan untuk menu dan tampilan permainan.

4. Pengembangan perangkat lunak
   Pembangunan *game* akan dilakukan dengan menggunakan bahasa pemrograman GML dan Game Engine GameMaker Studio: Professional.

5. Pengujian dan evaluasi

Pengujian yang akan dilakukan adalah pengujian fungsionalitas. Pengujian ini bertujuan untuk memeriksa apakah *game* sudah terimplementasikan atau belum. Kemudian akan dilakukan evaluasi untuk memeriksa hasil implementasi *game*.

6. Penyusunan buku tugas akhir
   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini. Pada tahap ini juga disertakan hasil dari implementasi metode dan algoritma yang telah dibuat. Sistematika penulisan buku tugas akhir ini secara garis besar antara lain:
   1. Pendahuluan
      a. Latar Belakang
      b. Rumusan Masalah
      c. Batasan Tugas Akhir
      d. Tujuan
      e. manfaat
      f. Metodologi
      g. Sistematika Penulisan
   2. Tinjauan Pustaka
   3. Desain dan Implementasi
   4. Pengujian dan Evaluasi
   5. Kesimpulan dan Saran
   6. Daftar Pustaka

1.7 Sistematika Penulisan

Buku tugas akhir ini disusun dengan sistematika penulisan sebagai berikut:

BAB I. PENDAHULUAN

Bab ini berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

BAB II. TINJAUAN PUSTAKA

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

BAB III. ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang yang diangkat dalam pengerjaan tugas akhir.

BAB IV. IMPLEMENTASI

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Bab ini berisi proses implementasi dari setiap kelas pada semua modul.

BAB V. PENGUJIAN DAN EVALUASI

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

BAB VI. KESIMPULAN DAN SARAN

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan metode yang diajukan pada pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Permainan RPG

Role-Playing Game (RPG) adalah sebuah jenis permainan video game dimana pemain memainkan peran dari suatu karakter dan latar belakang yang fiktif. Pemain bertanggung jawab untuk memainkan peran mereka, baik melalui suatu proses *decision-making* yang terstruktur atau pengembangan karakter [6]. Sukses tidaknya segala pilihan dan tindakan yang diambil pemain ditentuan beradasarkan aturan dan alur dari masing-masing game [7]. Sebagai suatu jenis permainan video game, tentunya RPG memiliki sisi positif dan negatif untuk remaja/dewasa yang memainkannya. Sisi positif yang bisa didapat oleh remaja/dewasa diantaranya adalah:

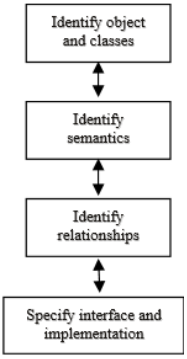
1. Permainan RPG mampu mengasah kreativitas
   RPG memiliki alur yang jelas dengan beberapa kemungkinan yang mengarah pada hasil akhir yang berbeda. Banyaknya pilihan membuat pemain memiliki kebebasan untuk menggerakkan permainan ke arah yang mereka inginkan. RPG memiliki aturan, namun itu semua hanyalah dasar dan pondasi dari permainan itu. Untuk selanjutnya, semua keputusan dan hasil akhir dari permainan tesebut ada di tangan pemain.
2. Permainan RPG mengajarkan kemampuan memecahkan masalah

Pemecahan masalah adalah salah satu hal inti dari RPG. Dalam setiap RPG, selalu ada beberapa lapisan masalah yang harus dihadapi. Masalah tersebut berbeda-beda untuk tiap keputusan, pilihan, dan tindakan yang diambil pemain. Dalam RPG, pemain diajarkan untuk melihat masalah dari berbagai sudut pandang.

2.2 Pendekatan Object-Oriented Design

Object-Oriented Design adalah sebuah proses dimana desainer mensintesis segala kebutuhan yang telah dikumpulkan dalam fase analisis dan mengelompokkan menjadi object-object serta hubungan antara object tersebut [8]. Suatu objek mengandung data yang terenkapsulasi dan prosedur-prosedur yang dikelompokkan untuk merepresentasikan suatu entitas.

Proses mendesain elemen yang banyak digunakan oleh desainer adalah proses analisis dan desain yang dikemukakan oleh Booch, dimana desain tersebut bersifat *incremental* dan iteratif. Meskipun proses ini telah digunakan secara luas, masih banyak detail-detail penting yang perlu diperhatikan oleh desainer untuk merealisasikan setiap prosesnya [8]. Proses analisis dan desain ini dapat dilihat pada Gambar 2.1.



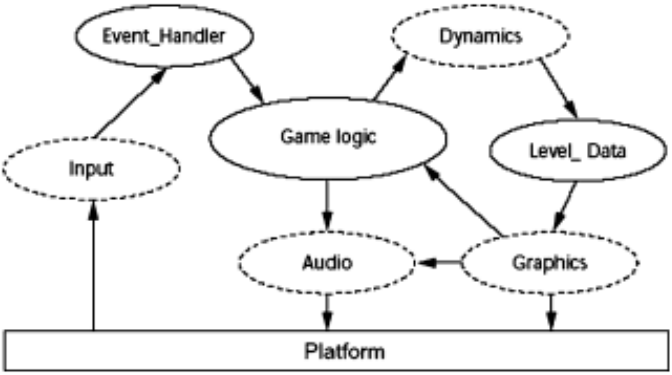
Gambar 2.1. Proses analisis dan desain [9]

Menurut Booch, terdapat 4 tahapan dalam menganalisa dan mendesain secara *object-oriented* [9]. Keempat tahapan tersebut adalah :

1. Identifikasi object dan kelas-kelas yang dibutuhkan,
2. Identifikasi semantic-semantik,
3. Identifikasi hubungan antar object,
4. Spesifikasi tampilan pengguna dan implementasi.

Pemrograman berbasis Object-Oriented sangat efektif dalam menghilangkan perulangan kodingan sehingga memudahkan kita bilamana kita ingin mengganti suatu hal pada kode seperti memperbaiki *bugs*, atau hanya sekedar menambah fitur baru. Penggunaan fungsi bisa memecahkan masalah duplikasi kode (kode yang sama diulang-ulang di berbagai tempat) yang selama ini banyak dilakukan oleh *programmer*.

Keuntungan mengimplementasikan Object-Oriented Design pada RPG adalah : elemen-elemen permainan yang memiliki jumlah banyak dapat didesain berdasarkan kemiripan fungsinya. Hal tersebut dapat meminimalisir kesalahan pembuatan banyak fungsi yang sebenarnya memiliki sifat yang sama atau mirip di beberapa kode sumber berbeda (repetitif). Asitektur pengembangan *game* yang digunakan merujuk pada arsitektur game secara umum pernah dikemukakan oleh Bishop et al, dapat dilihat pada Gambar 2.2.



Gambar 2.2. Arsitektur game secara umum [1]

Pada gambar diatas terdapat 2 jenis bagan : solid dan putus-putus. Aspek yang berada dalam bagan solid berarti aspek-aspek tersebut bersifat penting dan tidak dapat tergantikan dalam suatu game. Sedangkan aspek-aspek yang dibatasi oleh garis putus-putus melambangkan aspek-aspek tersebut bersifat opsional dan biasanya ada di dalam game yang bersifat rumit. *Game logic* merupakan jalan dan inti cerita dari suatu game. *Audio* dan *Graphic* adalah modul-modul yang membantu penulis menarasikan jalan cerita ke pemain. *Event-handler* dan *Input* merupakan modul-modul yang memenuhi jalan cerita dari suatu game berdasarkan *action* yang dilakukan pemain.

2.3 Permainan Serupa

Final Fantasy X11

Final Fantasy XII merupakan referensi game yang saya pakai. Game ini mengajak pemain untuk memainkan seorang

pemuda biasa berpendirian kuat yang kemudian terseret dalam kompleksitas perebutan tahta negara tempat dirinya dilahirkan. Sebagaimana game bertajuk Final Fantasy lainnya, permainan ini memiliki unsur RPG yang sangat kental, mulai dari banyaknya *non-playable character* yang menjadi teman tim pemain, berbagai jenis persenjataan dengan kelebihan dan kekurangannya masing-masing yang bisa dikenakan pemain untuk bertarung, *skill magic* yang membantu pemain dalam pertarungan, poin untuk bertambah level, dan sebagainya.

Aturan bermain :

1. Pemain harus menyelesaikan segala misi dan *quest* yang diberikan sepanjang alur cerita.
2. Saat semua anggota dalam tim mati, game dinyatakan *game over*.

Cara bermain :

1. Pemain bebas bergerak menjelajah area yang sudah disediakan
2. Setiap kali berhasil mengalahkan musuh, pemain akan mendapatkan poin *experience* untuk naik level dan uang untuk digunakan dalam transaksi.
3. Pemain bisa membeli item, senjata, armor.
4. Pemain bisa berkomunikasi dengan *non-playable character* (NPC).

Aspek dalam game dimana *object-oriented design* bisa diimplementasikan :

1. Persenjataan *(Weaponries)*

Gambar 2.3. Contoh berbagai jenis senjata dalam game

Di dalam permainan Final Fantasy XII, ada berbagai jenis persenjataan; mulai dari pedang, palu, kapak, hingga tongkat biasa. Tiap jenis memiliki setidaknya 2 sub-jenis persenjataan. Palu, misalnya. Ada Palu A dan Palu B. Keduanya memiliki *strength point* yang berbeda, namun ada 1 kesamaan. Mereka sama-sama memiliki *strength point* paling besar dibandingkan jenis senjata lainnya. Kesamaan pola dalam mendesain kekuatan persenjataan dapat dilihat. Contoh persenjataan dapat dilihat pada Gambar 2.3.

2. *Item*

Gambar 2.4. Contoh berbagai jenis item dalam game

Item terdiri dari berbagai macam kegunaannya. Ada yang bersifat pengobatan maupun racun. *Item* pengobatan pun tidak hanya terdiri dari 1 jenis. Ada berbagai macam jenis, namun fungsinya sama : mengobati pemain saat dia terluka. Satu-satunya hal yang membedakan adalah sebanyak apa tiap *item* bisa mengobati kondisi kesehatan pemain. Contoh jenis *item* dapat dilihat pada Gambar 2.4.

3. Musuh *(Enemies)*

Di dalam permainan RPG, musuh tidak hanya terdiri dari 1 jenis. Setiap jenis musuh memiliki tingat kesulitan dan kemampuannya sendiri, namun semuanya memiliki satu pola yang sama : menyerang pemain. Semua jenis musuh memiliki objektivitas yang sama, yaitu melukai pemain hingga poin

kesehatan menjadi 0 dan *game over*. Contoh musuh dapat dilihat pada Gambar 2.5.

Gambar 2.5. Contoh jenis musuh dalam game

2.4 GameMaker Studio

GameMaker Studio adalah sebuah *game engine* berbayar yang dikembangkan oleh YoYo Games. GameMaker : Studio sendiri merupakan rilisan queue atau tingkatan dari GameMaker versi sebelumnya yaitu GameMaker 8, dimana di dalam versi tersebut fitur da4n tampilannya kurang lengkap. Untuk bisa mengembangkan *game* di platform yang berbeda, dibutuhkan GameMaker Studio : Professional. Beberapa *platform* yang didukung oleh GameMaker Studio: Professional adalah : Android, iOS, PlayStation 4, Windows, Xbox 360, dan Xbox One.

BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini membahas mengenai perancangan dan pembuatan perangkat lunak. Perangkat lunak yang dibuat pada tugas akhir ini adalah *game* dengan elemen-elemen di dalamnya didesain sedemikian rupa menggunakan prinsip Object-Oriented Design. Aplikasi yang digunakan untuk perancangan adalah aplikasi online *draw.io.*

3.1 Perancangan Permainan

Pada subbab ini akan dibahas skenario permainan dan aturan main pada *game* ini.

3.1.1 Skenario Permainan

Game ini adalah *game* 2D dengan tema petualangan. *Game* ini dimainkan oleh satu orang saja (*single-player)*. Target dari *game* ini adalah menjelajahi semua room dan pada akhirnya mengalahkan musuh besar tanpa kehilangan seluruh *health point* yang berarti *game over*. Pemain dapat melewati musuh tanpa bertarung dengan musuh tersebut untuk menghindari berkurangnya *health point* yang tidak perlu. Musuh dapat berjalan-jalan sendiri dan akan mengejar pemain bila pemain berada dalam radar. Pemain dapat bertarung dengan serangan biasa maupun dengan serangan elemental (syarat berlaku). Pemain juga dapat mengambil dan menggunakan *item* yang sudah diambil.

Pengaturan musuh juga diatur pada tahap ini. Ada 3 jenis musuh, yaitu musuh lemah, musuh sedang, dan musuh kuat. Rancangan musuh dapat dilihat pada Tabel 3.1.

