RANCANG BANGUN GAME RPG SWORD MASTER OF DESTINY DENGAN TACTICAL BATTLE

SKRIPSI

Disusun Oleh :

Mukhammad Dian Amirul Azmi
09.18.065

PROGRAM STUDI TEKNIK INFORMATIKA S-1

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

2014

LEMBAR PERSETUJUAN DAN PENGESAHAN

RANCANG BANGUN GAME RPG SWORD MASTER OF DESTINY DENGAN TACTICAL BATTLE

SKRIPSI

Disusun dan Diajukan untuk melengkapi dan memenuhi persyaratan guna mencapai Gelar Sarjana Teknik Informatika Strata Satu (S-1)

Disusun Oleh :
Mukhammad Dian Amirul Azmi
09.18.065

Diperiksa dan Disetujui,

Dosen Pembimbing I	Dosen Pembimbing II
IR. YUSUF ISMAIL NAKKODA, MT	**NURLAILY VENDYANSYAH, ST**
NIP. 1018800189	

Ketua Program Studi Teknik Informatika S-1

Joseph Dedy Irawan, ST, MT
NIP. 19740416200501 1002

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

Ucapan Rasa Syukur dan Terima Kasih Kepada
Tuhan Yang Maha Esa Serta Ayah dan Ibu

RANCANG BANGUN GAME RPG SWORD MASTER OF DESTINY DENGAN TACTICAL BATTLE

Mukhammad Dian Amirul Azmi

Program Studi Teknik Informatika S-1
Fakultas Teknologi Industri
Institut Teknologi Nasional Malang
Jl. Raya Karanglo Km. 2 Tasikmadu-Malang
Email: amiruldian@gmail.com

Dosen Pembimbing: 1. IR. YUSUF ISMAIL NAKOODA, MT
2. NURLAILY VENDYANSYAH, ST

ABSTRAK

Latar Belakang adalah masalah yang terjadi di masyarakat yang menyebabkan penulis mengambil judul skipsi ini. Salah satu contoh pengguna game online sangat banyak digemari oleh kalangan anak-anak dalam sistem informasi jarak jauh. Disadari sekarang ini masyarakat luas hanya mengetahui informasi dari mulut kemulut. Dan juga jarak tempuh yang mengakibatkan kurangnya informasi secara lengkap.

Tujuan Penelitian adalah yang pertama diharapkan dapat memberikan manfaat dan kemudahan dalam mengakses game yang ada di Indonesia khususnya game petualangan.

Kedua, supaya penulis dapat memberikan teori-teori yang telah diterima sekaligus mematangkan dan meningkatkan pengetahuan tentang perancangan pembangunan game tersebut dengan menggunakan aplikasi yang tersedia dengan Metode Knowledgebase adalah cara yang dipakai penulis dalam membuat game untuk menulis skripsi. Metode pengambilan data yang dilakukan dalam kegiatan ini meliputi metode kwonledgebase untuk menampilkan pilihan pada list search pada program "RANCANG BANGUN GAME RPG SWORD MASTER OF DESTINY DENGAN TACTICAL BATTLE yang telah dibuat oleh penulis sebagai game yang bisa menunjang anak untuk bisa berimprovisasi merancang game-game yang lebih baik dan digunakan sebagai tempat untuk membuat layout dari game yang telah dirancang. Dan, melakukan studi kepustakaan bisa berupa buku dan sumber-sumber informasi dari internet. Pada pengembangan game yang akan datang penulis berharap untuk anak-anak dimasa yang akan datang.

Hasil yang ingin dicapai adalah pertama semoga program bermanfaat buat masyarakat luas khususnya dalam game ini adalah pengguna bisa memudahkan untuk bermain game petualangan. Kedua, semoga program bermanfaat untuk Lembaga yang terkait untuk menunjang para gammer yang ada di Indonesia.

Kesimpulan adalah setelah penulis telusuri dan pelajari dari hasil dan perancangan pada skripsi ini. Dan juga saran yang berguna dan bermanfaat untuk penyempurnaan pengembangan lebih lanjut dan memberikan kemudahan

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan YME atas karunia, rahmat dan berkatNya, sehingga penulis dapat menyelesaikan penelitian yang berjudul **"RANCANG BANGUN GAME RPG SWORD MASTER OF DESTINY DENGAN TACTICAL BATTLE"**.

Skripsi ini dapat terselesaikan tidak terlepas dari dukungan berbagai pihak. Penulis mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Bapak Ir. Soeparno Djiwo, MT selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. H. Anang Subardi, MT selaku Dekan Fakultas Teknologi Industri Institut Teknologi Nasional Malang.
3. Bapak Joseph Dedy Irawan, ST, MT selaku Kepala Jurusan Teknik Informatika S-1 Institut Teknologi Nasional Malang.
4. Bapak Ir. Yusuf Ismail Nahkoda, MT selaku Dosen Pembimbing I yang telah memberikan saran dan bimbingannya dalam penyusunan laporan ini.
5. Bapak Nurlaily Vendyansyah, ST selaku Dosen Pembimbing II yang telah memberikan saran dan bimbingannya dalam penyusunan laporan ini.
6. Bapak dan Ibu Dosen yang telah mengajar penulis selama studi di Institut Teknologi Nasional Malang.
7. Ayahanda Abd. Munif dan Ibunda Sunanik yang selalu memberikan do'a restu, dorongan dan semangat.
8. Rekan-rekan yang turut membantu dalam penyelesaian laporan ini.

Semoga apa yang telah disajikan dapat memberikan manfaat dan pengetahuan bagi para pembaca. Segala kritik dan saran yang bersifat membangun, diterima dengan senang hati sebagai tambahan ilmu pengetahuan.

Malang, 24 Agustus 2014

Penulis

DAFTAR ISI

BAB III PERANCANGAN APLIKASI

BAB IV IMPLEMENTASI DAN PENGUJIAN

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan *game* baru-baru ini mengalami kemajuan pesat, mulai dari *game* sederhana yang dibuat oleh para pengembang *game* biasa hingga *game* buatan *vendor* (perusahaan) ternama. *Game* yang dibuat terdiri dari bermacam-macam *Genre* (jenis) mulai dari *game Shooting, Arcade, Strategy, Fantasy, RPG* (*Role Playing Game*) hingga *MMORPG* (*Massively Multiplayer Online Role Playing Game*). Banyaknya jenis *game* yang ada sekarang, membuat mayoritas orang baik itu kalangan anak muda, pria, wanita, remaja, hingga orang tua tidak bisa lepas dari *game* dan menjadikannya sebagai alternatif hiburan tidak hanya untuk melepas lelah atau sekedar mengisi waktu luang.

Semakin berkembangnya *game* yang telah dibuat dengan memperhatikan kualitas grafik dan ide cerita yang semakin bagus, hal ini membuat *game-game* lama dilupakan oleh sebagian orang dan dianggap tidak menarik lagi untuk dimainkan, padahal *game* jenis lama merupakan tonggak sejarah berdirinya *game*.

Dari permasalahan di atas maka muncul suatu ide tentang bagaimana merancang sebuah *game* sederhana bergenre *RPG* (*Role Playing Game*) dengan judul *Sword Master Of Destiny* menggunakan software *RPG Maker VX Ace*, dengan *grid battle system atau tactical battle system* jenis *Classical RPG* (*Role Playing Game*). Ide cerita dari *game* ini adalah seorang pemuda yang bertualang untuk mengalahkan kejahatan, sambil mengasah kemampuan berpedangnya.

Dalam pembuatan *game* ini diterapkan juga suatu metode untuk sistem *battle* (pertarungan) antara *player* dengan musuh menggunakan metode *Tactical Battle*. Metode *Dice Roll* adalah metode yang digunakan dalam sistem pertarungan antara *player* dengan musuh yang saling menyerang secara bergantian.

1.2. Rumusan Masalah

Adapun rumusan masalah dari latar belakang di atas adalah bagaimana cara merancang dan menciptakan suatu *game* dengan *genre RPG* (*Role Playing Game*) dengan judul *Sword Master Of Destiny* dengan menggunakan metode *Tactical Battle atau Grid Battle System*.

1.3. Tujuan Penulisan

Tujuan dari penyusunan skripsi ini adalah untuk menciptakan dan merancang *game* bergenre *RPG* (*Role Playing Game*) dengan menggunakan metode *Tactical Battle atau Grid Battle System*.

1.4. Batasan Masalah

Adapun batasan-batasan masalah dalam pembuatan program ini adalah sebagai berikut :

1. Dalam dialog antar karakter, tidak terdapat suara asli yang mewakili percakapan antar karakter tersebut, hanya terdapat teks dialog yang ditampilkan.
2. Dalam dialog antar karakter, juga tidak terdapat efek suara saat dialog antar karakter terjadi, hanya terdapat kata-kata yang diucapkan saat dialog berlangsung.
3. Terdapatnya karakter-karakter bawaan yang sudah tersedia dalam engine *game RPG Maker VX Ace*. Jika seseorang ingin menambahkan dan membuat sendiri karakternya, maka orang tersebut dapat membuat sendiri karakternya dengan menggunakan *Character Generator* pada *toolbar RPG Maker VX Ace*.
4. *Tools* yang digunakan hanyalah *tools* yang berhubungan dengan pembuatan *game RPG* (*Role Playing Game*) ini yang meliputi pembuatan cerita (*Storyline*), peta *game* (*Maps Game*), musik dan efek suara, dan hal-hal yang berkaitan dengan *game*.
5. Perangkat lunak yang digunakan untuk pembuatan *game* ini adalah *RPG Maker VX Ace*, *Audacity*, *Adobe Photoshop CS3*.

1.5. Metodologi Penelitian

Metode pengembangan sistem yang digunakan dalam menyelesaikan *game* jenis *RPG* ini menggunakan model *RAD* (*Rapid Application Development*) yang terdiri atas:

1. Tahapan Perencanaan Kebutuhan

   Pada tahapan perencanaan kebutuhan ini, sang pembuat *game* merencanakan fungsi apa saja yang terdapat dan dibutuhkan dalam suatu *game* yang ingin diciptakannya. Sebagai contohnya adalah inti cerita dari suatu *game* yang akan dibuat dan terdapat tujuan yang jelas yang harus diraih dalam *game* tersebut. Sang pembuat *game* harus membuat terlebih dahulu alur cerita (*Storyboard*), setelah alur cerita selesai dibuat hal yang dilakukan selanjutnya adalah mempersiapkan karakter-karakter yang akan dimainkan, dan efek suara dalam *game*.

2. Tahapan Perancangan Penggunaan

   Pada tahapan perancangan penggunaan ini, sang pembuat *game* mulai mendesain Peta (*Maping*) yang merupakan lahan atau tempat suatu karakter akan bergerak nantinya. *Maping game* bisa meliputi wilayah utama (*Field*), kota-kota, tempat bersejarah, dan tempat pertarungan antar karakter nantinya. Selain proses *Maping*, pembuat *game* juga diharuskan membuat *NPC* (*Non Playable Character*) agar *game* yang dibuat terlihat lebih realistis. *NPC* (*Non Playable Character*) adalah suatu karakter tambahan yang berfungsi membantu *Player* menyelesaikan setiap misi dengan memberikan suatu petunjuk berupa langkah-langkah apa saja yang akan dilakukan oleh *Player* nantinya.

3. Tahapan Implementasi

   Pada tahapan implementasi, pembuat *game* menambahkan beberapa script dalam *game* yang dibuat untuk melengkapi fungsi system tambahan yang terdapat di dalam *game*.

4. Tahapan Pengujian

   Pada tahapan ini dilakukan pengujian dan analisis pengujian terhadap *game* yang dibuat menggunakan *RPG Maker VX Ace* untuk mengetahui apakah sistem berjalan sesuai dengan yang diharapkan.

1.6. Sistematika Penulisan

Untuk mempermudah dan memahami pembahasan penulisan skripsi ini, maka sistematika penulisan disusun sebagai berikut :

BAB I : PENDAHULUAN

Berisi tentang latar belakang, perumusan masalah, tujuan dan manfaat penulisan, batasan masalah, metodologi dan sistematika penulisan.

BAB II : LANDASAN TEORI

Berisi tentang tinjauan pustaka, menguraikan teori-teori yang mendukung judul, dan pembahasan secara detail. Landasan teori 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 : ANALISIS DAN PERANCANGAN SISTEM

Berisi tentang analisa kebutuhan sistem baik software maupun hardware yang diperlukan untuk membuat kerangka global yang menggambarkan mekanisme dari sistem yang akan dibuat.

BAB IV : IMPLEMENTASI DAN PENGUJIAN

Berisi tentang implementasi dari perancangan yang dibuat serta pengujian aplikasi, struktur, dan tampilan aplikasi.

BAB V : PENUTUP

Berisi tentang inti sari dari hasil pembahasan yang berisikan kesimpulan dan saran yang dapat digunakan sebagai pertimbangan untuk pengembangan penulisan selanjutnya.

BAB II
LANDASAN TEORI

2.1. *Game* dan Pengertiannya

Game memiliki pengertian yang sangat luas, bisa diartikan sebagai suatu permainan, dan bisa juga diartikan sebagai hiburan yang dapat diwujudkan melalui media elektronik maupun non elektronik yang dapat dimainkan oleh banyak orang. Menurut para pakar *game* dan sosiolog dunia menyatakan *game* adalah kegiatan yang melibatkan keputusan pemain yang berupaya untuk mencapai tujuan tertentu yang dibatasi dengan suatu aturan. Aturan-aturan tersebut menjadi acuan yang diterapkan dalam *game* dan digunakan untuk mengarahkan pemain dalam menyelesaikan suatu alur cerita dari *game* tersebut.

Game yang dibuat oleh para pakar menggunakan berbagai macam software aplikasi. Salah satunya adalah dengan menggunakan *Game Engine*. *Game Engine* adalah sebuah sistem perangkat lunak (*Software*) yang dirancang untuk pembuatan dan pengembangan suatu video *game*. *Game Engine* memberikan kemudahan dalam menciptakan konsep sebuah *game* yang akan di buat oleh para *Developer Game*. Mulai dari sistem *rendering*, *physics*, arsitektur suara, *scripting*, kecerdasan buatan (*Arificial Intelligence*), dan bahkan sistem *networking*. *Game engine* dapat dikatakan sebagai jiwa dari seluruh aspek sebuah *game*.

Terdapat banyak mesin permainan yang dirancang untuk bekerja pada beberapa *konsol* video *game* dan sistem operasi desktop seperti *Microsoft Windows, Linux, dan Mac OS X*. Fungsionalitas inti biasanya disediakan oleh mesin permainan mencakup mesin render (*renderer*) untuk *grafis* 2D atau *grafis* 3D, mesin fisika atau tabrakan (*fisika momentum*), keluaran suara, bahasa pemrograman yang diintegrasikan (*script programming*), animasi yang dipakai, kecerdasan buatan, jaringan, *streaming*, manajemen memori, *threading*, dukungan lokalisasi, dan adegan grafik. Proses pengembangan permainan sering dihemat sebagian besar menggunakan kembali mesin permainan yang sama untuk menciptakan permainan yang berbeda.

2.2. *RPG (Role Playing Game)*

RPG adalah salah satu genre *game*, yang merupakan singkatan dari *Role Playing Game*. Sesuai dengan namanya, *player* diajak untuk bermain peran sebagai karakter yang berada pada *game* tersebut. Dalam *game* ini juga *player* dapat berperan sebagai orang lain, dan biasanya mengendalikan lebih dari satu tokoh, biasanya tiga atau empat, yang akan dimainkan dalam waktu bersamaan.