Tabel 3. 1 Rancangan level musuh

Dungeon	Musuh Lemah	Musuh Sedang	Musuh Kuat
1	5	0	0
2	2	3	1

3.1.2 Aturan Permainan

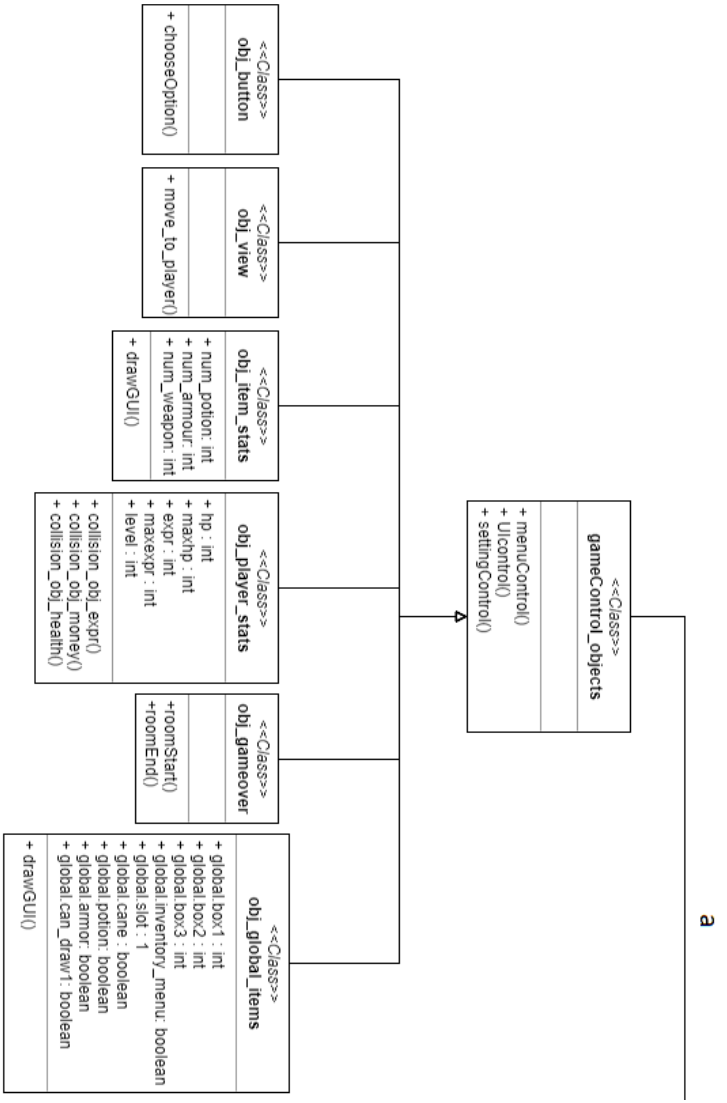
Permainan ini memiliki beberapa aturan main, yaitu:

1. Pada setiap *dungeon* akan ada musuh yang muncul.
2. Pemain dapat menyerang musuh dengan menggunakan serangan fisik atau serangan elemental.
3. Pemain dapat menyerang secara fisik dengan cara menekan tombol keyboard ‘X’ kearah musuh saat musuh dirasa berada dalam jarak jangkauan serang.
4. Pemain hanya dapat mengeluarkan serangan elemental bila pemain telah memiliki senjata tongkat sihir *(cane)*. Untuk dapat mengeluarkan serangan elemental, tekan tombol ‘C’.
5. Pemain dapat menggunakan *item armor* untuk menguatkan *defence* pemain.
6. Pemain dapat memilih untuk mengabaikan musuh.
7. Pemain dan musuh akan melakukan serangan secara otomatis sesuai dengan atribut *speed*, *attack,* dan *defence* dari masing-masing karakter.
8. Atribut *attack* dan *defence* akan mempengaruhi berapakah *damage* yang akan dihasilkan, minimal nilai *damage* adalah 1.
9. Apabila musuh kehilangan seluruh *health*-nya maka musuh akan hancur.
10. Apabila pemain kehilangan seluruh *healthpoint*-nya maka permainan akan kalah.

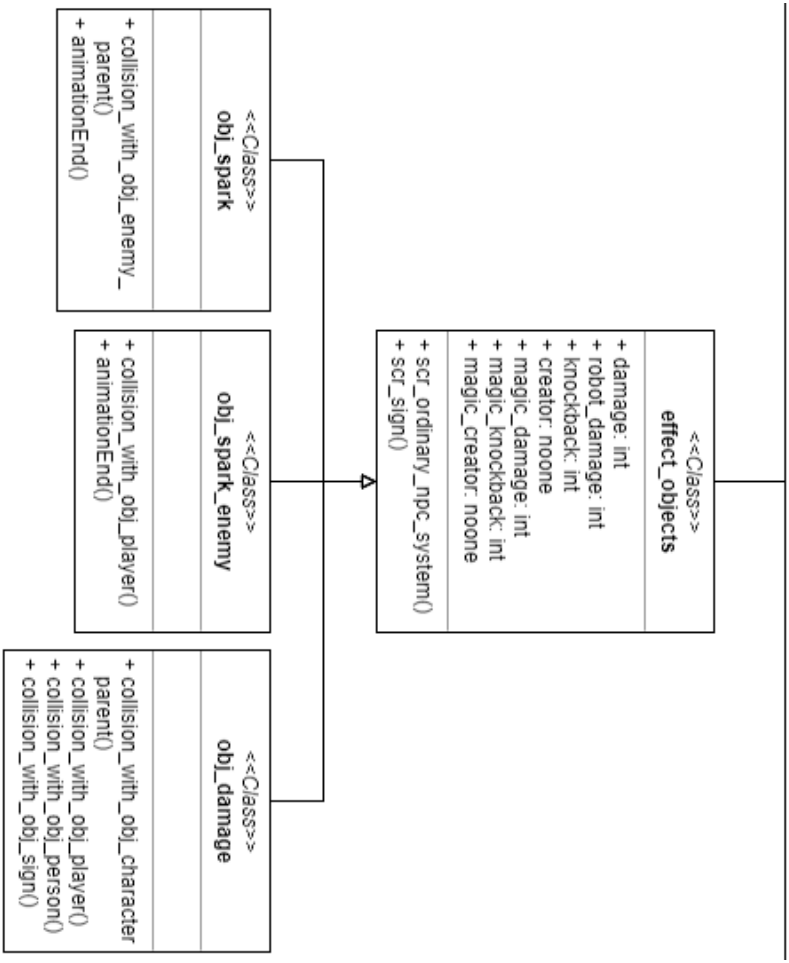
11. Pemain dapat mengambil *healthbar*, *experience point*, dan uang yang secara random dijatuhkan musuh saat musuh berhasil dikalahkan.
12. Apabila *Experience point* (EXP) mencapai nilai maksimum level tersebut, pemain akan naik level.
13. Pemain dapat mengambil item *potion, armor,* dan *cane* yang tergeletak di beberapa *dungeon*.
14. Pemain dapat berbicara dengan *Non-Playable Character* (NPC) yang berada di desa.
15. Permain dapat melakukan transaksi terhadap item yang didapat.

3.2 Perancangan Elemen-Elemen pada Game

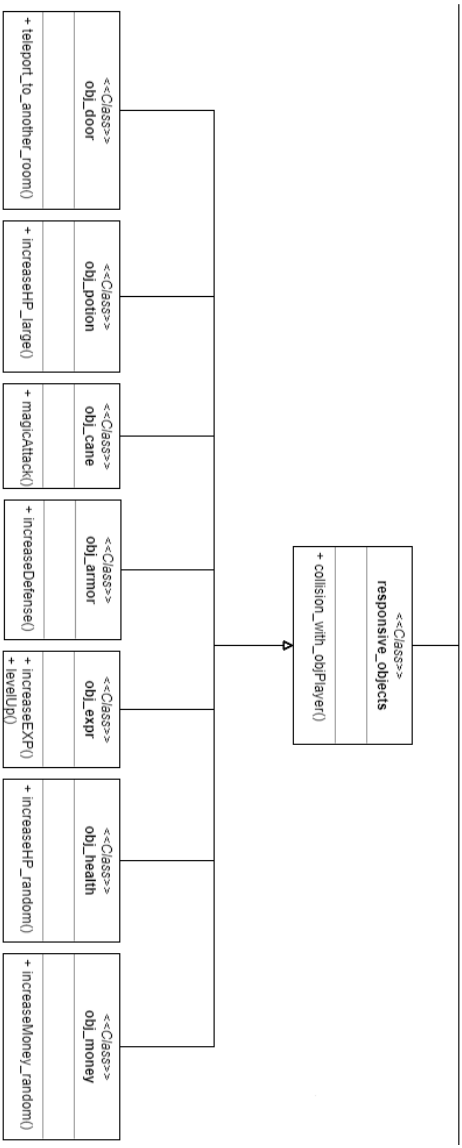
Pada subbab ini akan dibahas rancangan elemen-elemen yang digunakan di dalam *game* dengan menggunakan prinsip Object-Oriented Design. Terdapat 6 *instances* elemen utama *game* yang menjadi *parent* dari elemen-elemen *game* lain dibawahnya yaitu : obj_character_parent, obj_item_parents, obstacle_objects, responsive_objects, effect_objects, dan gameControl_objects. Keenam *child* ini akan memiliki kedua sifat turunan dari *instance* ‘elemen_game’ yaitu (nama dan gambar). Perancangan elemen-elemen pada permainan ini dapat dilihat pada Gambar 3.1, Gambar 3.2, Gambar 3.3, Gambar 3.4, dan Gambar 3.5.



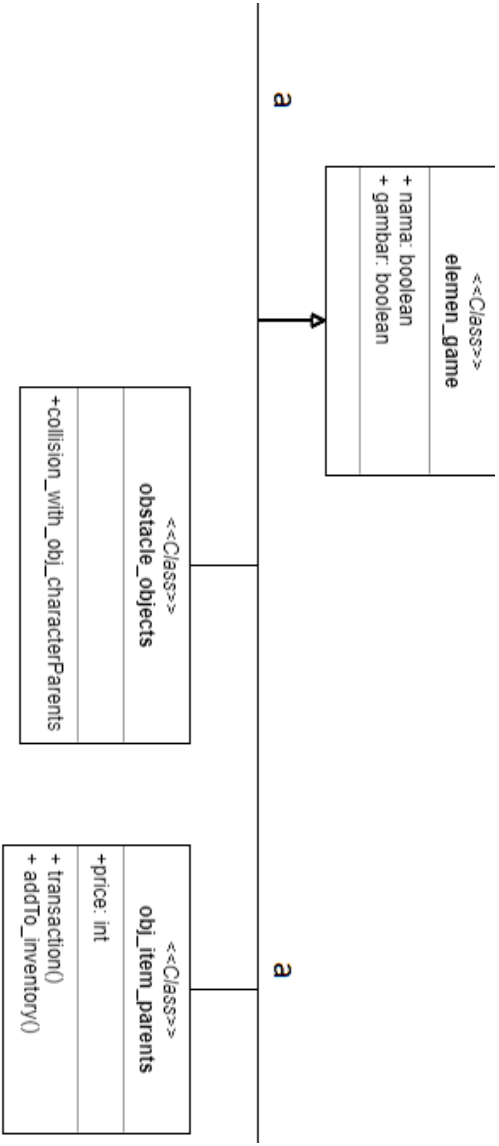
Gambar 3.1. Perancangan desain 'obj_elemen_game' : 'gameControl_objects'



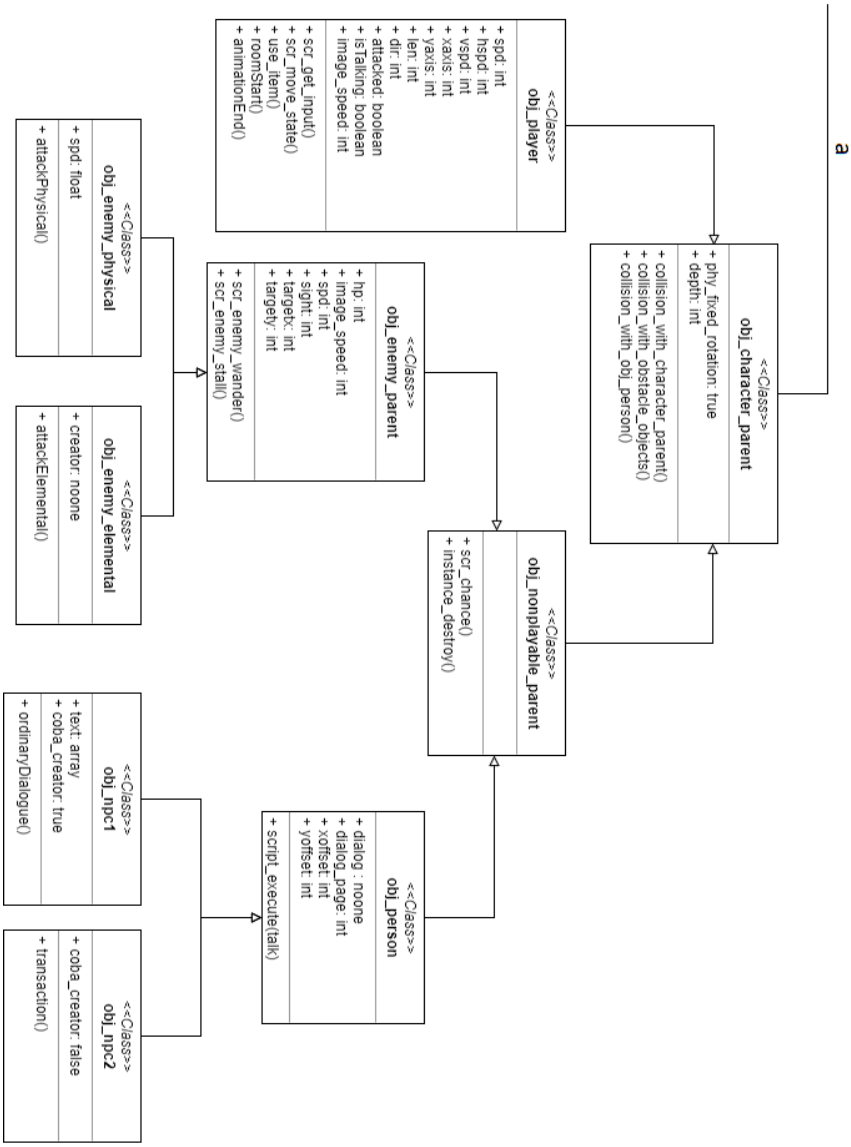
Gambar 3.2. Perancangan desain 'obj_elemen_game' : 'effect_objects'



Gambar 3.3. Perancangan desain 'obj_elemen_game' : 'responsive_objects'

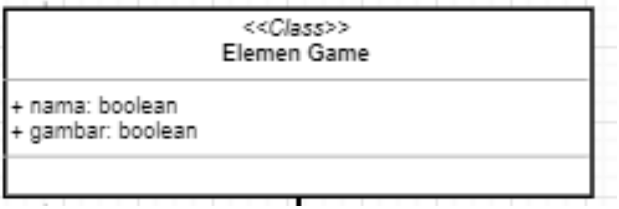


Gambar 3.4. Perancangan desain 'obj_elemen_game' : 'obstacle_objects', 'obj_item_parents'



Gambar 3.5. Perancangan desain 'obj_elemen_game' : 'obj_character_parent'

Seperti yang dapat dilihat, instance ‘elemen_game’ yang menjadi *parent* paling utama *game* ini memiliki 2 sifat yang akan diturunkan oleh semua elemen yaitu *nama* dan *gambar*, yang dapat bernilai *false* atau *true*. Instance ini dapat dilihat pada Gambar 3.6.

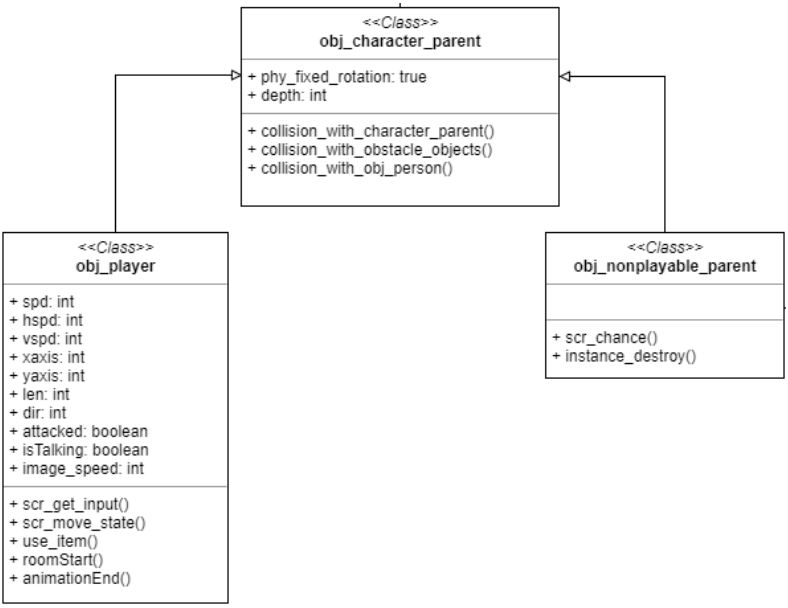


Gambar 3.6. Instance ‘elemen_game’

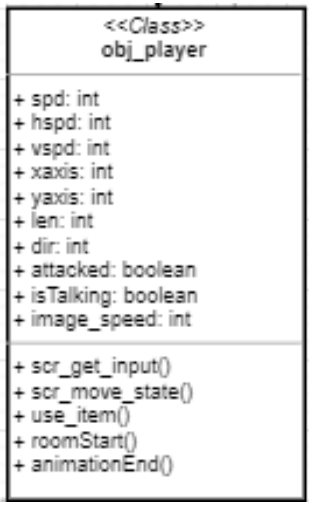
3.2.1 Instance ‘obj_character_parent’

Selain memiliki sifat yang diturunkan oleh instance ‘elemen_game’, instance ‘obj_character_parent’ memiliki beberapa sifat yang diturunkan kepada 2 instance di bawahnya (*child)*. Instance-instance tersebut adalah instance ‘obj_player’ dan instance ‘obj_nonplayable_parent’.

Sifat-sifat tersebut adalah : phy_fixed_rotation, depth, *Collision* event dengan elemen ‘obj_character_parent’, *collision* event dengan elemen ‘obstacle_objects’, dan *collision* event dengan elemen ‘obj_person’.



Gambar 3.7. Instance 'obj_character_parent'



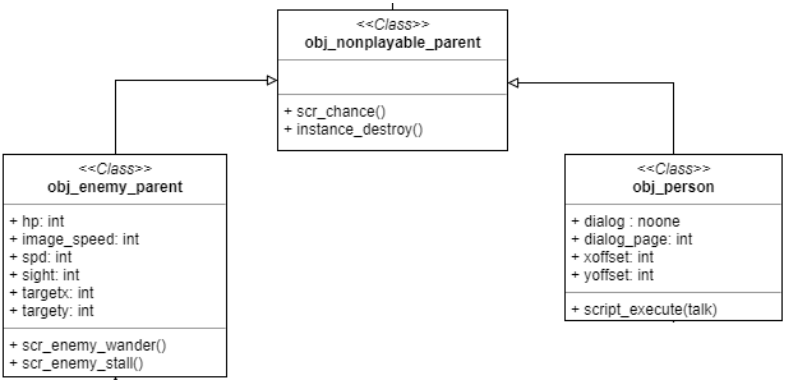
Gambar 3.8. Instance 'obj_player'

3.2.1.1 Instance ‘obj_player’

Selain memiliki sifat yang diturunkan oleh instance ‘obj_character_parent’, instance ‘obj_player’ memiliki beberapa sifat-sifatnya sendiri. Instance ini tidak memiliki anak (*child*). Instance ‘obj_player’ dapat dilihat pada Gambar 3.8.

3.2.1.2 Instance ‘obj_nonplayable_parent’

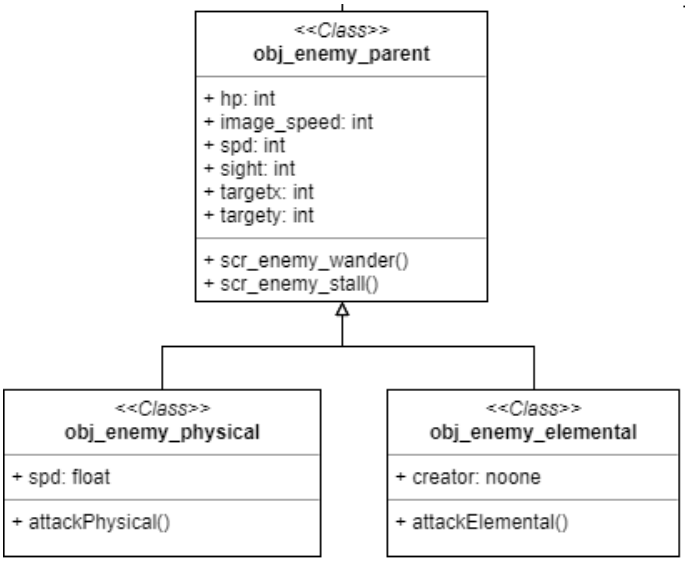
Instance ‘obj_nonplayable_parent’ memiliki sifat-sifat yang diturunkan oleh *parent* dari instance ini yakni ‘obj_character_parent’. Selain itu, instance ‘obj_nonplayable_parent’ memiliki beberapa sifat-yang juga diturunkan ke 2 instance-instance dibawahnya : instance ‘enemy_parent’ dan ‘obj_person’. Kedua instance ini merupakan *children* dari instance ‘obj_nonplayable_parent’. Instance ini dapat dilihat pada Gambar 3.9.



Gambar 3.9. Instance ‘obj_nonplayable_parent’

a) Instance ‘obj_enemy_parent’

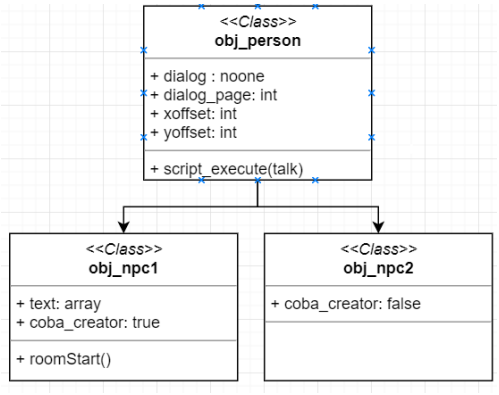
Instance ‘obj_enemy_parent’ memiliki beberapa sifat yang diturunkan oleh instance *parent* ‘obj_nonplayable_parent’. Selain itu, instance ini juga memiliki sifat-sifatnya sendiri yang diturunkan ke 2 instance dibawahnya, yakni instance ‘obj_enemy_physical’ dan ‘obj_enemy_elemental’. Instance in dapat dilihat pada Gambar 3.10.



Gambar 3.10. Instance ‘obj_enemy_parent’

b) Instance ‘obj_person’

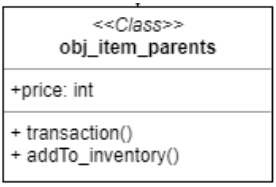
Selain memiliki sifat yang diturunkan oleh instance *parent* ‘obj_nonplayable_parent’, instance ‘obj_person’ memiliki beberapa sifat yang diturunkan pula ke instance *children* ‘obj_npc1’ dan ‘obj_npc2’. Instance ini dapat dilihat pada Gambar 3.11.



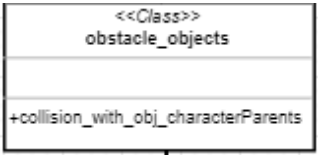
Gambar 3.11. Instance 'obj_person'

3.2.2 Instance 'obj_item_parents'

Selain menurunkan sifat yang dimiliki oleh instance elemen_game, instance 'obj_item_parents' memiliki sifat-sifat yang diturunkan kepada anak-anaknya. Instance ini dapat dilihat pada Gambar 3.12.



Gambar 3.12. Instance 'obj_item_parents'



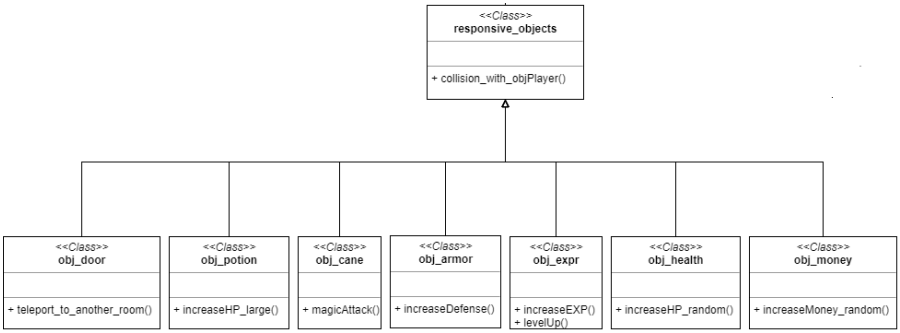
Gambar 3.13. Instance 'obstacle_objetcs'

3.2.3 Instance ‘obstacle_objects’

Instance ini merupakan salah satu *child* dari instance ‘elemen_game’ Selain memiliki sifat yang diturunkan oleh instance *parent* ‘elemen_game’, instance ini memiliki 1 sifat yang diturunkan ke keempat instance *child,* yakni sifat berbenturan dengan instance ‘obj_character_parent’. Instance ‘obstacle_objects’ dapat dilihat pada Gambar 3.13.

3.2.4 Instance ‘responsive_objects’

Instance ‘responsive_objects’ memiliki sifat yang diturunkan oleh instance *parent* ‘elemen_game’. Tidak hanya itu, instance ini memiliki sifat yang diturunkan ketujuh instance *child*. Sebagai object yang bersifat responsif, sifat unik yang hanya dimiliki oleh instance ini adalah sifat saat berbenturan dengan ‘obj_player’. Instance ini dapat dilihat pada Gambar 3.14.