2.2.1 Elemen-elemen pada *RPG (Role Playing Game)*

Dalam permainan *RPG* (*Role Playing Game*) terdapat beberapa hal yang perlu diperhatikan saat kita ingin mengembangkan permainan *RPG* (*Role Playing Game*) menurut cerita dan versi kita sendiri. Hal inilah yang menjadi acuan dan dasar dari alur cerita *game* yang akan dibuat. Elemen-elemen tersebut ditunjukkan sebagai berikut:

1. Storyline dan Character Development

Cerita yang disajikan dalam *game RPG* (*Role Playing Game*) harus terkesan kreatif, inovatif, dan dapat menceritakan perkembangan karakter utama serta pemain pendukung dalam *game* yang berperan membantu karakter utama serta inti cerita dari suatu *game* tersebut. Pada elemen *Storyline* dan *Character Development* ini, inti cerita *game* yang akan dibuat, disusun dan dikonsep sebaik mungkin dengan menuangkan semua ide cerita pada *Storyboard*. *Storyboard* digunakan untuk membantu mengonsep semua ide cerita dan langkah per langkah apa-apa saja hal yang terdapat di dalam *game* yang akan dibuat.

Hal-hal tersebut bisa mencakup saat *player* bertemu musuh dan rekan-rekan dalam perjalanan, saat *player* dihadapkan pada persoalan sulit dan hampir tidak ada jalan keluar, saat musuh menguasai suatu wilayah, saat karakter pendamping mengkhianati *player*. *Storyboard* bisa juga digunakan untuk mengatur alur cerita dari karakter utama, hal apa yang harus dilakukan *player* jika menghadapi masalah tertentu, jalur manakah yang akan dilewati oleh *player* setelah *player* berhasil menyelesaikan misinya yang lama untuk kemudian melanjutkan misi barunya. *Storyboard* yang perlu diperhatikan dalam *game RPG* (*Role Playing Game*) adalah sebuah pergerakan yang jelas dari tokoh utama, Pemain pendukung (rekan dari tokoh utama), kekasih

tokoh utama, musuh bebuyutan, dan tujuan akhir yang ingin diraih oleh tokoh utama (tujuan akhir dalam *game*).

2. Battle System

Battle System adalah teknik pertarungan yang terdapat pada *game RPG* (*Role Playing Game*). *Battle System* terdiri dari beberapa teknik, sebagai berikut:

a. Wait and See Battle

RPG (*Role Playing Game*) yang menggunakan sistem ini disebut juga *old school RPG* (*Role Playing Game*), karena memang sistem ini dulu sering digunakan dan pertama kali muncul. Pada teknik *game* ini, *player* memberi perintah pada setiap *hero* pada posisi masing-masing. Setelah selesai, *player* tinggal *wait and see*. Serangan dilakukan bergiliran antara *hero* dan musuh. Periode ini dinamakan satu ronde.

Pada *RPG* (*Role Playing Game*) jenis ini, ada 3 jenis serangan:

1) ***Normal battle***, yaitu *player* dan musuh sama sama mendapat giliran pada ronde pertama.

2) ***First strike battle***, yaitu *player* mendapat kesempatan untuk memulai ronde pertama tanpa musuh menyerang.

3) ***Ambushed battle***, yaitu *player* mendapat serangan terlebih dahulu oleh musuh pada ronde pertama. Pada serangan jenis ini, *player* baru mendapat giliran saat ronde kedua. Siapa dulu yang boleh menyerang ditentukan melalui suatu perhitungan sederhana. Tiap *hero* dan musuh memiliki kemampuan khusus yang disebut *Agility, Speed*, atau yang sejenisnya. Dalam sistem lama *game RPG (Role Playing Game)*, karakter yang memiliki nilai *Agility* terbesar boleh mulai duluan, diikuti *Agility* berikutnya, dan seterusnya.

b. Real Time Battle

Berbeda dengan *Wait and see Battle system*, pada *RPG* (*Role Playing Game*) ini *player* langsung berhadapan dengan musuh. Karakter tidak diberikan berbagai macam perintah saat akan menyerang, namun karakter akan langsung menyerang karakter lain secara frontal. Jadi,

tidak ada yang namanya perintah *Attack, Defense, Magic, Item*, dan sebagainya saat karakter akan bertarung. Tidak ada perhitungan *Agility* seperti sistem *wait and see battle* untuk karakter yang berhak menyerang pertama kali maupun menyerang kedua kali. Contoh *RPG* (*Role Playing Game*) yang menggunakan sistem ini adalah semua seri *game Legend of Zelda, Diablo*, dan lain-lain.

3. *Menu System*

Menu System berisi mengenai petunjuk dan beberapa manual yang ada dalam *game RPG* (*Role Playing Game*) tersebut. Menu sistem sangat membantu *player* dalam memahami konsep *game* sebenarnya sebelum *player* memainkan karakter yang telah dipilihnya.

a. *Mulai Baru* : Menu di mana *player* akan memulai memainkan *game*.
b. *Lanjutkan* : Menu di mana *player* melanjutkan lagi permainannya dari simpanan data *game* yang telah disimpan sebelumnya.
c. *Shut down* : Menu di mana *player* keluar dari game.

4. *Status Effect*

Status effect adalah keadaan yang menunjukkan berbagai macam efek dari suatu perilaku yang ditunjukkan oleh karakter (*Player* atau musuh) dan dapat mempengaruhi pergerakan karakter (*Player* atau musuh) tersebut. *Status Effect* terdiri dari beberapa macam, yaitu:

a. Paralysis

Status ini menyebabkan karakter yang bersangkutan tidak bisa bergerak maupun mengeluarkan kemampuannya sampai efek dari status ini menghilang. Menghilangkan efek dari status ini adalah dengan memberikan obat penawar pada karakter (*Player* atau musuh).

b. ***Confuse***

Status ini cukup merugikan karakter. Karakter (*Player* atau musuh) tidak akan bisa menyerang lawan dengan akurat atau serangan akan mengenai rekan sendiri dikarenakan karakter tersebut pusing dan hilang kesadaran sejenak. Status efek ini dapat dihilangkan dengan memberikan obat penawar, atau tetap membiarkannya berlangsung karena status ini akan hilang dengan sendirinya setelah 2 atau 3 kali putaran giliran permainan.

c. ***Blind***

Status ini hampir sama dengan *Confuse*, akan tetapi *status effect Blind* mengakibatkan karakter tidak mampu menyerang lawan secara akurat. Hal ini menyebabkan serangan karakter *"NULL"* pada lawan. *"NULL"* berarti tidak berdampak apa-apa pada lawan.

d. ***Sleep***

Salah satu status yang juga tidak menguntungkan bagi karakter (*Player* atau musuh), karakter yang terkena effect ini akan tertidur untuk beberapa saat dan akan terbangun kembali setelah beberapa giliran permainan atau bisa juga disadarkan dengan menggunakan obat penawar yang sudah dimiliki.

2.2.2 Kategori-kategori pada *RPG (Role Playing Game)*

Game RPG (Role Playing Game) juga memiliki kategori yang membaginya menjadi beberapa kelompok, hal ini ditunjukkan sebagai berikut:

1. ***Classical***

Classical RPG (Role Playing Game) adalah salah satu kategori dari *game RPG(Role Playing Game)* yang mengharuskan *player* memecahkan suatu teka-teki didalam *game*, untuk dapat melangkah ke tahap berikutnya. Dapat juga disebut sebagai *game RPG (Role Playing Game)* bersyarat. Apabila *player* tidak dapat menyelesaikan teka-teki atau masalah yang sedang dihadapi, *player* tidak akan bisa meneruskan permainan dan hanya akan terhenti pada masalah yang sedang dihadapi. Pada umumnya, *Classical*

RPG (Role Playing Game) memberikan sebuah teka-teki pada player berupa *Quest*. *Quest* bisa berisi teka-teki yang sedang dihadapi, misi dari organisasi tertentu, menolong *NPC (Non Playable Character)*, dan lain-lain. *Quest* harus bisa diselesaikan oleh *player* agar *player* dapat meneruskan langkah selanjutnya.

2. ***Fantasy***

   Kategori *fantasy* memiliki hal yang paling mencolok pada tata letak settingnya, rata-rata setting cerita kategori *fantasy* menceritakan tentang dunia zaman pertengahan dimana orang-orang pada waktu itu mengenal ilmu gaib yang dinamakan sebagai sihir. Hal ini diimplementasikan pada tiap-tiap karakter dalam *game* yang mampu menggunakan *magic* (sihir) sebagai kekuatan dalam bertarung. Contoh: *Legend Of The Dragoon*, *Ninja Saga*, dan *Eden*.

3. ***Sci-Fi (Science Fiction)***

   Kategori *Sci-Fi* merupakan kebalikan dari kategori *fantasy RPG* (*Role Playing Game*), setting cerita pada kategori ini menggambarkan tentang zaman yang jauh lebih maju dari zaman sekarang. Istilah *magic* (sihir) diganti dengan istilah semacam *Telekinesis* atau *Psy*. Contoh: *Star Wars* dan *Sword Art Online*.

4. ***Historical***

   Kategori *RPG* (*Role Playing Game*) *historical* lebih menekankan pada aspek sejarah dimana kegiatan dan hal-hal sebenarnya yang dilakukan manusia benar-benar terjadi di masa lalu, dan juga mengisahkan tentang perang antar kerajaan yang saling berebut wilayah kekuasaan. Ciri khas utama kategori *historical* ini adalah settingnya yang menggambarkan peperangan dan perselisihan antar kerajaan. Contoh: *Age of Empire*.

5. ***Horror***

   Kategori *RPG* (*Role Playing Game*) *Horror* jarang diadaptasi oleh para pembuat *game* dikarenakan setting cerita kurang mendongkrak minat *player* untuk memainkannya, namun para

pembuat *game* berusaha sebaik mungkin untuk menghasilkan ide cerita yang menegangkan dari *genre horror* ini. Ciri utama dari *genre* ini adalah pertarungan antara manusia biasa melawan makhluk-makhluk gaib seperti *vampire*, *werewolf* (manusia serigala), *zombie*, dan *mutan*. Contoh: yang paling terkenal dari *genre game* ini adalah semua seri *Resident Evil*.

6. ***Funny***

   Kategori *RPG* (*Role Playing Game*) *funny* merupakan kategori yang cocok untuk dimainkan oleh semua kalangan karena mengandung unsur menyenangkan dan mudah dipahami saat memainkannya. Dalam kategori *funny*, ide cerita yang disuguhkan sangat sederhana dan dibuat semenarik mungkin dan *player* diberikan petunjuk yang kadangkala terlihat jelas saat memainkan *game* tersebut. Contoh: *Super Mario* dan *Super Mario Bros.*

7. ***Multigenre***

   Kategori *Multigenre* adalah kategori *game RPG* (*Role Playing Game*) yang terdiri dari 2 atau lebih kategori. Misalnya pada sebuah *game* terdapat ide cerita dimana pertarungan antar kerajaan terjadi pada zaman dulu dan karakter-karakternya mampu menggunakan kekuatan *magic* (sihir) untuk bertarung dengan karakter lain, kategori *game* tersebut adalah *historical* dan *fantasy*. Contoh lain dari kategori *Multigenre* adalah: *Castlevania* (*horror – fantasy*).

2.3. Metode *Tactical Battle System*

Metode *Tactical Battle System* adalah metode yang diterapkan pada sistem pertarungan antar karakter (*player* melawan musuh) dengan menggunakan grid squere, position Ability, dan turn system yang mempunyai fungsi sebagai berikut:

1. *Grid Squere* berfungsi sebagai tempat dimana untuk meletakkan posisi tiap-tiap karakter. Penempatan karakter *player* ditentukan oleh pemain. Selain penentuan posisi awal, juga berfungsi sebagai penentu posisi karakter saat melakukan serangan pada musuh.

2. *Position Squere Ability* memiliki fungsi yang berbeda Grid Squere. *Position Squere Ability* berfungsi untuk meletakkan karakter pada setiap posisi.

3. *Turn* System berfungsi sebagai penghitungan waktu untuk melakukan serangan terlibih dahulu.

Pergerakan karakter didalam sistem pertarungan yang telah ditambah dengan Grid squere, position Ability, dan Turn system musuh dapat bergerak bervariasi. Hal tersebut membuat *player* dan musuh terlihat bergerak berputar-putar.

2.4. *RPG Maker VX Ace*

RPG Maker VX Ace adalah software *game engine* seri kelima dalam seri PC *RPG Maker*, dimulai dari seri *RPG Maker 2000*, *RPG Maker 2003*, *RPG Maker XP*, *RPG Maker VX*, dan *RPG Maker VX Ace* . Seperti *RPG Maker* sebelumnya, *RPG Maker VX Ace* memiliki antarmuka grafis *user-friendly* (sangat bersahabat) yang sudah ditingkatkan set fiturnya sehingga sangat mengagumkan dalam pengolahan *maping* (perancangan peta dalam *game*), pembuatan karakter dengan bantuan *tools Character Generator*, dan penentuan atribut-atribut dari suatu karakter yang akan dibuat nantinya. *RPG Maker VX Ace* menggunakan Database yang terintegrasi didalam *engine software* dalam pengelolaan segala atribut dari karakter, *item*, jurus yang akan dipakai dan bahan-bahan sistem yang semakin dilengkapi, memungkinkan bahkan bagi pemula untuk menciptakan *role-playing game*nya sendiri tanpa perlu memiliki pengetahuan pemrograman.

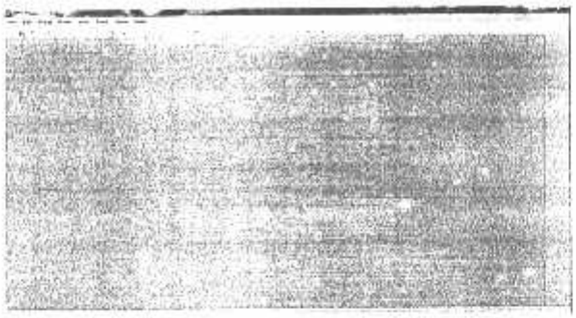
RPG Maker VX Ace memiliki fitur yang semakin diperbarui, selain fiturnya yang menarik, keefektivannya yang dapat membuat siapa saja cepat memahami dan mendesain *game* sesuai keinginan mereka. *RPG Maker VX Ace* dilengkapi dengan *tools tileset* yang sudah tersedia seperti *pintu, losmen, gunung, laut, gua, karakter, player, item* dan sebagainya. *Tools* pembuatan karakter dengan *Character Generator*, *tools* pengelola *database*, *tools eventing* (adalah *tools* dimana kita dapat mengolah *game* buatan kita semenarik mungkin), dan lain-lain.

2.4.1 Window window pada *RPG Maker VX Ace*

Ada beberapa macam *window* yang tersedia dalam *RPG Maker VX Ace*, yang dapat dijelaskan sebagai berikut:

1. *Window Editor RPG Maker VX Ace*

Merupakan *window* yang dibuka pertama kali saat *RPG Maker VX Ace* dijalankan. *Window Editor RPG Maker VX Ace* ditunjukkan pada gambar 2.1.

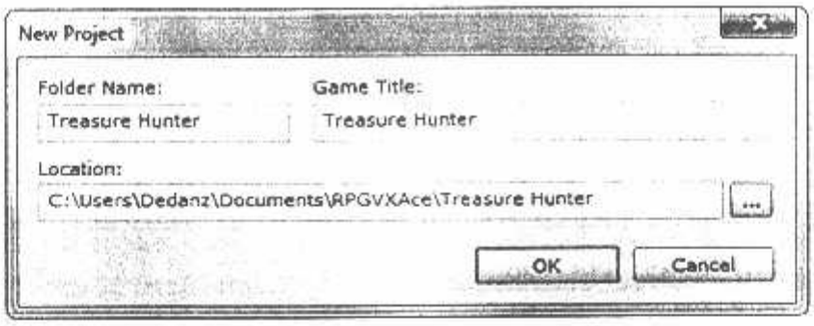


Gambar 2.1 *Window Editor RPG Maker VX Ace.*

Pada *window editor*, para pembuat *game* tidak akan bisa membuat langsung *game* yang akan dirancangnya dikarenakan jendela *editor* bukan berfungsi sebagai *project* untuk memulai proses perancangan *game*. Jika ingin melakukan perancangan, kita diharuskan membuat *project* terlebih dahulu. Yaitu dengan cara klik File – New Project, maka akan muncul *window* New Project yang berfungsi sebagai tempat untuk perancangan *game* nantinya. Cara membuat project baru ditunjukkan pada gambar 2.2.

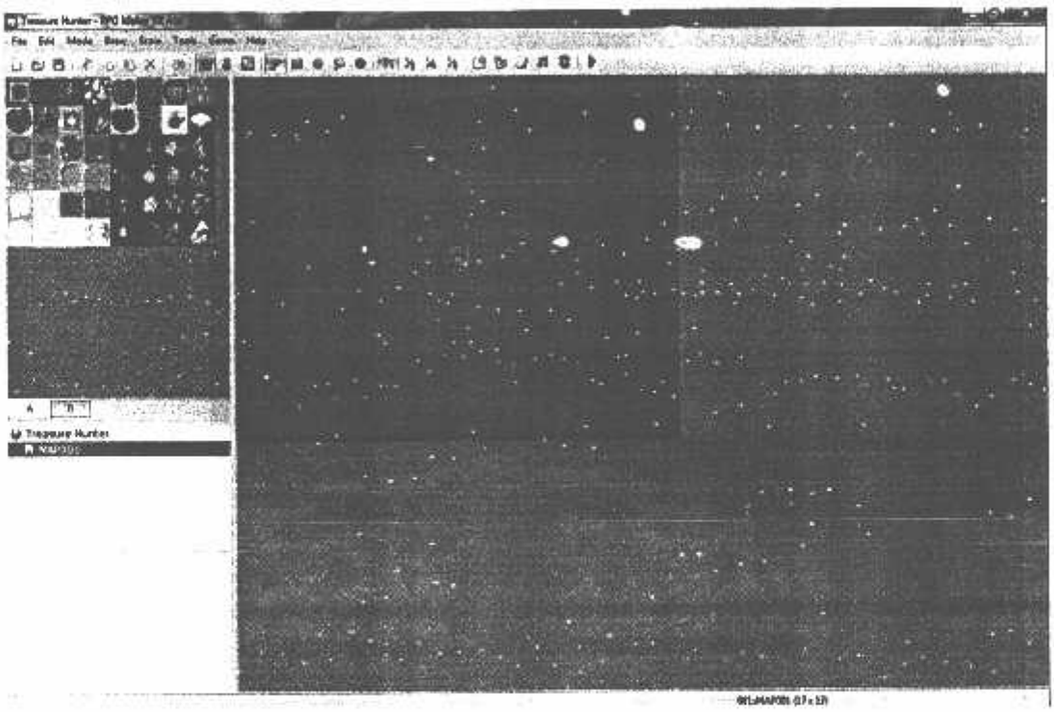
Gambar 2.2 Pembuatan proyek baru.

Maka akan muncul window New Project. Pada window New Project, kita diharuskan memberi nama project yang akan kita buat. Karena project kali ini berhubungan dengan *game* bajak laut, maka beri nama project tersebut Treasure Hunter. Lalu klik OK. Window New Project ditunjukkan pada gambar 2.3.



Gambar 2.3 *Window New Project.*

Setelah kita selesai menamai *new project*, barulah rancangan *game* yang akan kita buat bisa dikerjakan di *window project. Window project* ditunjukkan pada gambar 2.4.



Gambar 2.4 *Window Project.*

2. Window Database RPG Maker VX Ace

RPG Maker VX Ace memiliki Database yang sudah tersedia di dalam *engine game* tersebut. *Database* pada *RPG Maker VX Ace* berfungsi untuk mengelola data karakter, *item*, senjata, objek, dan *event* (kejadian) yang terdapat di dalam *game*. Di dalam *window Database* terdapat beberapa sub

menu yang dapat mempermudah kita untuk mengelola data-data selain karakter dan objek. Sub menu tersebut terdiri seperti berikut:

a. Sub menu *Actors* : berfungsi untuk mengelola karakter yang kita inginkan. Kita dapat membentuk karakter, mengatur kuat lemahnya karakter kita, memberikan beberapa atribut, dan bahkan memberinya kekuatan spesial sangat memungkinkan untuk dilakukan.
b. Sub menu *Classes* : pada sub menu *classes*, kita mengelola karakter kita agar karakter memiliki gelar, pangkat atau semacam jabatan yang memiliki peran di dalam *game*.
c. Sub menu *Skills* : pada sub menu *skills*, kita mengelola jenis serangan yang nantinya akan dimiliki oleh tiap-tiap karakter. Bagaimana bentuk serangan dan memiliki dampak sebesar apa bagi musuh semua jurus dan kekuatan karakter diatur dalam menu ini. *Skills* atau kemampuan tersebut kita olah dan kita tentukan sesuai dengan standart dari *game RPG* pada umumnya.
d. Sub menu *Items* : sub menu *item* digunakan untuk mengelola *item* (barang) yang nantinya berfungsi dan dipakai oleh karakter yang dapat membantunya saat berperang maupun saat mendesak. *Item* bisa ditentukan harganya, bisa juga ditentukan apa saja efek yang dimiliki ketika karakter menggunakannya, dan bisa juga *item* tersebut merugikan bagi karakter.
e. Sub menu *Weapon* : sub menu *weapon* digunakan untuk mengelola senjata yang nantinya berfungsi sebagai alat untuk berperang oleh karakter.
f. Sub menu *Armors* : sub menu *armor* digunakan untuk mengelola perlengkapan yang nantinya akan dipakai oleh karakter. Perlengkapan tersebut bisa melingkupi baju zirah, bandana, tameng pertempuran, cincin kekuatan, dan lain-lain.
g. Sub menu *Enemies* : sub menu *enemies* membantu kita dalam mengelola musuh-musuh yang nantinya akan dihadapi oleh karakter. Mulai dari bentuknya, kekuatannya, level musuh yang

akan dihadapi, kelemahan musuh terhadap suatu elemen, dan nilai yang didapat setelah musuh tersebut dikalahkan. Nilai yang dimaksud disini adalah *Nilai EXP*. *Nilai EXP* adalah nilai *Experienced Point* yang didapatkan oleh karakter ketika musuh telah dikalahkan.

h. Sub menu *Troops* : sub menu *troops* berfungsi menentukan kapan dan dimana waktu yang tepat saat-saat musuh akan muncul untuk menyerang karakter. *Troops* bisa diartikan juga sebagai jebakan, dimana jebakan-jebakan tersebut adalah musuh yang keluar untuk menyerang karakter.

i. Sub menu *States* : sub menu *states* digunakan untuk mengatur status atau keadaan yang nantinya dapat dialami oleh karakter yang terdapat di dalam *game*. Status tersebut melingkupi *Poison* (keracunan), *Confuse* (pusing), *Sleep* (tertidur), dan lain-lain.

j. Sub menu *Animations*: sub menu *animation* ini digunakan untuk menentukan semua jenis animasi gerakan. Baik itu animasi saat menyerang, animasi saat karakter sembuh dari status yang berbahaya, animasi serangan, animasi saat musuh keluar menyerang karakter, dan lain-lain

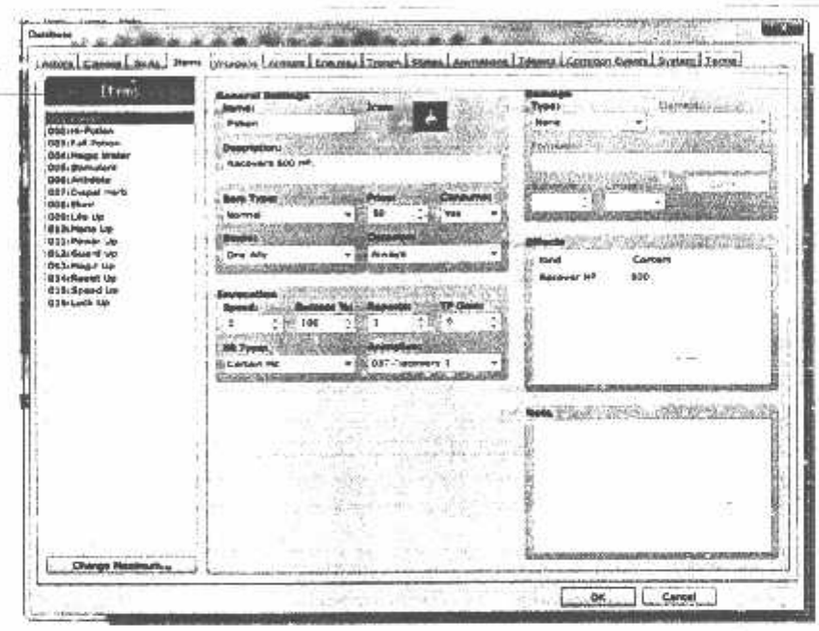
k. Sub menu *Events* : sub menu *event* atau bisa juga disebut *Common Event* adalah sub menu yang berfungsi mengelola setiap kejadian yang terdapat di dalam *game*. Kapan musuh akan menyerang, apa yang harus dilakukan saat karakter selesai mengalahkan musuh, semuanya diatur dan dikelola dalam menu *Common Event*.

l. Sub menu *System* : sub menu *system* digunakan untuk mengolah dan mengatur keseluruhan *system* di dalam *game*. Bisa juga berfungsi memantau aktifitas yang terdapat di dalam *game*.

m. Sub menu *Terms* : sub menu *Terms* adalah sub menu tambahan di dalam *RPG Maker VX Ace* yang berfungsi sebagai media pembantu di dalam pengelolaan *RPG Maker VX Ace*. Sebagai contoh yang paling dominan adalah pengelolaan *vocab*

(olah bahasa) dapat kita rubah pada sub menu *Terms*. Olah bahasa tersebut mencakup perubahan bahasa pada menu dan tampilan ketika bermain *game* nantinya.

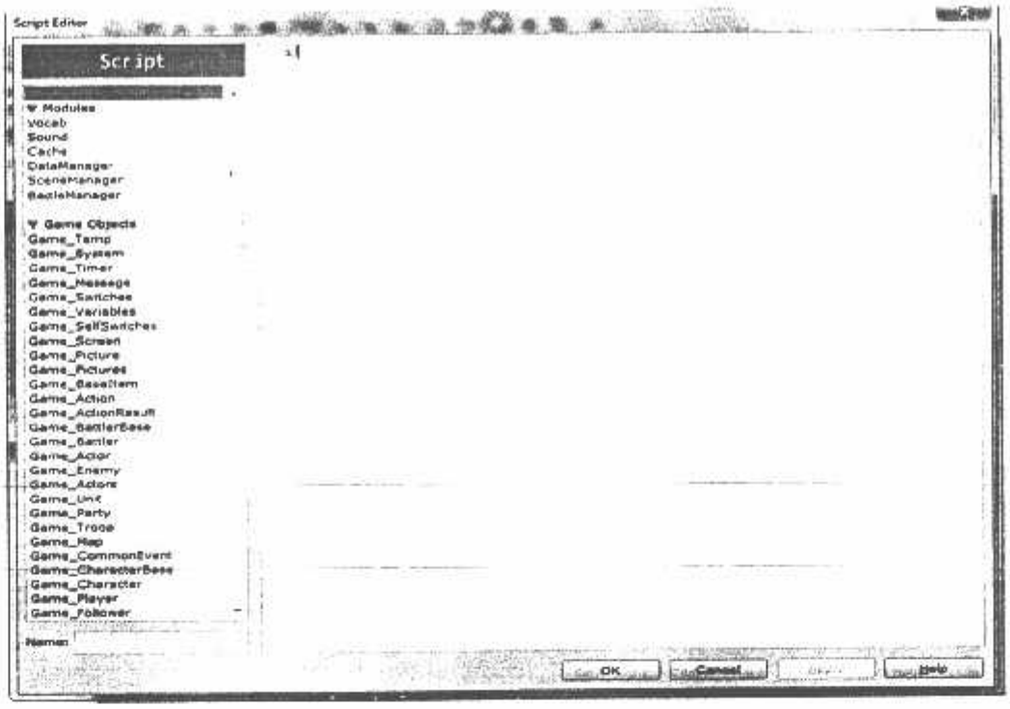
Database membantu kita di dalam mengelola apa-apa saja yang ingin kita buat di dalam *game* nantinya. Baik itu perlengkapan *(item)*, senjata *(weapon)*, kemampuan *(skill)*, animasi pertarungan, musuh yang harus dihadapi, dan karakter yang nantinya akan dimainkan. Window Database ditunjukkan pada gambar 2.5.



Gambar 2.5 *Window Database*.

n. Window Script Editor

Window Script Editor berfungsi untuk menginputkan *script* tambahan yang mendukung kinerja *game* menjadi lebih mengasyikkan baik dari segi tampilan maupun aksi yang dilakukan. *Window Script Editor* ditunjukkan pada gambar 2.6.