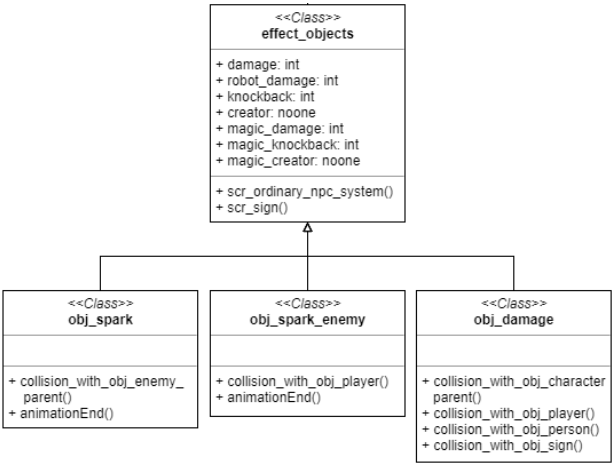


Gambar 3.14. Instance ‘responsive_objects’

3.2.5 Instance ‘effect_objects’

Selain memiliki sifat yang diturunkan oleh instance *parent* ‘elemen_game’, instance ‘effect_objects’ memiliki beberapa sifat yang diturunkan ketiga instance *child* : instance

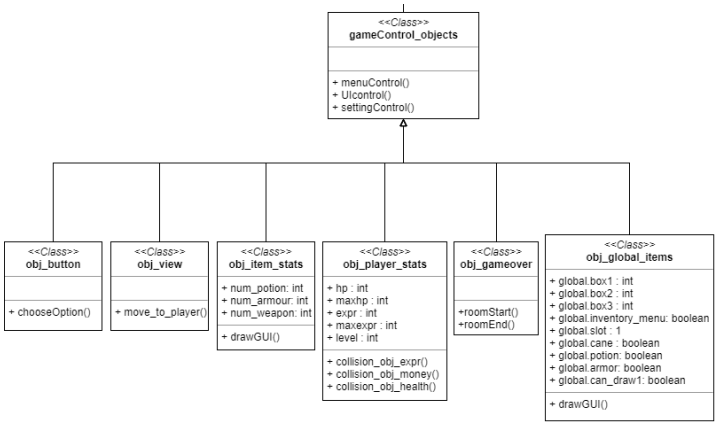
‘obj_spark’, ‘obj_spark_enemy’, ‘obj_damage’. Instance ini dapat dilihat pada Gambar 3.15.



Gambar 3.15. Instance ‘effect_objects’

3.2.6 Instance ‘gameControl_objects’

Selain memiliki sifat yang diturunkan oleh instance parent ‘elemen_game’, instance ‘gameControl_objects’ memiliki beberapa sifat yang diturunkan ketiga instance child : instance ‘obj_view’, ‘obj_gameover’, dan ‘obj_global_items’. Instance ini merupakan instance yang bertugas mengontrol lingkungan permainan. Instance ‘gameControl_objects’ dapat dilihat pada Gambar 3.16.



Gambar 3.16. Instance 'gameControl_objects'

3.3 Perancangan Tampilan Antarmuka

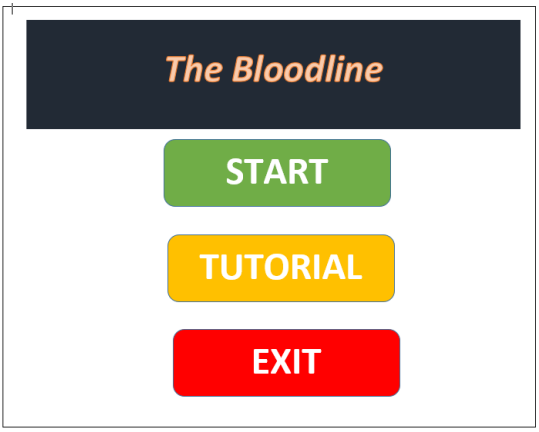
Subbab ini membahas bagaimana rancangan antarmuka pengguna yang akan digunakan untuk tugas akhir. Dalam *game* ini terdapat beberapa tampilan, yaitu tampilan awal (menu utama), tampilan permainan, tampilan saat menang, dan tampilan saat *gameover*.

3.3.1 Tampilan Awal

Tampilan awal merupakan halaman yang pertama kali muncul ketika aplikasi dijalankan. Pada halaman ini terdapat tiga tombol, yaitu tombol *start*, tombol *tutorial*, dan tombol *exit*. Rancangan antarmuka tampilan awal dapat dilihat pada Gambar 3.17.

Fungsi tiga tombol pada halaman awal, yaitu:

1. *Start* yang berfungsi untuk memulai permainan.
2. *Tutorial* yang berfungsi untuk melihat instruksi permainan.
3. *Exit* yang berfungsi untuk keluar dari aplikasi permainan.



Gambar 3.17 Rancangan tampilan awal

3.3.2 Tampilan Tutorial

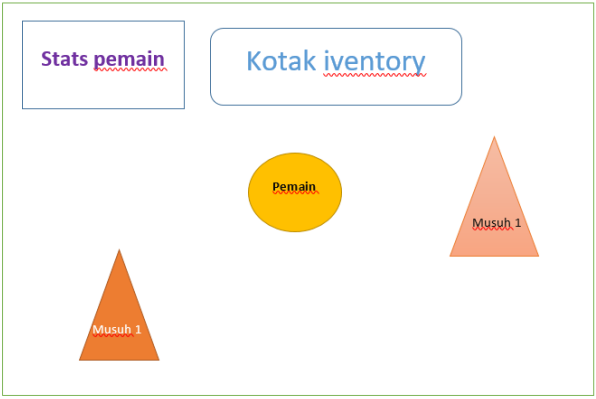
Halaman ini berisikan tentang petunjuk untuk mengendalikan karakter. Rancangan tampilan instruksi dapat dilihat pada Gambar 3.18. Petunjuk yang tampilkan berupa petunjuk untuk menggerakkan karakter dan petunjuk tentang aturan permainan.

Gambar 3.18. Rancangan tampilan tutorial

3.3.3 Tampilan Permainan

Tampilan permainan ialah halaman saat pemain telah menekan tombol *start*. Rancangan Tampilan Permainan dapat dilihat pada Gambar 3.19. Pada layar terdapat beberapa hal yaitu:

1. Stats pemain (dalam bentuk text) yang menunjukkan jumlah *health point*, level pemain saat itu, dan jumlah uang yang telah didapat terletak di pojok kiri atas.
2. Kotak inventory yang menyimpan *item-item* yang didapat pemain terdapat di bagian atas layar.



Gambar 3.19. Rancangan tampilan permainan

3.3.4 Tampilan Pesan Menang dan Kalah

Tampilan pesan menang ialah pesan yang muncul ketika pemain telah berhasil mengalahkan satu musuh besar di *dungeon* kedua. Tampilan pesan kalah ialah pesan yang muncul ketika pemain *game over* atau kehilangan seluruh *health point*. Tampilan pesan menang dan kalah tidak ada tombol karena secara otomatis akan memulai level selanjutnya setelah beberapa saat. Rancangan

tampilan pesan menang dan kalah dapat dilihat pada Gambar 3.20 dan Gambar 3.21.

Gambar 3.20. Rancangan tampilan pesan menang

Gambar 3. 21 Rancangan tampilan saat *gameover*

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Pada bab ini dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi merupakan lingkungan dimana *game* akan dibangun. Lingkungan implementasi dibagi menjadi dua, yaitu lingkungan implementasi berupa perangkat keras dan lingkungan implementasi berupa perangkat lunak.

4.1.1 Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan aplikasi ini adalah komputer dengan spesifikasi sebagai berikut:

- Tipe : Asus A456U.
- Prosesor : Intel® Core(TM) i7-7500U CPU @ 2.70GHz (8 CPUs) ~3.5 GHz.
- Memori : 8192 MB RAM.

Sedangkan untuk melakukan implementasi, digunakan perangkat desktop dengan spesifikasi yang sama.

4.1.2 Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan aplikasi ini adalah sebagai berikut:

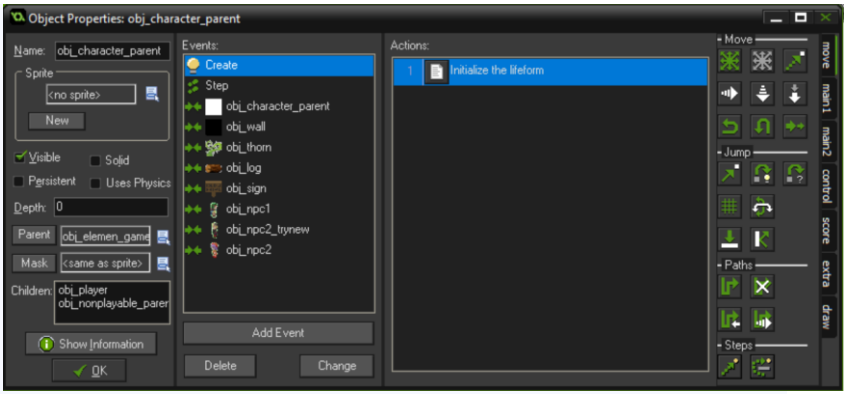
- Sistem Operasi : Windows 10 Home 64-bit.
- *Game Engine* : GameMaker Studio: Professional 1.4.

Sedangkan untuk melakukan implementasi, digunakan perangkat desktop dengan spesifikasi perangkat lunak sebagai berikut:

- Sistem Operasi : Windows 10 Home 64-bit.

4.2 Implementasi Desain Instance ‘obj_character_parent’

Subbab ini membahas mengenai implementasi rancangan labirin seperti yang sudah dijelaskan pada Subbab 3.3.1. Kode implementasi instance ‘obj_character_parent’ dapat dilihat pada Gambar 4. 1.



Gambar 4. 1 Implementasi instance ‘obj_character_parent’

Instance ‘obj_character_parent’ memiliki beberapa Event di dalamnya. Selain collision event dengan banyak benda, terdapat ‘Create’ event dan ‘Step’ event.

Dalam ‘Create’ event, instance memiliki turunan dari sifat instance *parent* ‘elemen_game’ dan menginisialisasikan 1 sifat baru yaitu ‘phy_fixed_rotation’ yang berfungsi untuk menyeimbangkan karakter saat bergerak sehingga saat berjalan tidak berputar. ‘Create’ event pada elemen ‘obj_character_parent’ dapat dilihat pada Kode Sumber 4.1.

```
1. event_inherited();
2. phy_fixed_rotation = true;
```

Kode sumber 4.1 ‘Create’ event pada ‘obj_character_parent’

Sedangkan ‘Step’ event berisi 2 sifat : insiasi variable ‘depth’ dan pengecekan nilai variabel ‘hp’.
‘Step’ event pada elemen ‘obj_character_parent’ ini dapat dilihat pada Kode Sumber 4.2.

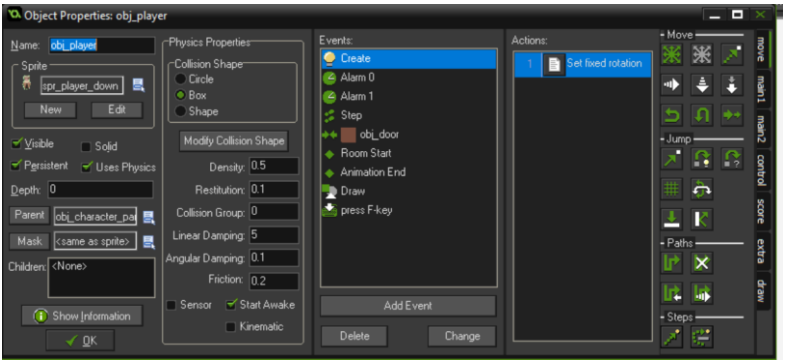
```
1. event_inherited();
2. depth = -y;
3. if (hp <= 0) {
4.  instance_destroy();
5. }
```

Kode sumber 4.2 ‘Step’ event pada ‘obj_character_parent’

Pada baris ke-3 dapat dilihat bahwa saat variable ‘hp’bernilai kurang dari sama dengan 0, instance akan hilang.

4.2.1 Implementasi Desain Instance ‘obj_player’

Subbab ini membahas mengenai implementasi desain instance ‘obj_player’ yang sudah dijelaskan secara garis besar dalam bentuk diagram pada subbab 3.3.1.1. Implementasi instance ini dapat dilihat pada Gambar 4.2.



Gambar 4.2. Implementasi Instance ‘obj_player’

‘Create’ event dapat dilihat pada Kode Sumber 4.3.

```
1. event_inherited();
2. spd = 2;
3. hspd = 0;
4. vspd = 0;
5. xaxis = 0;
6. yaxis = 0;
7. len = 0;
8. dir = 0;
9. attacked = false;
10. image_speed = 0; //Don’t play animations
11. scr_get_input();
12. state = scr_move_state;
13. face = DOWN;
14. isTalking = false; //Player is chatting or not
```

Kode Sumber 4.3. Implementasi ‘Create’ event pada ‘obj_player’

Baris 1 hingga baris 10 inisialisasi variable yang dibutuhkan untuk mendapatkan arah saat ini dari *sprite* pemain. Pada baris ke-11, *script* bernama *‘scr_get_input’* dipanggil. Pada *script* itu, terdapat *command* untuk navigasi pemain.

Pada ‘Step’ event yang dapat dilihat pada Kode Sumber 4.4, terdapat 2 hal yang dilakukan pada *step event*, yakni :

1. Menggerakkan pemain dan mengecek apakah HP > 0.

```
event_inherited();
script_execute(state);

// Check for death
if(obj_player_stats.hp <= 0)
{
    instance_destroy();
    room_goto(room_three);
}
```

Kode Sumber 4.4 ‘Step’ event pada ‘obj_player’

2. Menggunakan *item.*

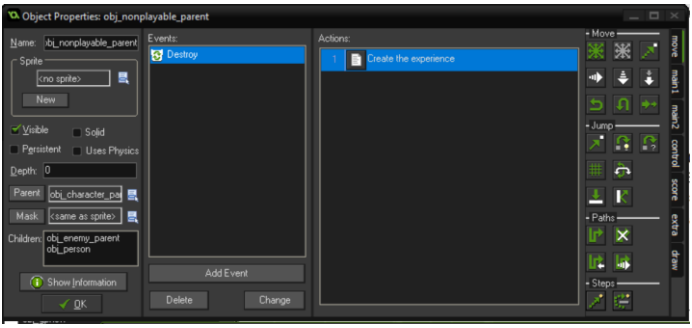
```
1. if(keyboard_check_pressed(ord('H')))
2. {
3.     with (obj_item_stats)
4.     {
5.         if(item_potion > 0)
6.         {
7.             with (obj_player_stats){
8.                 if(hp < maxhp)
9.                     {
10.                        hp += 3;
11.                    }
12.             }
13.             item_potion -= 1;
14.         }
15.     }
16. }
17.
18. if(keyboard_check_pressed(ord('J'))){
19.     with (obj_player_stats) {
20.         defense += 2;
21.     }
22.     with (obj_item_stats) {
23.          item_armor -= 1;
24.     }
25. }
```

Kode Sumber 4.5 'Step' event kedua pada 'obj_player'

Baris pertama hingga baris ke-16 adalah proses yang terjadi saat pemain menekan tombol 'H' pada keyboard, yakni pemakaian *item potion*. Sedangkan baris ke-18 hingga baris terakhir adalah pemakaian *item armor* dengan cara menekan tombol 'J' pada keyboard.