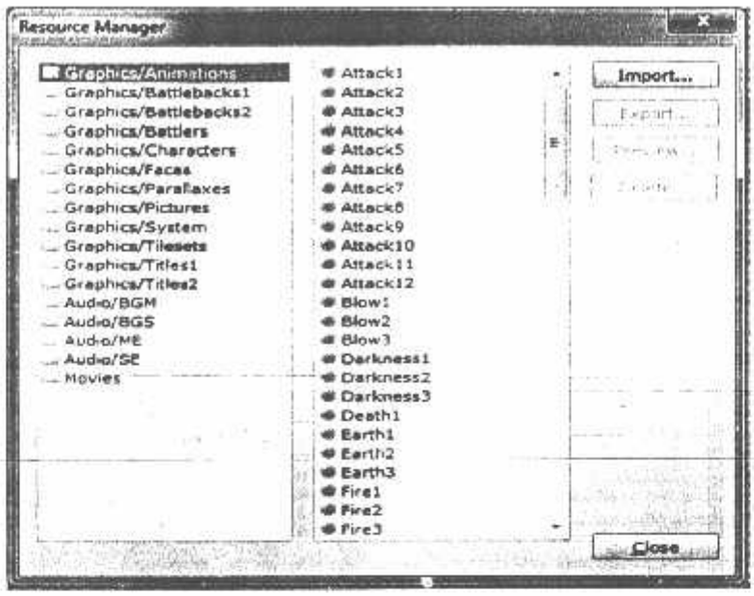


Gambar 2.6 *Window Script Editor*.

o. Window Resource Manager

Window Resource Manager adalah jendela kerja dimana kita dapat meng-import dan mengexport bahan-bahan yang nantinya akan digunakan dalam proses pembuatan *game* pada *RPG Maker VX Ace*. Bahan-bahan tersebut terdiri dari beberapa macam, yaitu desain *sprite* (desain karakter) jika seorang *game maker* membuat karakter mereka sendiri, desain *background* (latar belakang tempat), desain *faces* (desain wajah suatu karakter) yang nantinya menggambarkan wajah dari suatu karakter dalam *game*, desain *system* yang sudah dibuat sendiri, desain pertarungan yang telah dibuat sebelumnya, dan juga tersedia fitur untuk merubah kontrol *sound* (suara) yang menjadi latar *background* music, sisipan suara, dan masukan suara di dalam suatu *game*. Selain menyediakan fitur untuk mengimport dan mengexport desain serta *sound*, di dalam *window Resource Manager RPG Maker VX Ace* menyediakan menu tambahan di mana desain yang sebelumnya sudah tersedia di dalam *engine game* dapat kita hapus semau kita jika desain tersebut benar-benar tidak digunakan dalam *game* atau desain dan fitur yang sudah tersedia bisa tetap dibiarkan. Tujuan penghapusan desain yang tidak terpakai dapat meringankan berat *game* yang

nantinya akan dimainkan. *Window Resource Manager* ditunjukkan pada gambar 2.7.



Gambar 2.7 *Window Resource Manager*.

2.4.2 *RGSS* (*Ruby Game Scripting System*)

RGSS atau disebut juga *Ruby Game Scripting System*, adalah bahasa pemrograman yang berorientasi objek, bahasa pemrograman *Ruby* yang diciptakan oleh Yukihiro Matsumoto digunakan untuk membantu proses *coding* (pembuatan script) pada *RPG Maker VX Ace*. Pada *RPG Maker VX Ace*, bahasa pemrograman *Ruby* yang digunakan bukan lagi *RGSS* melainkan *RGSS2*. *RGSS3* (*Ruby Game Scripting System 3*) adalah bahasa pemrograman *Ruby* generasi ketiga yang juga dikembangkan oleh Yukihiro Matsumoto yang lebih mengacu pada objek-objek di dalam *RPG Maker VX Ace* yang nantinya akan diberi coding untuk proses lanjutan.

Hal-hal yang perlu diperhatikan di dalam *RGSS* adalah sebagai berikut:

1. *Nilai* : adalah angka yang menunjukkan suatu identitas dari objek yang akan dibuat. Semua hal yang ditunjuk memiliki nilai yang menjadi tanda bagi setiap objek. Umur seseorang, *HP* (*Health Point*) suatu karakter, harga suatu benda (*item*) dalam *game*, nomor ID, nomor acak, semuanya adalah nilai yang menunjuk pada angka-angka. Dalam *RPG Maker VX Ace* nilai yang paling banyak muncul adalah nilai *Integer Boolean* dan *String*.

Nilai integer yang digunakan dalam *RGSS3* (*Ruby Game Scripting System 3*) tidak mengacu pada pecahan-pecahan nilai melainkan bilangan bulat yang paling banyak diolah dikarenakan dasar coding mengacu pada bilangan bulat. Sebuah kasus dari nilai *integer Boolean* adalah terdapat nilai on atau off pada switch *RPG Maker VX Ace* serta benar atau salah. Nilai On mengarah pada angka 1 dan nilai Off mengarah pada angka 0 yang memiliki arti nilai 1 adalah benar dan nilai 0 adalah salah.

2. *Variable* : adalah nama (*identifier*) yang mewakili nilai yang tersimpan dalam suatu memori. Saat programmer mengolah suatu nilai di dalam proses *coding* (pengolahan script program), terkadang nilai tersebut perlu disimpan untuk suatu proses lanjutan, nilai tersebut disimpan didalam suatu *variable*. Seorang programmer dapat menggunakan *variable* untuk menggunakan kembali nilai-nilai yang tadi telah diolah sebelum disimpan dalam bentuk *variable*, misalnya seorang programmer dapat melakukan operasi aritmatika pada *variable* numerik. Dalam *RGSS3* (*Ruby Game Scripting System 3*) *variable Ruby* mencakup semua benda (*objek*) yang digunakan, hal ini mengacu pada nama *variable* (pengidentifikasi) tidak dapat dimulai dengan huruf kapital untuk beberapa alasan dan dalam kasus-kasus tertentu. Contoh *variable* ditunjukkan pada gambar 2.8.

```
number1 = 9
number2 = 1
number3 = number1 + number2
cetak number3 # 10
```

Gambar 2.8 *Variable* pada *RGSS3*.

Pada dasarnya contoh pada gambar 2.8 mendeklarasikan dua *variable* yaitu *number1* dan *number2* dengan nilai 9 dan 1. Kita menambahkan dua *variable* dan menyimpan jumlah dalam *variable* ketiga (*number3*). Akhirnya, kita menampilkan nilai dari *variable* ketiga (*number3*). Contoh lain dari *variable* ditunjukkan pada gambar 2.9.

```
# sedikit contoh variable
print 'a = 2'
print 'b = 4'
print 'a + b = 6'
print 'a - b = -2'
print 'b - a = 2'
print 'a * b = 8'
print 'b / a = 2
```

Gambar 2.9 *Variable* lain yang dapat digunakan.

Variable pada dasarnya merupakan pengidentifikasi yang menyajikan nilai-nilai yang tersimpan dalam suatu memori. Nama-nama (pengidentifikasi) *variable* harus dimulai dengan huruf kecil (sebagian besar berdasarkan waktu) dan bersifat *case sensitive*. *Variable* memungkinkan kita untuk menyimpan nilai-nilai yang bervariasi, meskipun semua contoh yang ditampilkan tidak benar-benar menunjukkan *variable* yang terkadang berguna.

3. *Class* : sebuah *Class* (kelas) mendefinisikan tipe data, tipe data dapat berisi *variable* dan metode yang beroperasi pada *variable*. Pembuat *game* dapat menginputkan nilai *string* untuk mewakili teks, *integer* untuk mewakili bilangan bulat, *boolean* untuk mewakili benar atau salah (0 atau 1), dan lain sebagainya. Seperti yang sudah dijelaskan pada *nilai* dan *variable*, tipe data memiliki metode tertentu untuk memanggil fungsi *variable* yang digunakan. Sebagai contoh, tipe data untuk karakter yang bisa dimainkan, masing-masing karakter memiliki *HP* (*Health Point*), *MP* (*Magic Point*), *Level*, dan *variable* lainnya. Dengan *Class* pada *coding* (pembuatan script) *RPG Maker VX Ace* memungkinkan para *game maker* memiliki metode untuk memeriksa apakah karakter dipengaruhi oleh status efek, menghitung kerusakan yang diterima oleh karakter saat diserang oleh karakter lain, melengkapi *item* (benda) tertentu pada karakter, serta mengajarkan karakter keterampilan tertentu (*skills*). Sebagai contoh tipe data yang akan diolah diberi nama (Hero) ditunjukkan dalam *coding* (pengolahan script) ditunjukkan pada gambar 2.10.

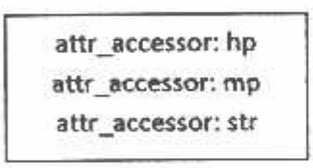
```
Class Hero      # Class Hero mendefinisikan kelas bagi Hero
attr_accessor: hp       #attr_accessor memungkinkan untuk
attr_accessor: mp       #merubah nilai-nilai dari variable
instan
attr_accessor: str
def initialize          #inisialisasi adalah metode yang
@hp = 0                 #digunakan saat pertama kali
@mp = 0                 #membuat objek
@str = 0
end
def equip(sword)        #metode penambahan senjata
print "Hero telah dilengkapi dengan #{sword} "! "
end
def learn_skill(skill)  #metode penambahan kemampuan
print "Hero telah mempelajari #{skill}!"
end
end             #akhir dari Class
```

Gambar 2.10 Contoh *Class*.

Berikut adalah penjelasan dari script *Class Hero*:

Gambar 2.11 *Class Hero*.

Pada gambar 2.11 dijelaskan bahwa dimulailah tahap pembuatan nama *Class*, disertai dengan nama yang diinginkan. Nama harus dimulai dengan huruf kapital (nama-nama *variable* dan metode harus dimulai dengan huruf kecil). Kelas A adalah representasi dari tipe, sehingga nama "Hero" akan sangat bagus untuk mendefinisikan karakter yang akan dimainkan dalam sebuah *game* nantinya.



Gambar 2.12 *Variable* dalam *Class*.

Kelas *Hero* dapat memiliki *variable* misalnya, *variable* yang dapat diakses dimana saja di dalam *Class*. Dalam contoh kasus seperti *Class Hero* yang ditunjukkan pada gambar 2.12, terdapat 3 *variable* yaitu: *@ hp*, *@ mp*, *@ str*. Semua *variable* diawali dengan tanda '*at*' atau *@*.

Akan tetapi, memiliki *variable* instan bukan berarti memungkinkan sang pembuat *game* dapat mengakses *variable* di luar kelas. Untuk dapat mengakses *variable-variable* yang telah dibuat, digunakanlah *accessor atribut* (attr_accessor) sebuah perintah yang diikuti oleh variable_name.

```
def initialize
@ hp = 0
@ mp = 0
@ str = 0
end
```

Gambar 2.13 *Proses pendefinisian inisialisasi.*

Pada proses pendefinisian inisialisasi yang ditunjukkan pada gambar 2.13, sebuah metode yang disebut "initialize" merupakan metode yang disebut ketika sebuah objek dari *Class Hero* dibuat. Untuk membuat sebuah objek dari *Class Hero* (*variable* dari tipe data *Hero*) dapat ditunjukkan pada gambar 2.14.

```
Mikael = Hero.new
```

Gambar 2.14 Pendefinisian *Mikael* sebagai *Hero.*

Setiap kali *compiler* melihat (.new) *compiler* akan memanggil metode *initialize* (jika ada). Sebagian besar waktu, metode *initialize* digunakan untuk mengatur nilai awal dari *variable* instan, dalam contoh *Class Hero* ditetapkanlah nilai 0. Hal tersebut dapat ditunjukkan pada gambar 2.15.

```
def initialize (hp = 0, mp = 0, str = 0)
@ hp = hp
@ mp = mp
@ str = str
end
```

Gambar 2.15 Penetapan nilai dalam *Class.*

Pada contoh script def initialize (hp=0, mp=0, str=0) terlihat bahwa @ hp dan hp berbeda, yang satu adalah *variable* lokal dan yang satunya lagi adalah *variable* instan. Nama-nama tersebut berbeda, dalam hal ini @ str bukanlah str, tetapi str memiliki tambahan @. Untuk menciptakan sebuah objek hp,mp,str dari Class Hero dapat dilakukan langkah yang ditunjukkan pada gambar 2.16.

```
Mikael = Hero.new (9999, 999, 500)
```

Gambar 2.16 Penciptaan objek *Class Hero.*

Dapat juga diberikan sebuah argumen pada proses pendefinisian inisialisasinya yang ditunjukkan pada gambar 2.17.

```
def equip (sword)
print "Hero telah dilengkapi dengan # {sword} "! "
end
def learn_skill (skill)
print "Hero telah mempelajari # {skill}!"
end
```

Gambar 2.17 Penambahan argumen pada *Class Hero.*

Hanya dua metode acak yang mengambil nilai *string* dan mencetaknya menjadi beberapa pesan. Untuk mengakses metode ini dapat dilakukan perintah yang ditunjukkan pada gambar 2.18.

```
Mikael = Hero.new
Mikael.equip("Long Sword")
Mikael.learn("Jet Punch")
```

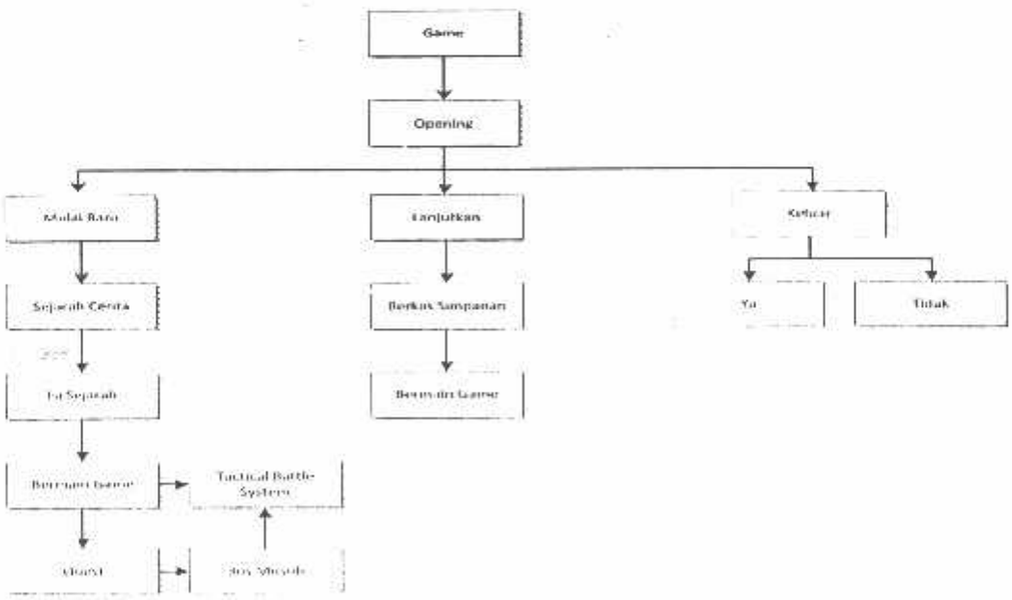
Gambar 2.18 Pengambilan sebuah nilai pada *Class Hero.*

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada tahap perancangan aplikasi, tampilan utama game menampilkan beberapa menu yang terdiri dari *Mulai Baru*, *Lanjutkan*, dan *Keluar* yang mendukung jalannya *game Sword Master of Destiny* dengan Metode *Tactical Battle*.

3.1. Perancangan Struktur *Game* Metode *Tactical Battle*

Pada Perancangan Struktur *Game* dengan Metode *Tactical Battle* adalah tahapan perencanaan dari proses bertarung (pertempuran antar karakter) dalam membangun *Game Sword Master of Destiny* dengan Metode *Tactical Battle* pada *Sistem Battle*. Pada perancangan ini terdapat beberapa tahapan proses yang dapat dijalankan saat pertama kali *Game* sudah dimainkan. *Game* ini mempunyai beberapa menu dan fitur yang sudah disediakan bagi *player* untuk mengakses data berupa Menu mulai baru, lanjutkan, dan keluar. Selain itu, pada *game* ini juga diberikan *tutorial* (petunjuk) cara bermain *game* bagi *player* mengenai hal apa saja yang akan dilakukan *player* nantinya pada saat bermain game yang terdapat pada menu pemilihan karakter. Gambar perancangan struktur ditunjukkan pada gambar 3.1.



Gambar 3.1 Perancangan Struktur *game* dengan Metode *Sword Master of Destiny*

Pada Perancangan Struktur game dengan Metode Tactical Battle dalam gambar 3.1 dapat dijelaskan sebagai berikut :

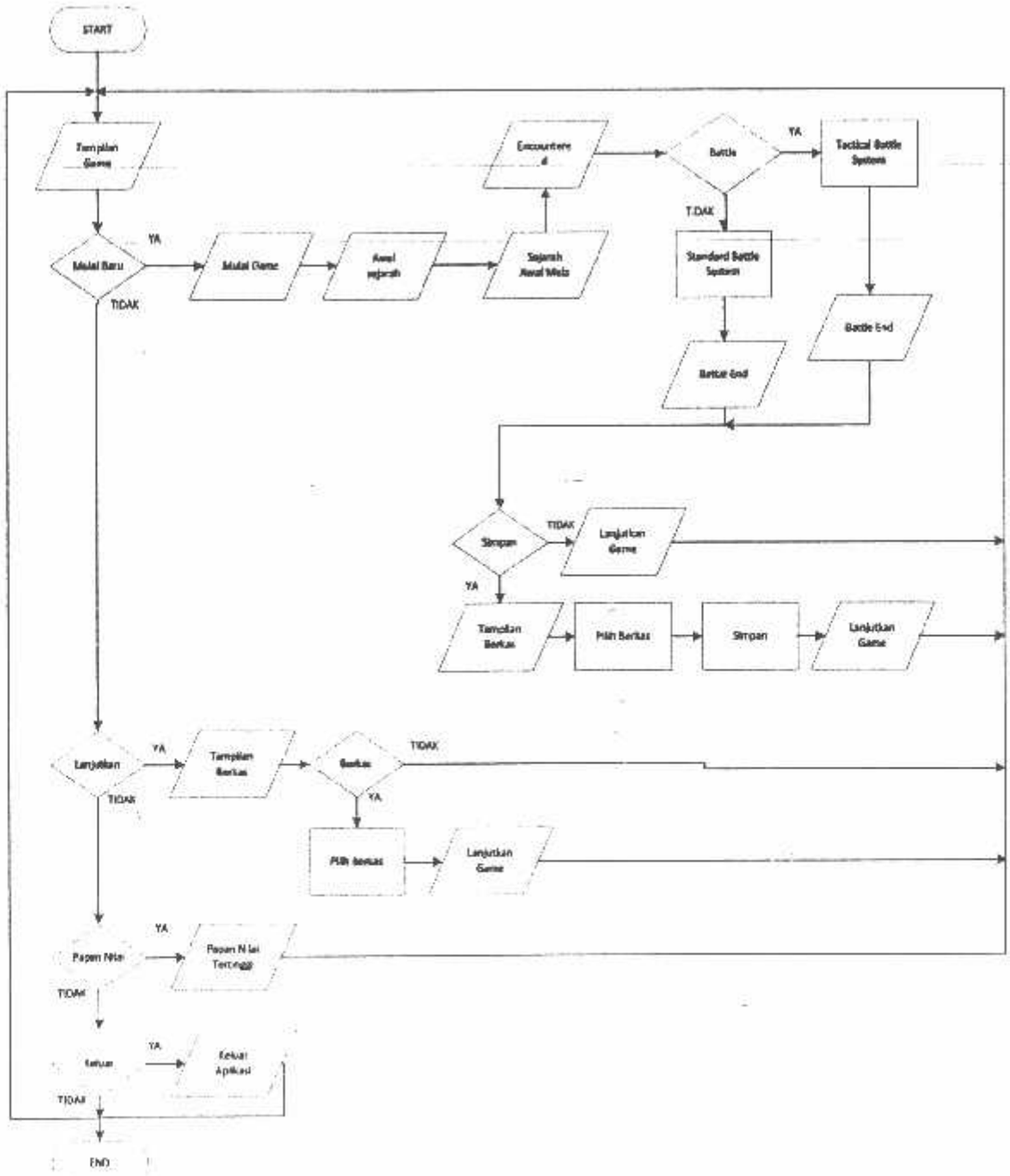
1. Proses *Opening* adalah proses pertama kali yang akan ditemui oleh *player* sebelum *player* dapat memainkan sebuah *game*. Pada tahap *Tampilan Awal Game* ini, *player* akan disuguhkan wallpaper dan *background* utama sebagai pendukung kelengkapan dalam *game* serta tombol navigasi yang memiliki beberapa menu fungsi.
2. Proses *Mulai Baru* adalah langkah awal seorang *player* untuk masuk kedalam *game* dan memainkannya.
3. Proses *Intro dalam game* adalah proses dimana *player* diberitahukan tentang jalan cerita awal dalam game, yang akan menjadi jalan cerita utama *player* untuk dapat meyelesaikan game ini.
4. Proses *Battle System* adalah proses dimana *player* akan menemui setiap musuh dalam perjalanannya dan kemudian akan masuk dalam proses pertarungan.
5. Pada proses pertarungan antar karakter, *sprite* karakter (representasi grafik 2 Dimensi karakter) mengalami proses inisialisasi oleh sistem. Kemudian *sprite* karakter dapat kita posisikan sendiri pada area karakter *player* yang terdapat pada *engine game*. Setelah karakter kita posisikan, aksi gerak yang diterapkan pada karakter dibantu dengan media *8D Battler*. *8D Battler* membantu karakter dapat melakukan aksi yang tidak terlihat monoton. Aksi yang dilakukan karakter dapat bergerak bebas dan terjadi *rolling* (perputaran).
6. Proses *Battle Karakter* adalah proses dimana tampilan dari penerapan metode *Tactical Battle* ditunjukkan dalam sebuah pertarungan.
7. Proses *Hasil Battle* adalah proses dimana *player* telah melalui proses *Tactical Battle* dan memenangkan pertarungan yang tengah dihadapi.

3.2. Perancangan Flowchart

Perancangan suatu aplikasi harus berdasarkan pada perancangan flowchart yang akan menjadi acuan agar suatu aplikasi dapat dibuat dengan konsep yang sudah ada, agar perancangan sistem yang dibuat berhasil dan berjalan sesuai dengan keinginan yang ada pada sistem yang sudah dirancang

3.2.1. Flowchart Aplikasi

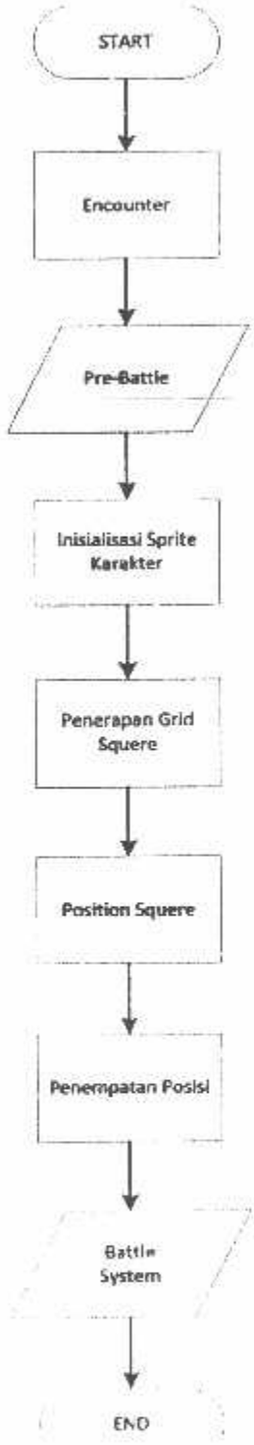
Flowchart perancangan *Game The Sword Master of Destiny* dengan Metode *Tactical Battle*, dimulai dari tampilan Home, tampilan menu-menu pendukung, proses *battle system*, hingga berakhirnya sebuah *game*, ditunjukkan pada gambar 3.2.



Gambar 3.2 Flowchart Perancangan *Game*

3.2.2. Flowchart Metode Tactical Battle

Pada tahap perancangan Metode *Tactical Battle* dalam *Battle Sistem game Sword Master of Destiny* ditunjukkan pada gambar 3.3.

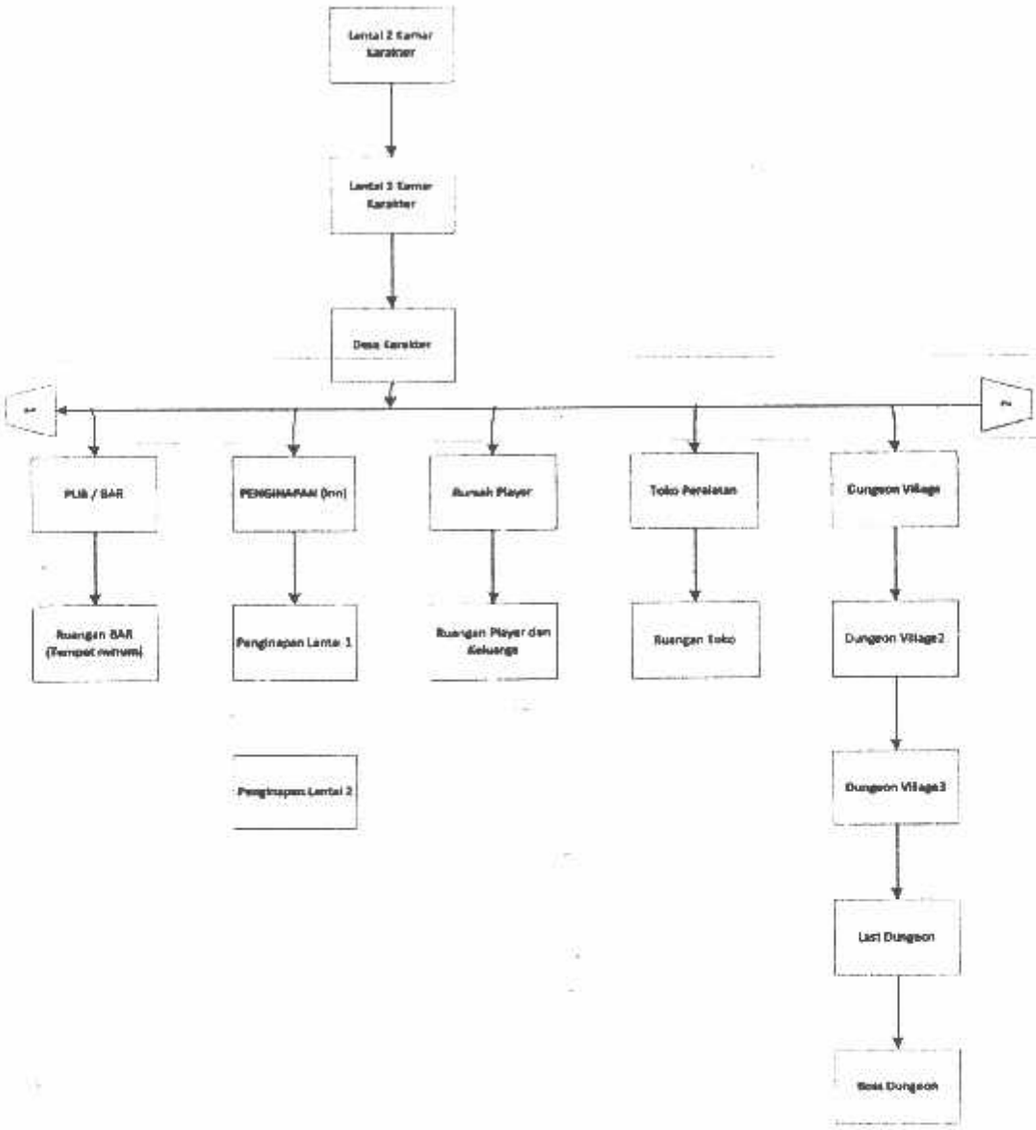


Gambar 3.3 Flowchart Metode *Tactical Battle*.

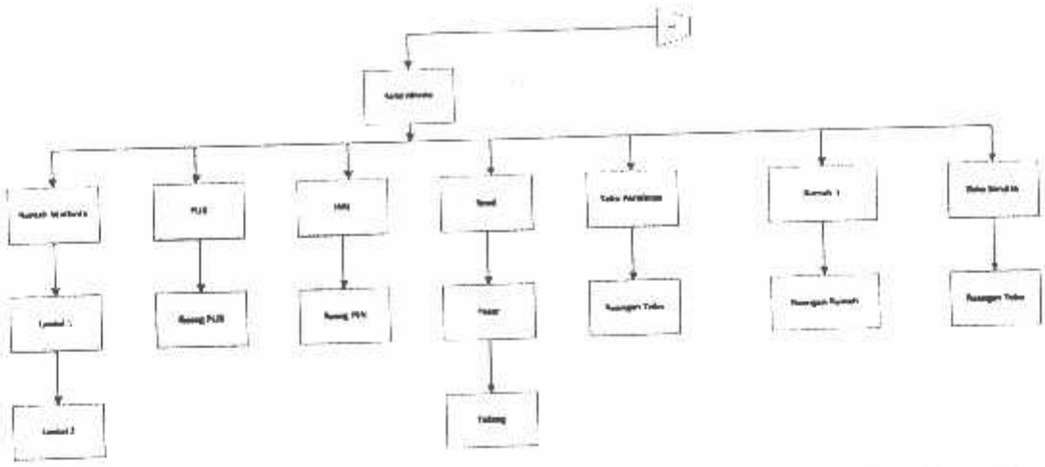
Pada flowchart metode *Tactical Battle* dalam gambar 3.3. dapat dijelaskan sebagai berikut:

1. Mulai pada saat *player* memainkan sebuah *game*.
2. *Player* akan menemui *Encounter*. *Encounter* adalah serangan dari musuh secara mendadak di wilayah tertentu dalam *Map game*, dan musuh yang melakukan *Encounter* dipilih secara acak dari beberapa musuh yang sudah ditentukan didalam *game*.
3. Saat pertarungan akan berlangsung, persiapan *screen* sebelum *battle system* ditampilkan.
4. Sebelum masuk dalam *battle system*, *sprite* karakter (representasi grafik 2 Dimensi) karakter diinisialisasi oleh sistem. Proses inisialisasi adalah penentuan letak posisi karakter baik itu *player* maupun musuh.
5. Kemudian setelah *sprite* karakter diinisialisasi, langkah berikutnya adalah *Grid squere* . Grid squere adalah tempat yang disediakan untuk penempatan karakter yang akan melakukan pertarungan.
6. Setelah penerapan proses *Grid Squere*, langkah berikutnya adalah menempatkan karakter pada *Grid squere*. Penempatannya di lakukan secara manual.
7. Masuk dalam *Battle Screen*.
8. Proses *Battle Screen* dengan *Tactical Battle*.
9. Selesai.