4.2.2 Implementasi Desain Instance ‘obj_nonplayable_parent’

Subbab ini membahas mengenai implementasi desain instance ‘obj_nonplayable_parent’ yang sudah dijelaskan secara garis besar dalam bentuk diagram pada subbab 3.3.1.2. Implementasi bisa dilihat pada Gambar 4.3.



Gambar 4.3. Implementasi ‘obj_nonplayable_parent’

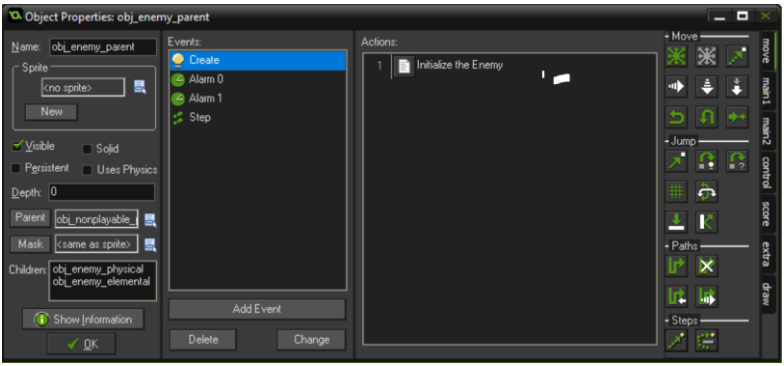
‘Destroy’ event dapat dilihat pada Kode Sumber 4.6.

```
/// Create the experience_point (EXP)
    instance_create(x, y, obj_expr);

/// Drop a health pack randomly
    if (scr_chance(.5))
    {
      instance_create(x+random_range(-4, 4),
      y+random_range(-4, 4), obj_health);
    }

/// Drop a money pack randomly
    if (scr_chance(.5))
   {
     instance_create(x+random_range(-4, 4),
     y+random_range(-4, 4), obj_money);
   }
```

Kode Sumber 4.6 ‘Destroy’ event pada ‘obj_nonplayable_parent’

4.2.2.1 Implementasi Desain Instance ‘obj_enemy_parent’

Subbab ini membahas mengenai implementasi desain pada instance ‘obj_enemy_parent’ seperti yang diilustrasikan pada subbab 3.3.1.2.1. Implementasi dapat dilihat pada Gambar 4.4.



Gambar 4.4 Implementasi desain instance ‘obj_enemy_parent’

Selain 2 ‘Alarm’ event yang masing-masing menandakan bahwa musuh bisa berjalan kesana kemari *(wander_state)* dan bisa hanya diam sesaat *(stall_state)*, terdapat 2 event lain yang dimiliki oleh instance ‘obj_enemy_parent’, yakni ‘Create’ event yang dapat dilihat pada Kode Sumber 4.7 dan ‘Step’ event yang dapat dilihat pada Kode Sumber 4.8.

```
/// Initialize the Enemy
1. event_inherited();
2.
3. image_speed = .1;
4. hp = 3;
5. spd = 1;       // kecepatan bergerak musuh
6. state = scr_enemy_idle_state;
7. alarm[0] = room_speed*irandom_range(2, 5);
8. sight = 64;
9. targetx = 0;  // untuk mendapatkan posisi x pemain
10.targety = 0;  // untuk mendapatkan posisi y pemain
```

Kode sumber 4.7 ‘Create’ event pada ‘obj_enemy_parent’

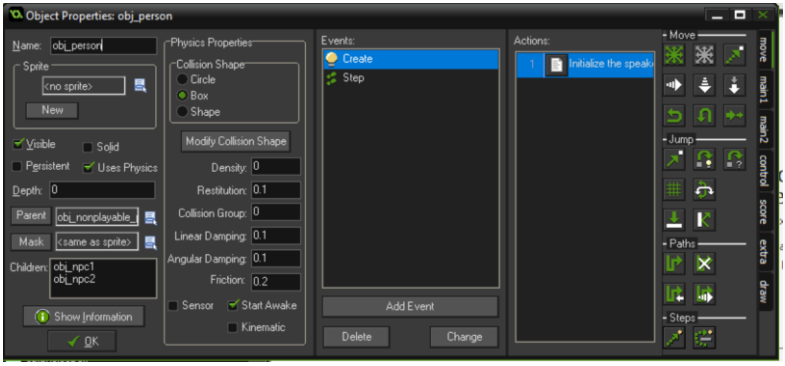
Dalam ‘Create’ event pada ‘obj_enemy_parent’, selain inisialisasi beberapa variabel (baris ke-3 hingga 5), inisialisasi *script* ‘scr_enemy_idle_state’ yang akan dipanggil dilakukan. Pada ‘Step’ event, selain mendapat turunan dari elemen orangtua-nya, sistem akan menjalankan *script* yang sudah diinisialisasi pada ‘Create’ event. ‘Step’ event pada ‘obj_enemy_parent’ dapat dilihat pada Kode Sumber 4.8.

```
/// Run the current state
event_inherited();

script_execute(state);
```

Kode Sumber 4.8. ‘Step’ event

4.2.2.2 Implementasi Desain Instance ‘obj_person’

Subbab ini membahas mengenai implementasi desain pada instance ‘obj_person’ seperti yang diilustrasikan pada subbab 3.3.1.2.2. Implementasi dapat dilihat pada Gambar 4.5.



Gambar 4.5. Implementasi Instance ‘obj_person’

‘Create’ event pada elemen ‘obj_person’ dapat dilihat pada Kode

Sumber 4.9. Selain mendapatkan fungsi yang diturunkan oleh orangtua-nya, inisialisasi beberapa variabel pada ‘obj_person’ yang digunakan untuk memunculkan balon dialog dan teks dialog dilakukan di event ini.

```
/// Initialize the NPCs
event_inherited();
dialog = noone;
dialog_page = 0;
xoffset = -32;
yoffset = -40;
```

Kode Sumber 4.9. ‘Create’ event pada ‘obj_person’

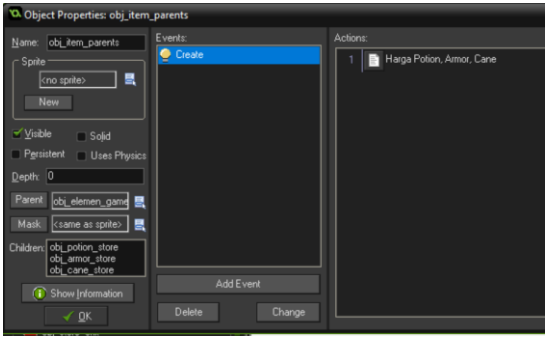
Sedangkan ‘Step’ event elemen ini dapat dilihat pada Kode Sumber 4.10.

```
depth = -y; //selalu bersifat negatif
```

Kode Sumber 4.10. ‘Step’ event pada ‘obj_person’

4.3 Implementasi Desain Instance ‘obj_item_parents’

Subbab ini membahas mengenai implementasi desain pada instance ‘obj_item_parents’ seperti yang diilustrasikan pada subbab 3.3.2. Implementasi dapat dilihat pada Gambar 4.6.



Gambar 4.6. Implementasi instance ‘obj_item_parents’

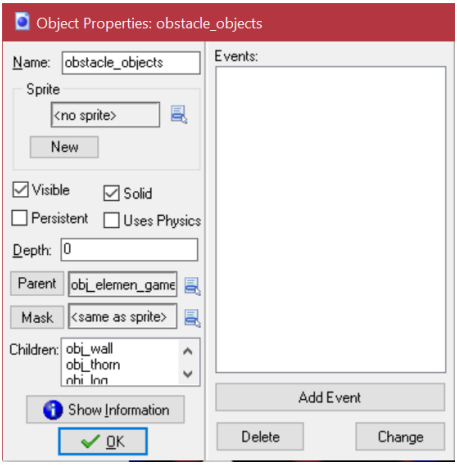
Di dalam ‘Create’ event yang dapat dilihat pada Kode Sumber 4.11, instance ini menginisialisasi harga *item potion*, *armor*, dan senjata *cane*.

```
/// Initialize the prices
price_potion = 200;
price_armor = 350;
price_cane = 500;
```

Kode Sumber 4.11. ‘Create’ event pada ‘obj_item_parents’

4.4 Implementasi Desain Instance ‘obstacle_objects’

Subbab ini membahas mengenai implementasi instance ‘obstacle_objects’ seperti yang sudah diilustrasikan pada Subbab 3.3.3.

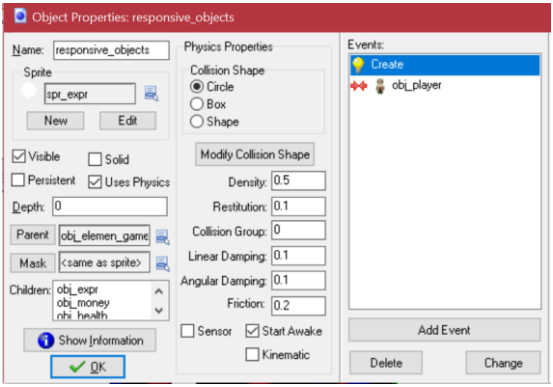


Gambar 4.7. Implementasi Desain Instance ‘obstacle_objects’

Pada Gambar 4.7, dapat dilihat bahwa tidak ada *event* apapun yang dimiliki oleh instance ‘obstacle_objects’.

4.5 Implementasi Desain Instance ‘responsive_objects’

Subbab ini membahas mengenai implementasi instance ‘responsive_objects’ seperti yang sudah diilustrasikan pada Subbab 3.3.4.



Gambar 4.8. Implementasi Desain Instance ‘responsive_objects’

Di dalam *Create* event yang dapat dilihat pada Kode Sumber 4.12, variabel-variabel yang dimiliki oleh instance-instance *child* diinisialisasi didalamnya.

```
/// Initialize variables of the children's
expr = false;
healthbar = false;
money = false;
potion = false;
cane = false;
armor = false;

// Initialize variables for door
door = true;
new_x = 0;
new_y = 0;
new_room = noone;
```

Kode Sumber 4.12. ‘Create’ event pada ‘responsive_objects’

Sedangkan *Collision* event dengan instance ‘obj_player’ berisi fungsionalitas pemain untuk mengambil *item*. *Collision* event ini dapat dilihat pada Kode Sumber 4.13.

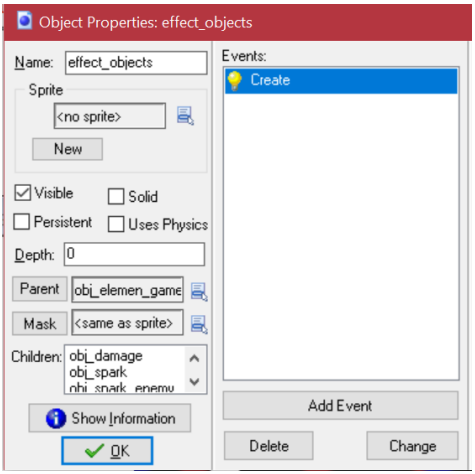
```
/// Collect items
with(self) {
    if(expr == true) {          //ambil EXP
        with (obj_player_stats)
        {
            expr += 1;
            if(expr >= maxexpr)
            {
                level += 1;
                expr = expr - maxexpr;
                maxexpr *= 2;
                hp += 4;
                maxhp +=4;
                stamina += 2;
                maxstamina += 2;
                attack += 1;
            }
        }
    }
    if(money == true) {         //ambil koin
        with (obj_player_stats)
        {
            money += 250;
        }
    }
    if(healthbar == true) {     //ambil healthbar
        with (obj_player_stats)
        {
            hp = min(hp+5, maxhp);
        }
    }
```

```
    if(cane == true) {          //ambil cane
        // Giving values to global.box2
        global.box2 = 2;
        global.cane = true;
        if(global.box2 = 2) {
            // Number of item changes
            with (obj_item_stats)
            {
                item_weapon += 1;
            }
        }
    }
    if(armor == true) {         //ambil armor
        global.box3 = 3;
        global.armor = true;

        if(global.box3 = 3) {
            // Number of item changes
            with (obj_item_stats)
            {
                item_armor += 1;
            }
        }
    }

    instance_destroy();
}
```

Kode Sumber 4.13. 'Collision' event pada instance 'responsive_objects'

4.6 Implementasi Instance 'effect_objects'

Subbab ini membahas mengenai implementasi instance 'effect_objects' seperti yang sudah diilustrasikan pada Subbab 3.3.5. Instance ini dapat dilihat pada Gambar 4.9.