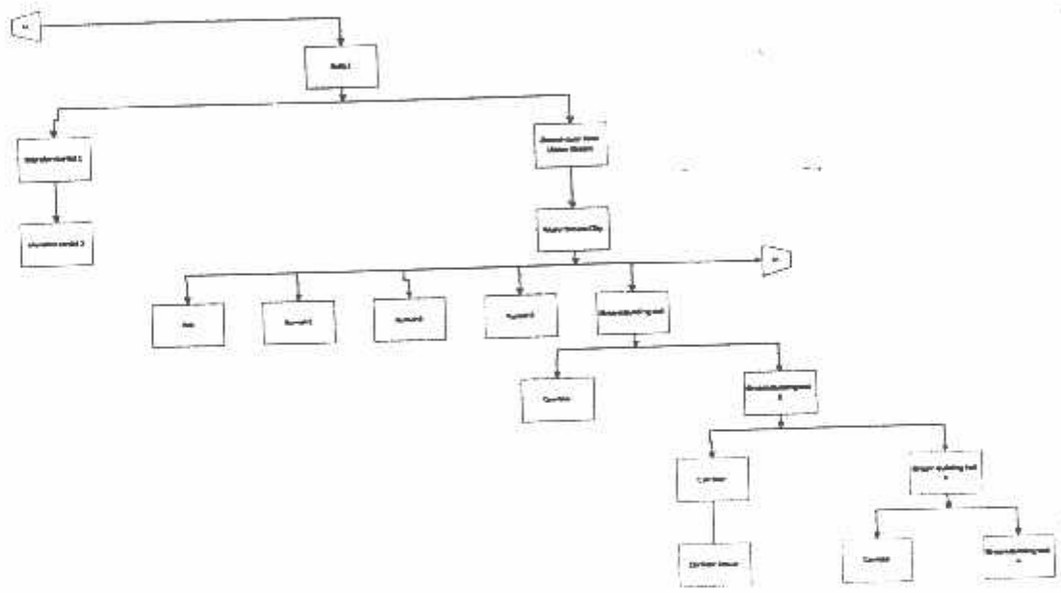
3.2.3. Flowchart Map Tree



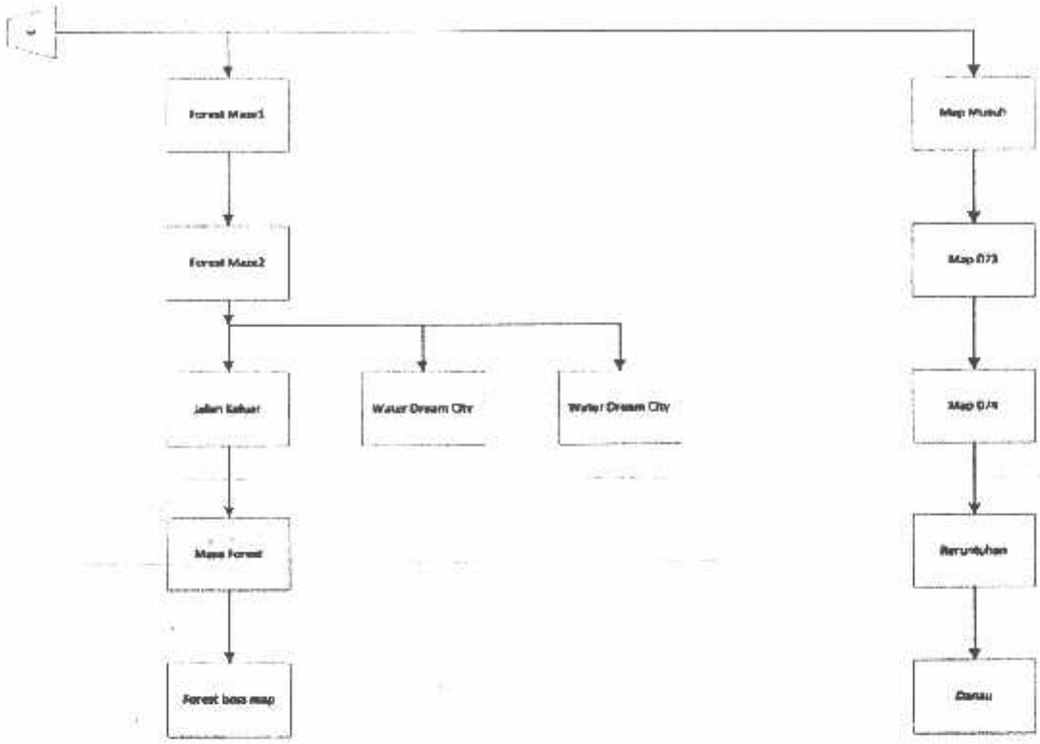
Gambar 3.4 Flowchart Map Tree Level Stage 1.



Gambar 3.4 Flowchart Map Tree Level Stage 2.



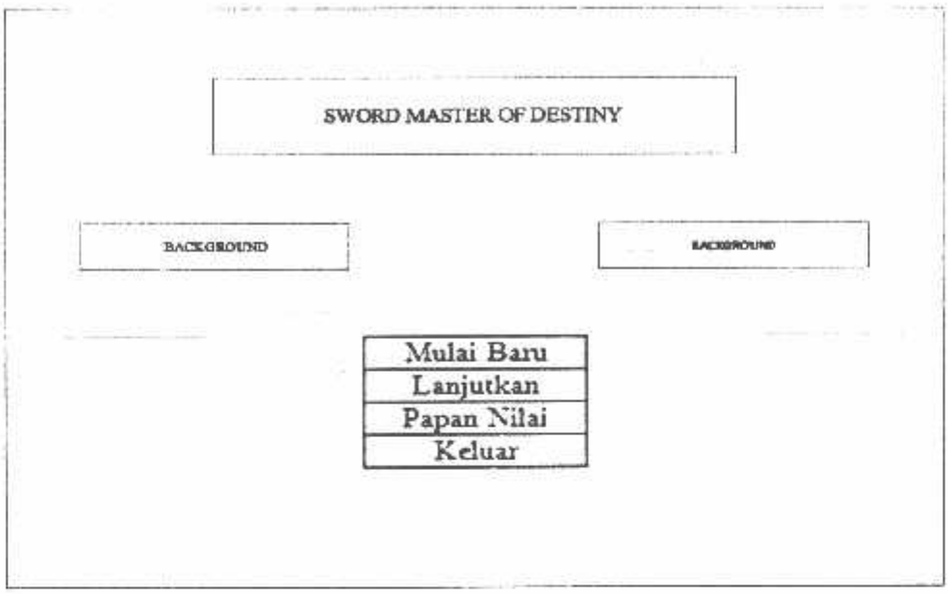
Gambar 3.5 Flowchart Map Tree Level Stage 3.



Gambar 3.6 Flowchart Map Tree Level Stage 4.

3.3. Perancangan Layout

Perancangan tampilan *game Sword Master of Destiny* dengan Metode *Tactical Battle* ditunjukkan pada gambar 3.7.



Gambar 3.7 Perancangan Layout Game.

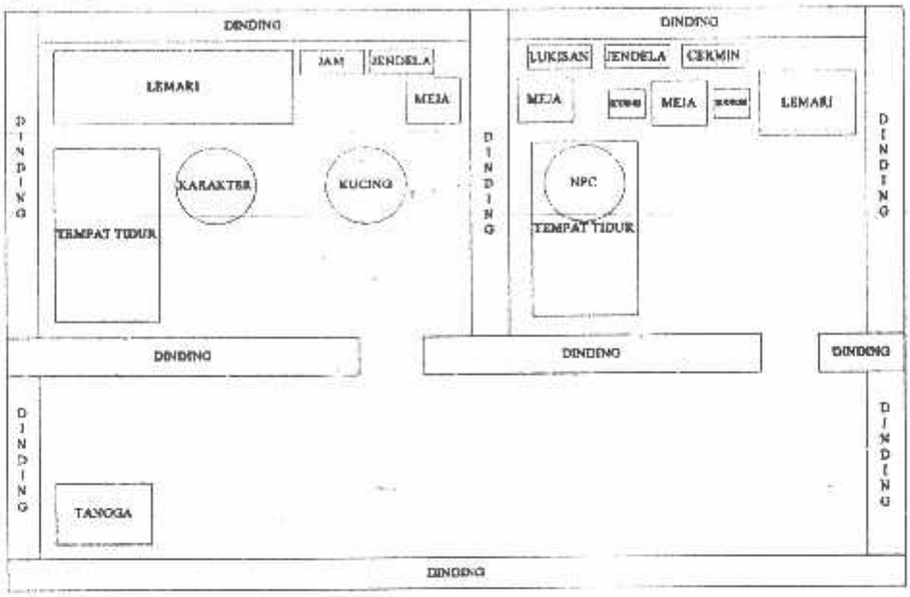
Pada perancangan tampilan game Sword Master of Destiny yang ditunjukkan dalam gambar 3.4 dapat dijelaskan sebagai berikut:

1. Kolom paling atas adalah nama dari sebuah game dengan judul *Sword Master of Destiny*.
2. Tampilan *background* (gambar latar) adalah pendukung menarik atau tidaknya game tersebut untuk dimainkan.
3. Menu *Mulai Baru* adalah pilihan dari menu yang pertama kali dipilih oleh *player* untuk memainkan game tersebut. Didalam menu *Mulai Baru*, *player* akan dihadapkan pada tampilan pemilihan karakter dan penjelasan tentang game sebelum memainkannya.
4. Menu *Lanjutkan* adalah pilihan dari menu yang berisi berkas data game yang sudah dimainkan sebelumnya oleh *player* yang telah disimpan. Jika *player* tidak menyimpan permainan, maka tidak akan ada berkas dalam file *Lanjutkan*, dan menu ini tidak akan dapat diakses

5. Menu *Keluar* adalah pilihan dari menu yang berfungsi untuk menutup jalannya aplikasi.

3.3.1. Perancangan Menu Mulai Baru

Menu mulai baru berisi tampilan dimana *player* memilih menu ini pertama kali untuk memainkan game *Sword Master of Destiny*. Perancangan tampilan penempatan karakter awal ditunjukkan pada gambar 3.8.



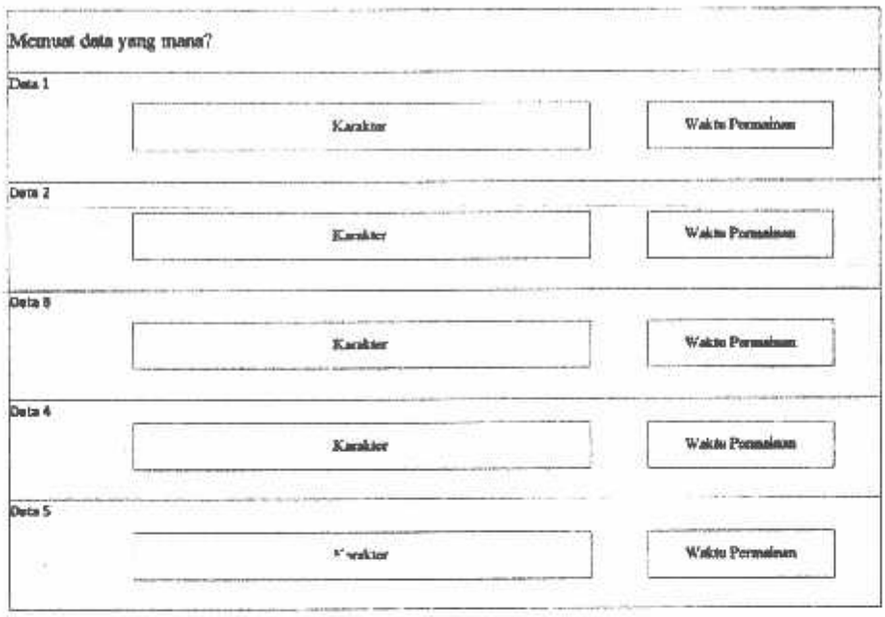
Gambar 3.8 Perancangan Penempatan Karakter Awal.

Pada perancangan Layout menu penempatan karakter awal yang ditunjukkan pada gambar 3.5 dapat dijelaskan sebagai berikut:

1. Setelah *player* memilih menu *Mulai Baru*, *player* akan masuk dalam tampilan *intro*(pengenalan awal mula game): Didalam tampilan ini, *player* diajak untuk melihat awal mula terjadinya konflik di dalam game.
2. Setelah *intro* (pengenalan awal mula game) player ditempatkan pada kamar karakter utama dari game.
3. *Player* juga dapat melihat petunjuk cara memainkan game *Sword Master of Destiny* di dalam rumah.

3.3.2. Perancangan Menu Lanjutkan

Menu *Lanjutkan* berisi tampilan dimana *berkas/data* simpanan game yang telah dimainkan *player* berada dan dapat diakses kembali untuk dimainkan. Perancangan tampilan menu *Lanjutkan* ditunjukkan pada gambar 3.9.



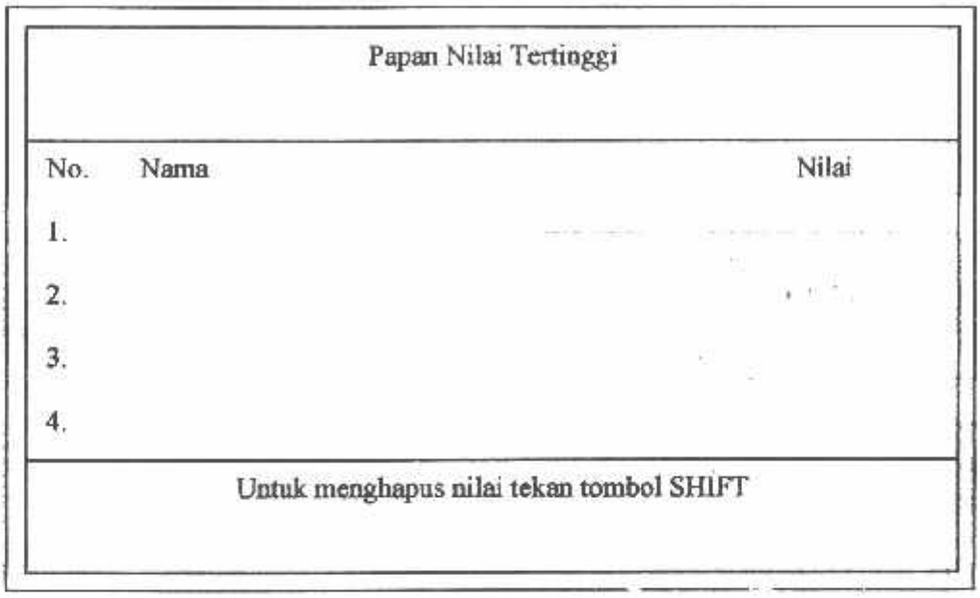
Gambar 3.9 Perancangan Layout Menu Lanjutkan.

Perancangan Layout menu *Lanjutkan* yang ditunjukkan pada gambar 3.6 dapat dijelaskan sebagai berikut.

1. Jika *player* menyimpan data yang sudah dimainkan sebelumnya, maka *player* akan diberi pilihan untuk menyimpan data tersebut pada *berkas* yang sudah tersedia.
2. Data yang sudah tersimpan, akan ditaruh didalam *berkas*. Jika *player* menginginkan datanya kembali untuk dimainkan, *player* tinggal mengakses menu *Lanjutkan* dan memilih *berkas* tempat *player* menyimpan data tersebut.
3. Jumlah maksimal *berkas* yang dapat dipakai adalah *14 berkas*. Maksud dari hal ini adalah *player* dapat menyimpan data sebanyak 14 kali data simpanan permainan yang berbeda.
4. *Player* juga diperbolehkan menumpuk *berkas* baru kedalam *berkas* lama yang sebelumnya pernah dimainkan

3.3.3. Perancangan Menu Score (Papan Nilai)

Menu *Papan Nilai* berisi tampilan dimana *nilai* yang didapat dari permainan setelah mengalahkan Bos Musuh ditampilkan. Perancangan tampilan menu *Lanjutkan* ditunjukkan pada gambar 3.10.



Gambar 3.10 Perancangan Layout Menu Papan Nilai.

Perancangan Layout menu *Papan Nilai* yang ditunjukkan pada gambar 3.10 dapat dijelaskan sebagai berikut:

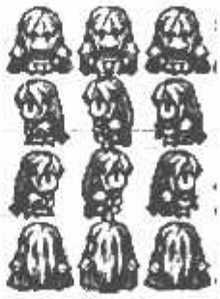
1. Menu *Papan Nilai* menampilkan nilai (*skor*) yang sudah dimainkan oleh *player* setelah mengalahkan tiap-tiap bos musuh sebelumnya.
2. Jika berhasil mengalahkan bos musuh, nilai akan keluar dan *player* diminta menyebutkan nama.
3. Hasil nilai akan ditampilkan pada menu *Papan Nilai*.
4. *Player* juga diperbolehkan untuk menghapus nilai (*skor*) dengan cara menekan tombol *SHIFT* pada *keyboard* komputer.

3.3.4. Perancangan Desain *Sprite* Karakter

Sprite adalah representasi grafik *2 Dimensi* dari karakter yang nantinya akan dimainkan oleh *player*. *Sprite* bisa disebut sebagai objek. Pada *game The Sword Master of Destiny*, terdapat 3 Karakter Utama yang dapat dipilih oleh *player*. Desain *Sprite* Karakter Utama ditunjukkan pada gambar 3.11 hingga gambar 3.13.

Gambar 3.11 *Sprite* Eric

Gambar 3.12 *Sprite* Natalie



Gambar 3.13 *Sprite* Terrence

Selain Karakter Utama didalam *game*, Karakter Pendamping juga sangat membantu *player* nantinya. Karakter Pendamping bertugas membimbing *player* dalam hal memberikan suatu informasi dan juga membantu *player* dalam pertarungan. Desain *Sprite* Karakter Pendamping ditunjukkan pada gambar 3.14 dan gambar 3.15.

Gambar 3.14 Sprite Rioma

Gambar 3.15 Sprite Ernest

3.4. Perancangan Sistem

Perancangan Sistem adalah perancangan kelengkapan keseluruhan dari fungsi *game The Sword Master of Destiny* yang mendukung berjalan baik atau tidaknya *game* ini nantinya. Perancangan Sistem terdiri dari Perancangan *Class* (Jenis Kelas / golongan pada Karakter), *Skills* (Kemampuan *Player* dan musuh), *Items* (Benda), *Enemies* (Musuh), dan *Formula* (Model penghitungan nilai serangan). Masing-masing sistem berhubungan erat dengan *Player* dan musuh.

3.4.1. Perancangan *Class*

Perancangan *Class* adalah perancangan golongan yang ditujukan hanya untuk karakter yang nantinya akan dimainkan *player* dan terdapat didalam *game The Sword Master of Destiny*. Pemberian *Class* dimaksudkan untuk membedakan kemampuan masing-masing karakter. Perbedaan kemampuan tersebut terdiri dari penggunaan *Skill* dan *Parameter Curve* yang diterapkan. Perancangan *Class* dari karakter ditunjukkan pada Tabel 3.1.

Tabel 3.1 *Tabel Perancangan Class Karakter*

CLASS	PARAMETER CURVE								SKILL	
	MHP	MMP	ATK	DEF	MAT	MDF	AGI	LUK	LEVEL	NAMA SKILL
ARCHER	423	78	16	13	10	12	34	33	10	ARMOUR BREAK
									15	ZERO SHADOW
									20	TRIPLE SHOT
									25	THOUSAND ARROW
MONK	662	46	22	14	11	12	35	15	10	CHAKRA
									15	TIGER STANCE
									20	NO SHADOW KICK
									25	CLAW DANCE
PALADIN	645	95	19	21	12	16	19	29	10	PROVOKE
									15	SUPER GUARD
									20	RESTRENGTHEN
									25	ZERO STORM
PRIESTESS	327	104	14	15	19	19	32	27	10	HEAL'S II
									12	FAIRY'S BREATH
									15	SAINT
									22	RECOVERY
SAGE	429	131	12	13	32	20	38	27	6	MAGIC PULSE
									10	SILENCE
									17	RECOVERY
									24	MYSTIC SPELL

CLASS	PARAMETER CURVE								SKILL	
	MHP	MMP	ATK	DEF	MAT	MDF	AGI	LUK	LEVEL	NAMA SKILL
SAMURAI	523	77	33	17	13	10	25	20	10	SHINTOU MEKKYAKU
									15	HASSOU
									20	TSUBAMEKAESHI
									25	OUKA MUGEN JIN
SOLDIER	562	41	30	16	10	10	20	15	10	CLEAVE
									15	BERSERK'S ROAR
									20	BERSERK'S DANCE
									25	GIANT'S RAMPAGE
SPELLBLADE	556	96	16	7	17	15	25	27	10	DISPEL
									15	MAGIC BARRIER
									20	REINFORCE
									25	RADIANCE BLADE
THIEF	467	66	17	14	11	11	39	30	10	FLARE
									15	THIEF'S LUCK
									20	ASSASIN'S EDGE
									25	VALIANT EDGE
WITCH	297	110	10	11	19	20	22	25	8	MAGIC REFLECT
									12	ENHANCE SPELL
									14	BLIZZARD
									24	MAGIC TORRENT

Pada Tabel 3.1 ditunjukkan bermacam-macam *Class* yang menjadi pembeda antar karakter. Terdapat 10 macam *Class* yang dapat diterapkan pada masing-masing karakter. Pada Tabel 3.1, terdapat 3 Hal utama yang harus diperhatikan yaitu *Class, Parameter Curve*, dan *Skill*. *Class, Parameter Curve*, dan *Skill* dapat dijelaskan sebagai berikut:

1. ***Class*** : adalah nama golongan yang terdapat didalam *Game*. *Class* terdiri dari 10 pembagian yang terdiri dari *Class Archer* (Pemanah), *Class Monk* (Rahib), *Class Paladin* (Ksatria Kerajaan), *Class Priestess* (Pendeta Wanita), *Class Sage* (Guru Bijak), *Class Samurai* (Pendekar Pedang), *Class Soldier* (Prajurit), *Class SpellBlade* (Ahli Mantra Pedang), *Class Thief* (Pencuri), dan *Class Witch* (Penyihir).

2. ***Parameter Curve*** : perbedaan yang paling terlihat dari tiap-tiap karakter adalah dari nilai *Parameter Curve*. Masing-masing *Class* memiliki nilai *Parameter Curve* yang berbeda. *Parameter Curve* adalah atribut yang berisi keterangan tentang status suatu karakter. *Parameter*

Curve terdiri dari *MHP* (*Maximum Hit Point* / Nyawa karakter), *MMP* (*Maximum Magic Point* / Kemampuan Sihir), *ATK* (*Attack Power* / Besar Serangan Fisik), *DEF* (*Defense Power* / Kemampuan Bertahan), *MAT* (*Magic Attack Power* / Besar Serangan Sihir), *MDF* (*Magic Defense Power* / Besar Kemampuan Bertahan terhadap Sihir), *AGI* (*Agility* / Kecepatan), *LUK* (*Luck* / Keuntungan). Nilai *Parameter Curve* ditetapkan mulai *level* paling dasar dari Karakter, yaitu *Level* 1.

Nilai *Parameter Curve* dapat diganti sesuai dengan kebutuhan, sebagai contoh terdapat 2 *Class* yang berbeda, yaitu *Class Witch* dan *Class Soldier*. *Class Witch* adalah sosok karakter yang identik dengan kekuatan sihir, maka nilai *MAT* (*Magic Attack Power* / Besar Serangan Sihir) dari *Class Witch* ditetapkan 2 / 3 kali nilai *MAT* dari *Class Soldier*. Sedangkan untuk nilai *ATK* (*Attack Power* / Besar Serangan Fisik) milik *Class Soldier* ditetapkan 2 / 3 kali nilai *ATK* dari *Class Witch*. Hal tersebut dikarenakan *Class Soldier* tidak memiliki kemampuan serangan sihir dan hanya memiliki kemampuan serangan fisik, maka nilai *MAT Class Soldier* lebih kecil.

3. Skill : perbedaan kedua yang mencolok adalah dari *Skill* (Kemampuan) yang dimiliki tiap-tiap *Class*. *Skill* yang dimiliki tiap-tiap karakter tidaklah sama. Masing-masing karakter dibatasi dengan *Class* dan perubahan *Level* ke tahap selanjutnya. *Level* pada Kolom *Skill* yang terdapat pada Tabel 3.1 mempunyai maksud bahwa pada *level* yang sudah ditentukan tersebut, suatu karakter akan mendapatkan *Skill* baru yang dapat digunakan segera. *Standart* umum *game RPG (Role Playing Game)* memiliki aturan bahwa setiap kenaikan 5 *level* yang dialami oleh *player*, *player* tersebut akan mendapatkan *Skill* baru. Berdasarkan standart tersebut, *game The Sword Master of Destiny* menetapkan suatu karakter akan memiliki *Skill* baru setiap kali karakter tersebut naik 5 *level*, yaitu pada *level* 10, 15, 20, dan *level* 25. Selain kenaikan 5 *level*, *game The Sword Master of Destiny* juga menambahkan beberapa *Skill* baru pada *level* tertentu seperti yang terdapat pada *Class Sage* dan *Witch* agar *game* tidak berkesan monoton.

3.4.2. Perancangan *Skills*

Skills didalam *game RPG (Role Playing Game)* adalah sebuah kemampuan yang dimiliki oleh karakter untuk bertarung dan berinteraksi terhadap musuh yang dihadapi. *Skill* terdiri dari berbagai macam bentuk, baik itu *Skill* yang dikhususkan untuk kekuatan serangan fisik, *Skill* untuk menghindari serangan, *Skill* untuk menyembuhkan diri dari luka, maupun *Skill* yang dikhususkan untuk serangan kekuatan sihir. Perancangan *Skills* karakter ditunjukkan pada Tabel 3.2.

Tabel 3.2 *Tabel Perancangan Skills Karakter*

Skills (Kemampuan)							
No	Nama	Skill Type	Hit Type	No	Nama	Skill Type	Hit Type
1	Attack	None	Physical Attack	24	Heal	Magic	Certain Hit
2	Guard	None	Certain Hit	25	Heal II	Magic	Certain Hit
3	Dual Attack	None	Physical Attack	26	Heal III	Magic	Certain Hit
4	Triple Attack	None	Physical Attack	27	Recovery	Magic	Certain Hit
5	Escape	None	Certain Hit	28	Recovery II	Magic	Certain Hit
6	Wait	None	Certain Hit	29	Cure	Magic	Certain Hit
7	Ice Breath	Special	Certain Hit	30	Cure II	Magic	Certain Hit
8	Fire Breath	Special	Certain Hit	31	Raise	Magic	Certain Hit
9	Shock	Special	Certain Hit	32	Raise II	Magic	Certain Hit
10	Poison Cloud	Special	Certain Hit	33	Poison	Magic	Magical Attack
11	Forget Cloud	Special	Certain Hit	34	Blind	Magic	Magical Attack
12	Dark Cloud	Special	Certain Hit	35	Silence	Magic	Magical Attack
13	Sonic Wave	Special	Certain Hit	36	Confuse	Magic	Magical Attack
14	Confusion Song	Special	Certain Hit	37	Paralyze	Magic	Magical Attack
15	Sleep Breath	Special	Certain Hit	38	Breath Weapon	Magic	Certain Hit
16	Sleep Pollen	Special	Certain Hit	39	Breath Armour	Magic	Certain Hit
17	Paralyze Breath	Special	Certain Hit	40	Divine Breath	Magic	Certain Hit
18	Paralyze Stare	Special	Certain Hit	41	Quick Move	Magic	Certain Hit
19	Sweep Kick	Special	Physical Attack	42	Curse Weapon	Magic	Magical Attack
20	Body Slam	Special	Physical Attack	43	Curse Armour	Magic	Magical Attack
21	Shout	Special	Certain Hit	44	Slow Move	Magic	Magical Attack
22	Vampire	Special	Physical Attack	45	Life Drain	Magic	Magical Attack
23	Meditate	Special	Certain Hit	46	Mana Drain	Magic	Magical Attack

Skills (Kemampuan)							
No	Nama	Skill Type	Hit Type	No	Nama	Skill Type	Hit Type
47	Fire	Magic	Magical Attack	70	Nuclear	Magic	Magical Attack
48	Fire II	Magic	Magical Attack	71	Fire Weapon	Special	Certain Hit
49	Flame	Magic	Magical Attack	72	Ice Weapon	Special	Certain Hit
50	Flame II	Magic	Magical Attack	73	Thunder Weapon	Special	Certain Hit
51	Ice	Magic	Magical Attack	74	Strong Attack	Special	Certain Hit
52	Ice II	Magic	Magical Attack	75	Cleave	Special	Physical Attack
53	Blizzard	Magic	Magical Attack	76	Berserk's Roar	Special	Certain Hit
54	Blizzard II	Magic	Magical Attack	77	Berserk's Dance	Special	Physical Attack
55	Thunder	Magic	Magical Attack	78	Giant's Rampage	Special	Physical Attack
56	Thunder II	Magic	Magical Attack	79	Tackle	Special	Physical Attack
57	Spark	Magic	Magical Attack	80	Chakra	Special	Certain Hit
58	Spark II	Magic	Magical Attack	81	Tiger Stance	Special	Certain Hit
59	Water	Magic	Magical Attack	82	No Shadow Kick	Special	Physical Attack
60	Wave	Magic	Magical Attack	83	Claw Dance	Special	Physical Attack
61	Stone	Magic	Magical Attack	84	Cover	Special	Certain Hit
62	Quake	Magic	Magical Attack	85	Provoke	Special	Certain Hit
63	Wind	Magic	Magical Attack	86	Super Guard	Special	Certain Hit
64	Tornado	Magic	Magical Attack	87	Restrengthen	Special	Certain Hit
65	Saint	Magic	Magical Attack	88	Zero Storm	Special	Physical Attack
66	Starlight	Magic	Magical Attack	89	Aura Blade	Special	Physical Attack
67	Shade	Magic	Magical Attack	90	Dispel	Special	Certain Hit
68	Darkness	Magic	Magical Attack	91	Magic Barrier	Special	Certain Hit
69	Burst	Magic	Magical Attack	92	Reinforce	Special	Certain Hit

Pada Tabel 3.2 ditunjukkan 2 hal utama yang menjadi faktor pendukung didalam skills, yaitu adanya *Skill Type* dan *Hit Type*. *Skill Type* adalah kekuatan karakter yang berasal dari jenis sumbernya, sedangkan *Hit Type* adalah bentuk keluaran kekuatan dari karakter yang mengenai musuh. Penjelasan mengenai *Skill Type* dan *Hit Type* adalah sebagai berikut:

1. *Skill Type* : *Skill Type* adalah sumber kekuatan yang dimiliki oleh karakter. Terdapat 3 sumber yang menjadi kekuatan karakter, yaitu "*None*", "*Special*", dan "*Magic*". 3 sumber kekuatan karakter dapat dijelaskan sebagai berikut:
   a. *None* : adalah sumber kekuatan karakter yang dapat dipakai kapanpun tanpa mengurangi suatu nilai, nilai yang dimaksud dalam game ini adalah nilai *TP (Technical Point)* dan

nilai *MP (Magical Point)*. *None* bisa disebut juga sebagai sumber kekuatan tingkat dasar dari karakter.

b. *Special* : adalah sumber kekuatan karakter yang berasal dari stamina karakter. Dalam penggunaan *Skill* yang memiliki *Skill Type* jenis *Special*, selalu terdapat pengurangan nilai *TP (Technical Point)* sebesar jumlah yang sudah ditetapkan pada suatu *Skill* tersebut. Masing-masing karakter dibekali dengan nilai *TP (Technical Point)*. Nilai *TP (Technical Point)* disebut juga sebagai stamina dari karakter, dan nilai *TP (Technical Point)* didapatkan dari sejauh mana karakter tersebut berjalan didalam *field game* (area permainan / peta didalam *game*).

c. *Magic* : sesuai namanya, *Magic* adalah sumber kekuatan yang berasal dari kemampuan sihir yang dimiliki oleh karakter. Terdapat pengurangan nilai *MP (Magical Point)* dalam setiap penggunaan *Skill* yang memiliki *Type Magic*. Tidak semua karakter mempunyai *Skill Type Magic*, dikarenakan *Skill Type Magic* hanya dapat digunakan oleh karakter yang memiliki nilai *MMP (Maximum Magic Point)* tinggi didalam *Class*-nya. Karakter yang mampu menggunakan *Skill Type Magic* secara maksimal adalah karakter *Class Priestess, Sage, Thief, dan Witch*.