Gambar 4.9. Implementasi Instance 'effect_objects'

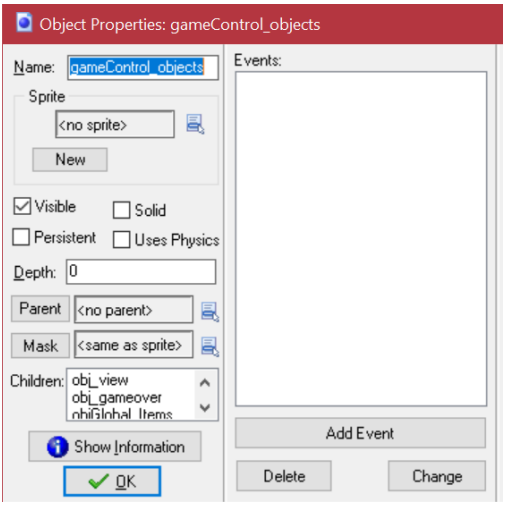
Instance ini memiliki *Create* event yang berisi inisialisasi nilai *damage* dan *knockback* dari instance-instance *child* yang dimiliki. 'Create' event ini dapat dilihat pada Kode Sumber 4.14

```
/// Initialize Damage and Knockback
damage = 1;
robot_damage = 5;
knockback = 1;
creator = noone;
magic_damage = 3;
magic_knockback = 10;
magic_creator = noone;
```

Kode Sumber 4.14. 'Create' event pada instance 'effect_objects'

4.7 Implementasi Instance 'gameControl_objects'

Subbab ini membahas mengenai implementasi instance 'gameControl_objects' seperti yang sudah diilustrasikan pada Subbab 3.3.6. Instance ini menjadi *parent* dari beberapa instances pengontrol permainan.



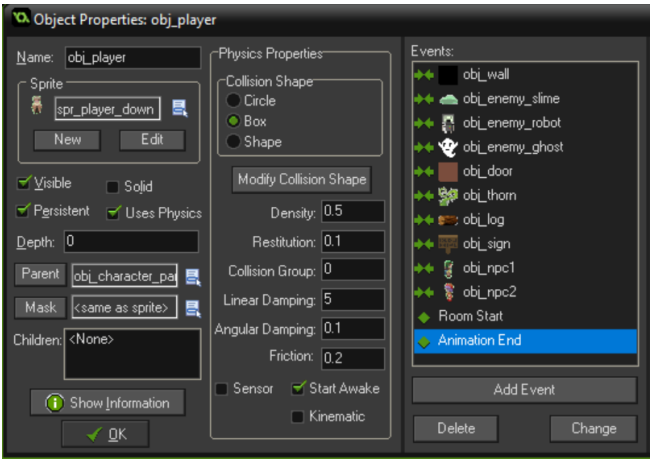
Gambar 4.10. Implementasi Desain Instance 'gameControl_objects'

4.8 Implementasi Perbedaan Desain Konvensional dengan Object-Oriented

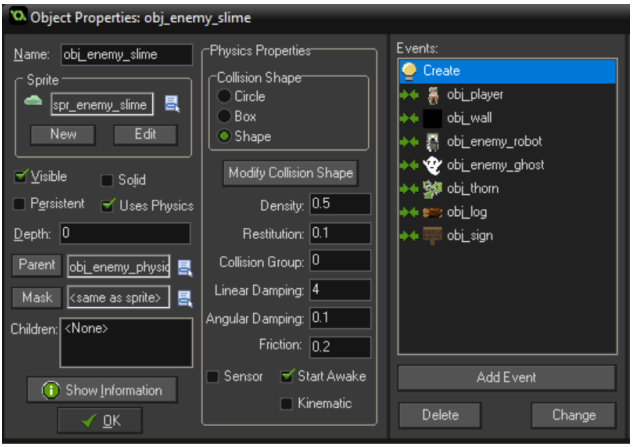
Penyusunan elemen secara konvensional memiliki perbedaan dengan perancangan elemen menggunakan prinsip *Object-Oriented Design*. Secara umum, desain elemen menggunakan secara konvensional lebih rawan akan kesalahan karena sifatnya yang tidak terkategorikan dan fungsi yang diulang-ulang. Sebaliknya, desain secara *Object-Oriented* lebih efisien karena terkategorikan secara rapi berdasarkan sifat-sifatnya. Perbedaan lebih jelas dapat dilihat pada Tabel 4.1.

Tabel 4.1. Perbedaan Desain Konvensional dengan Object-Oriented

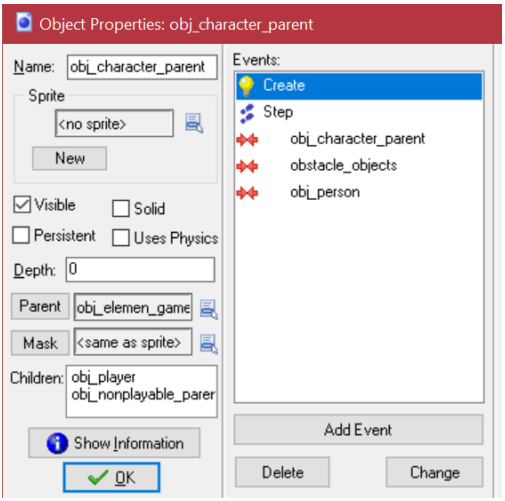
No	Nama Elemen	Konvensional	Object-Oriented
1.	obj_character_parent	- Berantakan - Tidak efisien - Repetitif	- Terkategorikan - Fleksibel terhadap perubahan - Tidak repetitif
2.	obj_item_ parents	- Berantakan - Tidak efisien - Repetitif	- Terkategorikan - Fleksibel terhadap perubahan - Tidak repetitif
3.	obstacle_objects	- Berantakan - Tidak efisien - Repetitif	-Terkategorikan - Mudah dilakukan pengubahan - Tidak repetitif
4.	responsive_object	- Berantakan - Tidak efisien - Repetitif	- Terkategorikan - Fleksibel terhadap perubahan - Tidak repetitif
5.	effect_objects	- Berantakan - Tidak efisien - Repetitif	- Terkategorikan - Fleksibel terhadap perubahan - Tidak repetitif
6.	gameControl_objects	- Berantakan - Tidak efisien - Repetitif	- Terkategorikan - Fleksibel terhadap perubahan - Tidak repetitive



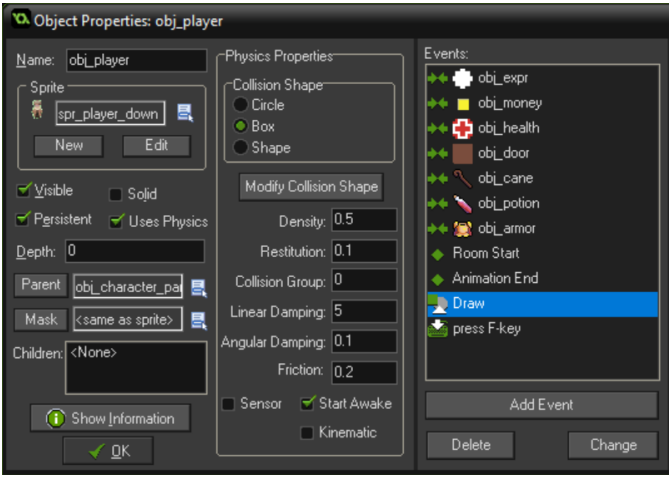
Gambar 4.11. Bila 'obj_character_parent' dan 'obstacle_objects' didesain secara konvensional pada 'obj_player'



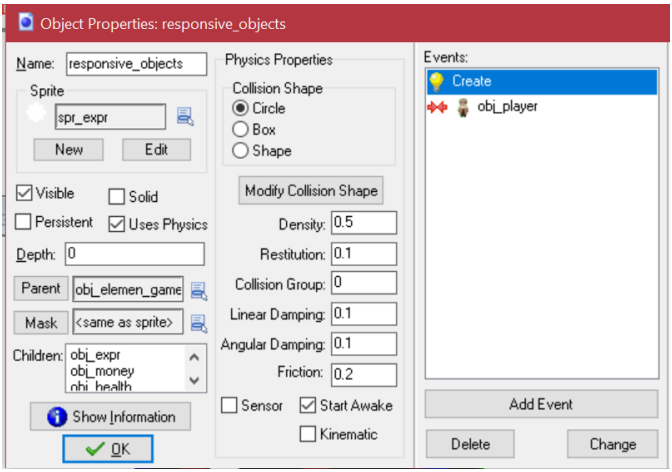
Gambar 4.12. Bila 'obj_character_parent' dan 'obstacle_objects' didesain secara konvensional pada 'obj_enemy_slime'



Gambar 4.13. 'obj_character_parent', 'obstacle_objects', dan 'obj_person' didesain secara Object-Oriented



Gambar 4.14. Bila 'responsive_objects' didesain secara konvensional



Gambar 4.15. 'responsive_objects' didesain secara Object-Oriented

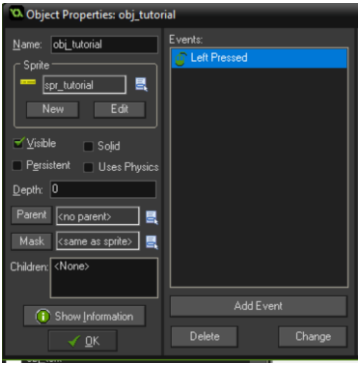
4.9 Implementasi Permainan

Subbab ini membahas mengenai implementasi pada permainan. Implementasi yang dimaksud dapat berupa implementasi aturan main, implementasi fungsi, dan implementasi rancangan tampilan awal. Implementasi permainan akan dibahas berdasarkan antarmuka-antarmuka yang ada pada permainan.

4.9.1 Implementasi Tampilan Awal

Hasil implementasi tampilan awal (menu) dapat dilihat pada Gambar 4.11. Object button 'Start', 'Tutorial', dan 'Exit' diletakkan pada room_menu.

Gambar 4.16. Implementasi desain tampilan menu



Gambar 4. 17 Event 'Left Pressed' pada tombol 'Tutorial'

4.9.2 Implementasi Tampilan Tutorial

Hasil implementasi desain tampilan 'Tutorial' dapat dilihat pada Gambar 4.18. Saat tombol 'Tutorial' (obj_tutorial) diklik, pemain akan diarahkan ke room lain yang berisi gambar dengan arahan-arahan sebagai *guide* (room_tutorial2). Event yang mentrigger hal tersebut bisa dilihat pada Gambar 4.17. Pada

room_tutorial1, terdapat 2 tombol bertuliskan ‘Back’ dan ‘Next’. Object button ‘Next’ dan ‘Back’ diletakkan pada room_tutorial2 untuk navigasi halaman tutorial.

Gambar 4.18 Tampilan ‘Tutorial’

4.9.3 Implementasi Tampilan Permainan

Hasil implementasi tampilan permainan dapat dilihat pada Gambar 4.19. Status player *(stats)* terletak di sebelah kiri atas, sedangkan kotak inventory terletak di bagain atas tengah pada layar. *Item* yang tergeletak di tanah dapat diambil.

Gambar 4. 19 Tampilan permainan

Kode sumber cara pemain bergerak dapat dilihat pada Kode Sumber 4.15.

```
// Bila pemain tidak sedang mengobrol
if (!isTalking) {

    if(attack_key)        // tombol 'X' pada keyboard
    {
        image_index = 0;
        state = scr_attack_state;
    }

    // Mendapatkan arah karakter pemain saat ini
    dir = point_direction(0, 0, xaxis, yaxis);

    // Mendapatkan variabel 'len'
    if (xaxis == 0 and yaxis == 0)
    {
        len = 0;
    }

    // Mendapatkan hspd dan vspd
    hspd = lengthdir_x(len, dir); //jarak x pemain
    vspd = lengthdir_y(len, dir); //jarak y pemain

    // Bergerak sejauh hspd dan vspd
    phy_position_x += hspd;
    phy_position_y += vspd;

    // Mengontrol sprite
    image_speed = .2;
    if(len == 0) image_index = 0;

    // Mendapatkan arah pemain sesuai dengan navigasi
    switch (face)
    {
        case RIGHT:
            sprite_index = spr_player_right;
            break;
        case UP:
            sprite_index = spr_player_up;
            break;
```

```
        case LEFT:
            sprite_index = spr_player_left;
            break;

        case DOWN:
            sprite_index = spr_player_down;
            break;
    }
}

else
    image_speed = 0;          // pemain tidak bergerak
```

Kode Sumber 4. 15 Fungsi untuk menggerakkan karakter

Sedangkan kode sumber cara pemain bertarung dapat dilihat pada Kode Sumber 4.16.

```
//Player fights

image_speed = .4; //kecepatan animasi dimainkan

switch(sprite_index)
{
          // sprite_index saat pemain hadap bawah
    case spr_player_down:
    sprite_index = spr_player_attack_down;
    break;

          // sprite_index saat pemain hadap atas
    case spr_player_up:
    sprite_index = spr_player_attack_up;
    break;

          // sprite_index saat pemain hadap kanan
    case spr_player_right:
    sprite_index = spr_player_attack_right;
    break;
          // sprite_index saat pemain hadap kiri
    case spr_player_left:
    sprite_index = spr_player_attack_left;
    break; }
```

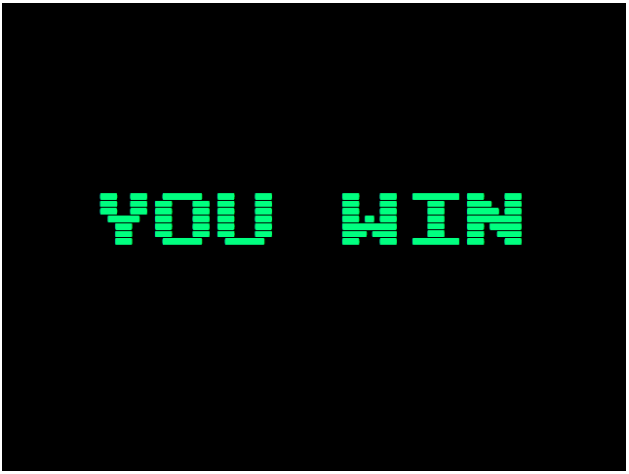
```
// saat pemain tidak sedang diserang
if (image_index > 2 && attacked == false)
{
    var xx = 0;
    var yy = 0;
    switch(sprite_index)
    {
        case spr_player_attack_down:
            xx = x;
            yy = y+12;
            audio_play_sound(sound_attack, 10,
            false);
            break;

        case spr_player_attack_up:
            xx = x;
            yy = y-10;
            audio_play_sound(sound_attack, 10,
            false);
            break;

        case spr_player_attack_right:
            xx = x+10;
            yy = y+2;
            audio_play_sound(sound_attack, 10,
            false);
            break;

        case spr_player_attack_left:
            xx = x-10;
            yy = y+2;
            audio_play_sound(sound_attack, 10,
            false);
            break;
    }

    // membuat instance obj_damage
    var damage = instance_create(xx, yy,
                 obj_damage);
    damage.sprite_index = 3;
    damage.creator = id;
    damage.damage = obj_player_stats.attack;
    attacked = true;}
```

Kode Sumber 4. 16 Fungsi bertarung

4.9.4 Implementasi Tampilan Pesan Menang Dan Kalah

Hasil implementasi tampilan pesan kalah dapat dilihat pada Gambar 4.20, sedangkan implementasi tampilan pesan menang dapat dilihat pada Gambar 4.21.

Gambar 4.20. Tampilan saat *gameover*



Gambar 4.21. Tampilan saat menang

(Halaman ini sengaja dikosongkan)

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini dijelaskan tentang uji coba dan evaluasi dari implementasi yang telah dilakukan.

5.1 Lingkungan Pengujian

Lingkungan uji coba yang digunakan adalah sebuah *smartphone* dengan spesifikasi sebagai berikut:

1. Perangkat Keras
   Tipe : ASUS A456U.
   Prosesor : Intel® Core(TM) i7-7500U CPU @ 2.70GHz (8 CPUs) ~3.5 GHz.
   Memori : 8192 MB RAM.
2. Perangkat Lunak
   Sistem Operasi : Windows 10-Home 64 Bit.

Pada skenario pengujian dijelaskan tentang skenario pengujian yang dilakukan. Pengujian yang akan dilakukan hanya 1 jenis, yaitu pengujian fungsionalitas. Pengujian ini dilakukan untuk mengetahui apakah fungsionalitas *game* telah berjalan sebagai mana mestinya setelah elemen-elemen game didesain dan diimplementasi menggunakan prinsip Object-Oriented Design.

5.2 Pengujian Fungsionalitas

Pengujian fungsionalitas dilakukan dengan menyiapkan beberapa skenario pengujian sebagai tolok ukur keberhasilan pengujian. Pengujian pertama ialah untuk mengetahui apakah variable-variabel milik *parent* dapat berhasil diturunkan ke anak-anaknya, pengujian kedua adalah pengujian aturan permainan untuk mengetahui apakah pemain dapat menjalankan fitur-fitur yang ada pada permainan dengan baik.

5.2.1 Pengujian Penurunan Sifat

Pengujian penurunan sifat merupakan pengujian terhadap sistem permainan untuk melihat apakah sifat-sifat di dalam elemen permainan berhasil diturunkan ke instance-instance *child*. Skenario pengujian ada dua belas, yaitu saat penurunan sifat instance ‘elemen_game’, penurunan sifat instance ‘elemen_game’. Hasil pengujian penurunan sifat dari instance yang menjadi *parent* dapat dilihat pada Tabel 5.1 Hasil Pengujian Penurunan Sifat.

Tabel 5. 1 Hasil pengujian penurunan sifat

ID	PF-001
Nama	Pengujian Penurunan Sifat dari Instance *Parent*
Tujuan uji coba	Sifat-sifat yang dimiliki instance *parent* berhasil diturunkan ke instance-instance *child*
Kondisi awal	Sifat-sifat belum diturunkan
Skenario 1	***Penurunan sifat instance ‘elemen_game’***
Masukan	-
Keluaran yang diharapkan	‘Create’ event dan ‘Step’ event pada instance ‘elemen_game’ berhasil diturunkan ke anak-anaknya (*child)*
Hasil uji coba	Berhasil
Kondisi Akhir	‘Create’ event dan ‘Step’ event pada instance ‘elemen_game’ berhasil diturunkan ke anak-anaknya (*child)*
Skenario 2	***Penurunan sifat instance ‘obj_character_parent’***
Masukan	-

ID	PF-001
Keluaran yang diharapkan	Sifat-sifat yang dimiliki instance ‘obj_character_parent’ berhasil diturunkan ke anak-anaknya *(child)*
Hasil uji coba	Berhasil
Kondisi Akhir	Sifat-sifat instance ‘obj_character_parent’ berhasil diturunkan ke anak-anaknya *(child)*
Skenario 3	***Penurunan sifat instance ‘obj_nonplayable_parent’***
Masukan	-
Keluaran yang diharapkan	Sifat-sifat instance ‘obj_nonplayable_parent’ berhasil diturunkan ke anak-anaknya *(child)*
Hasil uji coba	Berhasil
Kondisi Akhir	Sifat-sifat instance ‘obj_nonplayable_parent’ berhasil diturunkan ke anak-anaknya *(child)*
Skenario 4	***Penurunan sifat instance ‘obj_enemy_parent’***
Masukan	-
Keluaran yang diharapkan	Sifat-sifat instance ‘obj_enemy_parent’ berhasil diturunkan ke anak-anaknya *(child)*
Hasil uji coba	Berhasil
Kondisi Akhir	Sifat-sifat instance ‘obj_enemy_parent’ berhasil diturunkan ke anak-anaknya *(child)*
Skenario 5	***Penurunan sifat instance ‘obj_enemy_physical’***
Masukan	-
Keluaran yang diharapkan	Sifat-sifat instance ‘obj_enemy_physical’ berhasil diturunkan ke anak-anaknya *(child)*
Hasil uji coba	Berhasil
Kondisi Akhir	Sifat-sifat instance ‘obj_enemy_physical’

ID	PF-001
	berhasil diturunkan ke anak-anaknya *(child)*
Skenario 6	***Penurunan sifat instance 'obj_enemy_elemental'***
Masukan	-
Keluaran yang diharapkan	Sifat-sifat instance 'obj_enemy_elemental' berhasil diturunkan ke anak-anaknya *(child)*
Hasil uji coba	Berhasil
Kondisi Akhir	Sifat-sifat instance 'obj_enemy_elemental' berhasil diturunkan ke anak-anaknya *(child)*
Skenario 7	***Penurunan sifat instance 'obj_person'***
Masukan	-
Keluaran yang diharapkan	Sifat-sifat yang dimiliki instance 'obj_person' berhasil diturunkan ke anak-anaknya *(child)*
Hasil uji coba	Berhasil
Kondisi Akhir	Sifat-sifat instance 'obj_person' berhasil diturunkan ke anak-anaknya *(child)*
Skenario 8	***Penurunan sifat instance 'obj_item_parents'***
Masukan	-
Keluaran yang diharapkan	Sifat-sifat yang dimiliki instance 'obj_item_parents' berhasil diturunkan ke anak-anaknya *(child)*
Hasil uji coba	Berhasil
Kondisi Akhir	Sifat-sifat instance 'obj_item_parents' berhasil diturunkan ke anak-anaknya *(child)*
Skenario 9	***Penurunan sifat instance 'obstacle_objects'***
Masukan	-
Keluaran yang	Sifat-sifat yang dimiliki instance

ID	PF-001
diharapkan	‘obstacle_objects’ berhasil diturunkan ke anak-anaknya *(child)*
Hasil uji coba	Berhasil
Kondisi Akhir	Sifat-sifat instance ‘obstacle_objects’ berhasil diturunkan ke anak-anaknya *(child)*
Skenario 10	***Penurunan sifat instance ‘responsive_objects’***
Masukan	-
Keluaran yang diharapkan	Sifat-sifat yang dimiliki instance ‘responsive_objects’ berhasil diturunkan ke anak-anaknya *(child)*
Hasil uji coba	Berhasil
Kondisi Akhir	Sifat-sifat instance ‘responsive_objects’ berhasil diturunkan ke anak-anaknya *(child)*
Skenario 11	***Penurunan sifat instance ‘effect_objects’***
Masukan	-
Keluaran yang diharapkan	Sifat-sifat yang dimiliki instance ‘effect_objects’ berhasil diturunkan ke anak-anaknya *(child)*
Hasil uji coba	Berhasil
Kondisi Akhir	Sifat-sifat instance ‘effect_objects’ berhasil diturunkan ke anak-anaknya *(child)*
Skenario 12	***Penurunan sifat instance ‘gameControl_objects’***
Masukan	-
Keluaran yang diharapkan	Sifat-sifat yang dimiliki instance ‘gameControl_objects’ berhasil diturunkan ke anak-anaknya *(child)*
Hasil uji coba	Berhasil

ID	PF-001
Kondisi Akhir	Sifat-sifat instance ‘gameControl_objects’ berhasil diturunkan ke anak-anaknya *(child)*

5.2.2 Pengujian Aturan Permainan

Pengujian aturan main merupakan pengujian terhadap *game* untuk melihat apakah pemain dapat menemukan jalan untuk ke level selanjutnya. Skenario pengujian ada lima belas, yaitu ketika pemain bertemu musuh, pemain diserang musuh, pemain kehilangan seluruh *health point* (HP), pemain menyerang musuh, musuh kehilangan seluruh *health point* (HP), pemain mengambil *experience point* (EXP), *experience point* (EXP) mencapai maksimum, pemain mengambil *health-pack* yang dijatuhkan musuh secara random saat musuh kalah, pemain mengambil koin yang dijatuhkan musuh secara random saat musuh kalah, pemain mengambil *item* yang tergeletak, pemain menggunakan *item potion*, pemain menggunakan *item armor*, pemain menggunakan serangan elemental tanpa mengambil *item cane* terlebih dahulu, pemain menggunakan serangan elemental setelah mengambil *item cane*, pemain berbicara dengan *non-playable character* (NPC).

Hasil pengujian aturan main dapat dilihat pada Tabel 5.2 Hasil Pengujian Aturan Main. Tampilan ketika pemain menghindari musuh dapat dilihat pada Gambar 5.1, tampilan ketika pemain menyerang musuh dapat dilihat pada Gambar 5.2 dan Gambar 5.3, tampilan ketika pemain diserang musuh dapat dilihat pada Gambar 5.4 dan Gambar 5.5, tampilan ketika pemain mengambil item dapat dilihat pada Gambar 5.6, Gambar 5.7, dan Gambar 5.8, tampilan ketika pemain berbicara dengan NPC dapat dilihat pada Gambar 5.9 dan Gambar 5.10, tampilan ketika

pemain kalah dapat dilihat pada Gambar 5.11, dan tampilan ketika pemain menang dapat dilihat pada Gambar 5.12.

Tabel 5. 2 Hasil pengujian aturan permainan

ID	PF-002
Nama	Pengujian Aturan Main
Tujuan uji coba	Pengguna memainkan *game* sesuai aturan main
Kondisi awal	Pemain menekan tombol 'Start' dan sudah berada dalam *game*
Skenario 1	***Pemain bertemu musuh***
Masukan	Pemain menggerakkan karakter menghindari musuh
Keluaran yang diharapkan	1. Musuh mengejar bila pemain masih berada dalam jangkauan musuh 2. Pemain berhasil menghindari musuh
Hasil uji coba	Berhasil
Kondisi Akhir	Musuh tidak dapat mendeteksi pemain karena berada di luar radar
Skenario 2	***Pemain diserang oleh musuh***
Masukan	Pemain menggerakkan karakter menyentuh musuh
Keluaran yang diharapkan	*Health Point* (HP) pemain berkurang
Hasil uji coba	Berhasil
Kondisi akhir	*Health Point* (HP) pemain berkurang
Skenario 3	***Pemain kehilangan seluruh health points***
Masukan	Tidak ada
Keluaran yang diharapkan	Muncul tampilan saat gameover

ID	PF-002
Hasil uji coba	Berhasil
Kondisi akhir	Muncul tampilan saat gameover
Skenario 4	***Pemain menyerang musuh***
Masukan	Pemain menekan tombol keyboard ‘X’ (serangan biasa) atau ‘C’ saat musuh sudah dekat
Keluaran yang diharapkan	1. *Health Point* (HP) musuh berkurang 2. Musuh terlempar sedikit bila *Health Point* (HP) musuh belum habis
Hasil uji coba	Berhasil
Kondisi akhir	1. *Health Point* (HP) musuh berkurang 2. Musuh terlempar sedikit bila *Health Point* (HP) musuh belum habis
Skenario 5	***Musuh kehilangan seluruh Health Point (HP)***
Masukan	Tidak ada
Keluaran yang diharapkan	1. Musuh hilang 2. Muncul *experience point* (EXP) dan sewaktu-waktu *health-pack* dan uang secara random
Hasil uji coba	Berhasil
Kondisi akhir	Musuh hilang
Skenario 6	***Pemain mengambil experience point (EXP)***
Masukan	Pemain menggerakkan karakter menyentuh *experience point* (EXP)
Keluaran yang diharapkan	*Experience point* (EXP) pemain bertambah
Hasil uji coba	Berhasil
Kondisi akhir	*Experience point* (EXP) pemain bertambah

ID	PF-002
Skenario 7	***Experience point* (EXP) pemain mencapai maksimum**
Masukan	Tidak ada
Keluaran yang diharapkan	1. Level pemain naik 2. Nilai maksimum *Health Point* (HP) bertambah 5 3. *Health point* (HP) pemain pulih 4. *Attack* pemain bertambah 5. Defense pemain bertambah
Hasil uji coba	Berhasil
Kondisi akhir	1. Level pemain naik 2. Nilai maksimum *Health Point* (HP) bertambah 3. *Health point* (HP) pemain pulih 4. *Attack* pemain bertambah 5. Defense pemain bertambah
Skenario 8	***Pemain mengambil health-pack yang dijatuhkan musuh sewaktu-waktu***
Masukan	Pemain menggerakkan karakter menyentuh *health-pack*
Keluaran yang diharapkan	*Health Point* (HP) pemain bertambah
Hasil uji coba	Berhasil
Kondisi akhir	*Health Point* (HP) pemain bertambah
Skenario 9	***Pemain mengambil uang koin yang dijatuhkan musuh sewaktu-waktu***
Masukan	Pemain menggerakkan karakter menyentuh uang koin
Keluaran yang	Uang yang dimiliki pemain bertambah

ID	PF-002
diharapkan	
Hasil uji coba	Berhasil
Kondisi akhir	Uang yang dimiliki pemain bertambah
Skenario 10	***Pemain mengambil item yang tergeletak***
Masukan	Pemain menggerakkan pemain menyentuh *item* yang tergeletak
Keluaran yang diharapkan	1. Item tersimpan di kotak inventory 2. Jumlah item yang awalnya 0 bertambah 1
Hasil uji coba	Berhasil
Kondisi akhir	1. Item tersimpan di kotak inventory 2. Jumlah item yang awalnya 0 bertambah 1
Skenario 11	***Pemain menggunakan item potion***
Masukan	Pemain menekan tombol keyboard ‘H’
Keluaran yang diharapkan	1. *Health Point* (HP) pemain bertambah 2. Jumlah item potion berkurang 1
Hasil uji coba	Berhasil
Kondisi akhir	1. *Health Point* (HP) pemain bertambah 2. Jumlah item potion berkurang 1
Skenario 12	***Pemain menggunakan item armor***
Masukan	Pemain menekan tombol keyboard ‘J’
Keluaran yang diharapkan	1. *Defense* pemain bertambah 2. Jumlah item armor berkurang 1
Hasil uji coba	Berhasil
Kondisi akhir	1. *Defense* pemain bertambah 2. Jumlah item armor berkurang 1
Skenario 13	***Pemain menggunakan serangan magic tanpa mengambil item cane terlebih dahulu***
Masukan	Pemain menekan tombol keyboard ‘C’
Keluaran yang	Tidak terjadi apapun

ID	PF-002
diharapkan	
Hasil uji coba	Berhasil
Kondisi akhir	Pemain tidak dapat mengeluarkan serangan magic
Skenario 14	***Pemain menggunaan serangan magic dengan item cane sudah diambil***
Masukan	Pemain menekan tombol keyboard ‘C’
Keluaran yang diharapkan	Serangan magic berwarna biru keluar dari posisi pemain
Hasil uji coba	Berhasil
Kondisi akhir	Serangan magic berwarna biru keluar dari posisi pemain
Skenario 15	***Pemain berbicara dengan NPC***
Masukan	Pemain menekan tombol keyboard ‘X’ berhadapan dengan NPC yang dimaksud
Keluaran yang diharapkan	1. Muncul percakapan biasa bila NPC yang diajak bicara adalah orang biasa 2. Muncul proses transaksi bila NPC yang diajak bicara adalah pedagang
Hasil uji coba	Berhasil
Kondisi akhir	1. Muncul percakapan biasa bila NPC yang diajak bicara adalah orang biasa 2. Muncul proses transaksi bila NPC yang diajak bicara adalah pedagang

Gambar 5.1 Pemain menghindari musuh

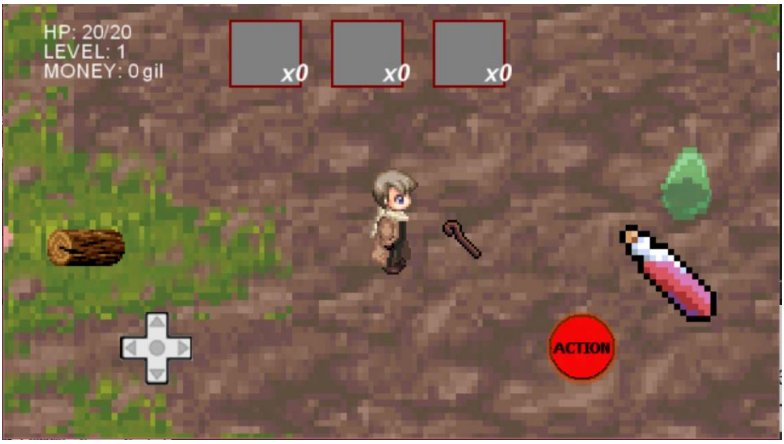
Gambar 5.2. Pemain menyerang secara fisik

Gambar 5.3. Pemain menyerang menggunakan serangan elemental

Gambar 5.4. Pemain diserang oleh musuh

Gambar 5.5. Pemain diserang musuh elemental

Gambar 5.6. Pemain telah mengambil *experience point*, *health-pack*, dan atau uang



Gambar 5.7. Pemain akan mengambil *item cane*

Gambar 5.8. Pemain telah mengambil *item cane*

Gambar 5.9. Pemain berbicara pada NPC Orang Biasa

Gambar 5.10. Pemain berbicara dengan NPC pedagang

Gambar 5.11. Pemain kalah

Gambar 5.12. Pemain menang

(Halaman ini sengaja dikosongkan)

5.3 Evaluasi Pengujian Fungsionalitas

Rangkuman hasil pengujian fungsionalitas dapat dilihat pada Tabel 5.3. Berdasarkan data pada tabel tersebut, dapat disimpulkan bahwa semua skenario pengujian berhasil dijalankan. Sehingga dapat disimpulkan bahwa fungsionalitas dari aplikasi telah bekerja sesuai dengan yang diharapkan.

Tabel 5. 3 Hasil pengujian fungsionalitas

Kode Pengujian	Nama	Hasil
PF001-1	Pengujian Penurunan Sifat (Skenario 1)	Berhasil
PF001-2	Pengujian Penurunan Sifat (Skenario 2)	Berhasil
PF001-3	Pengujian Penurunan Sifat (Skenario 3)	Berhasil
PF001-4	Pengujian Penurunan Sifat (Skenario 4)	Berhasil
PF001-5	Pengujian Penurunan Sifat (Skenario 5)	Berhasil
PF001-6	Pengujian Penurunan Sifat (Skenario 6)	Berhasil
PF001-7	Pengujian Penurunan Sifat (Skenario 7)	Berhasil
PF001-8	Pengujian Penurunan Sifat (Skenario 8)	Berhasil
PF001-9	Pengujian Penurunan Sifat (Skenario 9)	Berhasil
PF001-10	Pengujian Penurunan Sifat (Skenario 10)	Berhasil
PF001-11	Pengujian Penurunan Sifat (Skenario 11)	Berhasil
PF001-12	Pengujian Penurunan Sifat (Skenario 12)	Berhasil
PF002-1	Pengujian Aturan Main (Skenario 1)	Berhasil
PF002-2	Pengujian Aturan Main (Skenario 2)	Berhasil
PF002-3	Pengujian Aturan Main (Skenario 3)	Berhasil
PF002-4	Pengujian Aturan Main (Skenario 4)	Berhasil
PF002-5	Pengujian Aturan Main (Skenario 5)	Berhasil
PF002-6	Pengujian Aturan Main (Skenario 6)	Berhasil
PF002-7	Pengujian Aturan Main (Skenario 7)	Berhasil
PF002-8	Pengujian Aturan Main (Skenario 8)	Berhasil

Kode Pengujian	Nama	Hasil
PF002-9	Pengujian Aturan Main (Skenario 9)	Berhasil
PF002-10	Pengujian Aturan Main (Skenario 10)	Berhasil
PF002-11	Pengujian Aturan Main (Skenario 11)	Berhasil
PF002-12	Pengujian Aturan Main (Skenario 12)	Berhasil
PF002-13	Pengujian Aturan Main (Skenario 13)	Berhasil
PF002-14	Pengujian Aturan Main (Skenario 14)	Berhasil
PF002-15	Pengujian Aturan Main (Skenario 15)	Berhasil

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan yang dapat diambil dari hasil pengujian yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga diberi saran untuk pengembangan aplikasi kedepannya.

6.1 Kesimpulan

Dari hasil pengamatan selama proses pengerjaan mulai dari proses perancangan, implementasi, dan pengujian yang telah dilakukan terhadap aplikasi, diambil kesimpulan sebagai berikut :

1. Elemen-elemen permainan yang didesain menggunakan prinsip Object-Oriented Design lebih efisien dan fleksibel terhadap perubahan karena dikelompokkan berdasarkan sifat-sifatnya dan tidak repetitif, dapat dibuktikan pada Tabel 4.1.
2. Prinsip Object-Oriented Design, yaitu aspek *inheritance* antar object-object, telah berhasil diimplementasikan pada elemen-elemen game di dalam permainan ‘The Bloodline’ dan dapat dibuktikan pada Tabel 5.1.
3. Aturan permainan dan skenario telah berhasil diimplementasikan pada permainan ‘The Bloodline’ dan dapat dibuktikan pada Tabel 5.2.

6.2 Saran

Adapun saran yang diberikan untuk mengembangkan *game* ini adalah *game* ini masih memerlukan pengembangan lebih lanjut, terutama dalam hal pengaturan dungeon, tingkat kesusahan musuh, dan juga dari segi tampilan.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

[1] L. Bishop, D. Eberly, T. Whitted, M.Finch, M.Shantz., *Designing a PC game engine*, IEEE Computer Graphics and Application (1998) 46-53.

[2] S. Bjork, S.Lundgren, J. Holopainen., *Game design patterns*, Proceedings of Digital Games Research Conference 2003, Nov. 4-6, Utrecht, The Netherlands, 2003.

[3] Bjork, S and Holopainen, J., *Patterns in Game Design*, Charles River Media, 2004.

[4] Salen, K. and Zimmerman, E., *Rules of Play: Game Design Fundamentals*, MIT Press, 2003.

[5] Anpatzoglou, A. Chatzigeorgiou, A., *Evaluation of Object-Oriented Design Patterns in Game Development* (2006).

[6] Cover, Jennifer Grouling., *The Creation of Narrative in Tabletop Role-Playing Games*, McFarland & Company, 2010.

[7] Tychsen, Andres., *Role Playing Games : Comparative Analysis Across Two Media* Platforms, Proceedings of the 3rd Australasian conference on Interactive entertainment, Murdoch University, p. 76, 2006.

[8] Jamilah. D, Sufian, I., *Object-Oriented Design Model*, International Journal of Computer Science and Network Security, VOL 9 No. 10, October 2009.

[9] Booch, G., Object-Oriented Analysis and Design with Applications, The Benjamin/Cummings Publishing Company, Inc, 1994.

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Aliya Fathma Najihati, lahir pada tanggal 28 Agustus 1995 di Cirebon, Jawa Barat. Hobi yang dimiliki adalah bermain game, membaca, menulis, dan menonton film. Penulis menempuh pendidikan mulai dari SDN 3 Wonogiri (2001-2007), SMPN 1 Wonogiri (2007-2010), SMAN 3 Solo (2010-2013), dan Teknik Informatika ITS (2013-2017). Di jurusan Teknik Informatika ITS, penulis mengambil bidang minat Interaksi Grafika dan Seni (IGS) dan memiliki ketertarikan dalam eksplorasi teknologi di bidang *game*. Selama perkuliahan, penulis aktif dalam organisasi kemahasiswaan, antara lain staff departemen Dalam Negeri Himpunan Mahasiswa Teknik Computer-Informatika 2014, Staff Ahli Himpunan Mahasiswa Teknik Computer-Informatika 2015, Vice-Manager departemen Internal Affairs Badan Eksekutif Mahasiswa FTIF 2015, International Relation Manager departemen Incoming Global Internship AIESEC Surabaya, Anggota Sie Hubungan Masyarakat Schematics 2014, Anggota National Seminar Of Technology Schematics 2015, Penerjemah LPPM ITS. Penulis dapat dihubungi melalui surel : aliya.fathma13@gmail.com.