2. *Hit Type* : *Hit Type* adalah bentuk keluaran kekuatan karakter yang mengenai musuh. Terdapat 3 bentuk keluaran kekuatan yang terdapat pada karakter, yaitu "*Certain Hit*", "*Physical Attack*", dan "*Magical Attack*". Bentuk keluaran kekuatan karakter dapat dijelaskan sebagai berikut:

   a. *Certain Hit* : adalah bentuk keluaran kekuatan dari karakter yang dapat mempengaruhi musuh jika *Skill* yang dipakai tersebut memiliki efek yang sangat merugikan. Contohnya adalah ketika karakter menggunakan *Skill Ice Breath*, *Ice Breath* memiliki efek *Frozen* yang dapat membekukan musuh. Jika terkena *Skill* ini, musuh akan

membeku dan tidak dapat bergerak, dan hal ini menjadi kesempatan bagi *player* untuk menyerang secara terus menerus. Selain merugikan bagi musuh, *Hit Type* jenis *Certain* juga membantu karakter dalam keadaan terdesak ketika menghadapi musuh. Contohnya adalah ketika *player* terdesak dan menghadapi musuh tangguh, *player* dapat menggunakan *Skill Escape* untuk melarikan diri dari pertempuran.

b. *Physical Attack* : adalah bentuk keluaran kekuatan dari karakter berupa serangan fisik. *Physical Attack* disebut juga sebagai serangan langsung pada tubuh fisik lawannya.

c. *Magical Attack* : adalah bentuk keluaran kekuatan dari karakter berupa serangan sihir. Musuh akan terkena dampak dari kekuatan sihir yang dilepaskan oleh *player*.

Total *Skills* yang dapat diterapkan pada karakter adalah 111 *Skills*. Terdapat juga *Skills* tambahan yang dapat dipakai oleh karakter dan sangat special dikarenakan efek, *Skill Type*, *Hit Type*, dan nilai kerusakan yang diterapkan sangatlah besar dan berbeda. *Skills* tambahan ditunjukkan pada Tabel 3.3.

Tabel 3.3 *Tabel Perancangan Skills tambahan*

Skills (Kemampuan)							
No	Nama	Skill Type	Hit Type	No	Nama	Skill Type	Hit Type
84	Radiant Blade	Special	Physical Attack	98	Valiant Edge	Special	Physical Attack
85	Yoroidoshi	Special	Physical Attack	99	Spirit Blessing	Special	Certain Hit
86	Shintou Meikyoku	Special	Certain Hit	100	Fairy's Breath	Special	Certain Hit
87	Tsubamekaeshi	Special	Physical Attack	101	God's Will	Special	Certain Hit
88	Ouka Mujen Jin	Special	Physical Attack	102	Goddess Embrace	Special	Certain Hit
89	Weapon Break	Special	Certain Hit	103	Magic Reflect	Special	Certain Hit
90	Armour Break	Special	Certain Hit	104	Enhance Spell	Special	Certain Hit
91	Zero Shadow	Special	Certain Hit	105	Nightmare	Special	Certain Hit
92	Triple Shot	Special	Physical Attack	106	Magic Torrent	Special	Certain Hit
93	Thousands Arrow's	Special	Physical Attack	107	Magic Pulse	Special	Certain Hit
94	Vanish	Special	Certain Hit	108	Spell Resistance	Special	Certain Hit
95	Flare	Special	Certain Hit	109	Mana Barrier	Special	Certain Hit
96	Thief's Luck	Special	Certain Hit	110	Mystic Spell	Special	Certain Hit
97	Assassin's Edge	Special	Physical Attack	111	Hassou	Special	Certain Hit

3.4.3. Perancangan *Items*

Items biasa disebut dengan benda, perlengkapan, atau barang. *Items* sangat dibutuhkan didalam *game*, terutama *game The Sword Master of Destiny*. *Items* memiliki kegunaan yang bermacam-macam sesuai dengan fungsi dan nama yang diberikan pada *items* tersebut. Selain dimiliki oleh *player*, *items* juga dimiliki oleh musuh.

Didalam *game The Sword Master of Destiny*, musuh dibekali dengan *items*. Hal ini bertujuan setiap kali *player* mengalahkan musuh, *player* akan mendapat *reward* (hadiah) berupa *items* dari musuh yang telah dikalahkannya tanpa susah-susah membeli *items* di Toko Peralatan. Walaupun dibekali dengan *items*, musuh tidak dapat menggunakan *items* yang dimilikinya. *Items* hanya dapat dipakai oleh *player* saja. Perancangan *Items* ditunjukkan pada Tabel 3.4.

Tabel 3.4 *Tabel Perancangan Items*

Items(Barang-barang)							
No	Item	Harga (G)	Deskripsi	No	Item	Harga (G)	Deskripsi
1	Potion	50	Mengembalikan 500 HP	17	Dial Ring	0	Mengembalikan seluruh HP dan MP
2	Hi-Potion	150	Mengembalikan 2.500 HP	18	Cherry Allusia	100	Mengembalikan 3000 HP
3	Full Potion	450	Mengembalikan HP hingga penuh	19	Tomato Redo	80	Recover 3000 HP dan MP
4	Magic Water	300	Mengembalikan 200 MP				
5	Stimulant	250	Mengembalikan player dari kematian				
6	Antidote	30	Menetralisir racun				
7	Diapel Herb	75	Menyembuhkan semua status minor				
8	Elixir	3000	Mengembalikan seluruh HP dan MP				
9	Life Up	200	Meningkatkan Max HP 50 Point				
10	Mana Up	200	Meningkatkan Max MP 50 Point				
11	Power Up	200	Meningkatkan ATK 3 Point				
12	Guard Up	200	Meningkatkan DEF 3 Point				
13	Magic Up	200	Meningkatkan MAT 3 Point				
14	Resist Up	200	Meningkatkan MDF 3 Point				
15	Speed Up	200	Meningkatkan AGI 3 Point				
16	Luck Up	200	Meningkatkan LUK 3 Point				

Pada Tabel 3.4, diperlihatkan 33 *Items* yang dapat dipakai oleh *player*. *Items* pada *game The Sword Master of Destiny* dihargai dengan mata uang *Gold (G)*. Pada Tabel 3.4, terdapat beberapa *items* yang dihargai dengan 0 *Gold (G)*. Maksud dari 0 *Gold (G)* adalah, *items* tersebut tidak dapat dibeli maupun dijual di Toko Peralatan. Didalam *game The Sword Master of Destiny*, *player* mendapat beberapa *items* yang tidak dijual di Toko Peralatan dari setiap petualangan yang dihadapinya.

3.4.4. Perancangan *Enemies*

Enemies adalah musuh yang akan dihadapi oleh *player* didalam pertarungan. *Player* akan menghadapi musuh ketika *player* tersebut berhasil memecahkan suatu masalah dan ketika akan menemukan titik akhir dari permasalahan yang sedang dihadapi. Didalam *game The Sword Master of Destiny*, *player* akan menghadapi 3 *Stage* (bagian) dimana di tiap-tiap *stage* (bagian), *player* akan menghadapi musuh besar dan Bos Musuh. Total terdapat 7 musuh yang akan dihadapi *player*. Tabel Perancangan *Enemies* ditunjukkan pada Tabel 3.5.

Tabel 3.5 *Tabel Perancangan Enemies*

Enemies	Parameter Curves								Reward		Skill
	MHP	MMP	ATK	DEF	MAT	MDF	AGI	LUK	EXP	Gold	
	750	40	35	25	25	25	25	20	250	750	- Attack - - Ice Breath - - Double Attack - - Ice -
	850	60	40	30	30	30	30	25	260	750	- Attack - - Thunder - - Double Attack - - Shock - - Paralyze -
	950	70	50	40	40	30	35	30	300	800	- Attack - - Flame - - Triple Attack -

Enemies	Parameter Curves								Reward		Skill
	MHP	MMP	ATK	DEF	MAT	MDF	AGI	LUK	EXP	Gold	
	1000	0	65	40	45	45	55	50	1500	4000	- Sweep Kick - - Body Slam - - Assassin Edge -
	5000	1000	100	70	100	70	70	70	4500	5000	- Double Attack - - Sonic Wave - - Flame - - Thunder Weapon - - Spark - - Heal -
	4700	1500	135	80	90	80	83	85	6000	10000	- Triple Attack - - Heal - - Quake - - Dark Cloud - - Strong Attack - - Stone - - Tsubamekaeshi -
	9000	3000	170	120	130	100	110	100	15000	20000	- Attack - - Fire Breath - - Flame II - - Nuclear - - Giant's Rampage - - Recovery II -

Pada Tabel 3.5, diperlihatkan 7 musuh besar yang nantinya akan dihadapi oleh *player*. Sesuai dengan urutannya, ada 3 *Stage* (bagian) musuh yang menanti *player*. *Stage* 1 adalah *stage* Keluarga Morgan (*Morgan Family Qatar, Morgan Family Amaru, Morgan Family Isnan*, dan Sang Pemimpin Henry Morgan). *Stage* 2 adalah *stage* Barbossa (yang terdiri dari Lilith Dunois sang asisten dan Barbossa sebagai pemimpinnya). *Stage* 3 adalah *stage* terakhir yang akan dihadapi oleh *player*, yaitu Samuel Bellamy (Bos Besar dan dalang dibalik semua kekacauan).

Enemies juga dibekali dengan *Parameter Curves* sama seperti karakter *player*, hal tersebut dikarenakan *Parameter Curves* berperan penting dalam sistem pertarungan antar karakter. Selain dibekali dengan *Parameter Curves*, *enemies* juga dibekali dengan *Reward* (hadiah). *Reward* (hadiah) ditujukan bagi *player* apabila *player* berhasil mengalahkan *enemies* yang telah dihadapinya. *Reward* bagi *player* terdiri dari 2 jenis, yaitu *EXP (Experience*

Point) dan uang dalam bentuk mata uang *Gold*. *EXP (Experience Point)* adalah nilai yang didapat oleh *player* untuk menambah kenaikan *Level*.

Hal terakhir yang diterapkan pada *Enemies* adalah pemberian *Skill*. *Skill* tidak hanya dimiliki oleh karakter *player* saja, tetapi *skill* juga diterapkan pada karakter *enemies* untuk pertarungan melawan *player*.

3.4.5. Perancangan *Formula*

Formula biasa disebut juga sebagai ramuan, tetapi di dalam *game RPG (Role Playing Game)*, *formula* adalah sistem penghitungan nilai *damage* (kerusakan) berupa *rumus* yang ditujukan untuk mengurangi nilai *HP* (*Hit Points* / nyawa) pada *player* maupun musuh. Pemasangan *formula* terletak pada masing-masing *Skill* yang terdapat didalam *RPG Maker VX Ace*.

Terdapat 2 unsur utama didalam *formula*, yaitu *Entitas* dan *Variable*. *Entitas* berisi keterangan mengenai karakter, sedangkan *variable* berisi mengenai *rumus* penghitungan *formula*. Keterangan mengenai perancangan *formula* dapat dilihat pada Tabel 3.6.

Tabel 3.6 *Tabel Perancangan Formula*

FORMULA	KETERANGAN
a	Entitas dari karakter Player / Enemies
b	Entitas dari karakter Player / Enemies
v	Variabel (Rumus Formula)
a.v	Rumus dari Entitas dan Variabel
b.v	Rumus dari Entitas dan Variabel
a.v – b.v	Rumus sederhana dalam penghitungan Formula
a.atk*4 – b.def*2	Penjabaran rumus penghitungan dari Entitas dan Variabel

Pada tabel 3.6 diperlihatkan 2 keterangan penting mengenai *formula*, yaitu adanya *Entitas* dan *Variable*. *Entitas* adalah keterangan karakter sebagai pembeda antara *Player* dengan *Enemies*. *Player* dan *Enemies* dibedakan dengan huruf "*a*" dan huruf "*b*". Sedangkan *variable* adalah isi *rumus* sederhana dari sebuah *formula*.

Rumus dasar yang diterapkan didalam *game RPG Maker VX Ace* adalah besar nilai serangan *(ATK / Attack)* dikurangi dengan besar nilai bertahan *(DEF / Defense)*. Hal tersebut dapat dijelaskan sebagai berikut:

1. Patokan dasar yang paling utama digunakan sebagai *formula* adalah *ATK* dan *DEF*. *ATK* adalah atribut dari *Parameter Curves* yang berisi besar nilai serangan, sedangkan *DEF* adalah atribut yang berisi besar nilai pertahanan yang dimiliki oleh *player* maupun musuh.
2. Besar nilai serangan yang digunakan nantinya memiliki nilai 2x lipat lebih besar dari nilai pertahanan. Sebagai contoh dari sebuah *formula* patokan dasanya adalah *rumus* berikut *a.atk*4 - b.def*2*.
3. Nilai serangan *(ATK)* dari karakter dikalikan dengan 4, kemudian dikurangi dengan nilai pertahanan *(DEF)* dikalikan dengan 2.
4. Nilai serangan *(ATK)* memiliki nilai 2x lipat lebih banyak daripada nilai pertahanan *(DEF)*, hal tersebut dilakukan untuk mendapatkan nilai hasil penghitungan *formula*. Nilai hasil penghitungan ini nantinya akan menjadi nilai *DAMAGE* (kerusakan) yang berfungsi untuk mengurangi nyawa *(HP / Hit Points)* dari karakter *player* maupun musuh.
5. *Formula* terdapat didalam *Skill*. Jika *Skill* diterapkan pada *player*, maka *entitas "a"* pada formula adalah entitas milik *player* dan *entitas "b"* adalah milik musuh. Begitu pula sebaliknya, jika skill diterapkan pada musuh, *entitas "a"* adalah milik musuh, dan *entitas "b"* adalah milik *player*.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi

Pada tahapan implementasi ini akan dilakukan proses penerapan Metode *Tactical Battle* pada proses *Battle System* antara *player* dengan musuh, dimana konsep dan penerapan *Tactical Battle* berdasarkan pada nilai pertarungan terdapat 3 bagian nilai, yaitu nilai *Grid Squere, Position Ability. Dan Turn System.* Metode *Tactical Battle* diterapkan pada saat *player* bertarung melawan musuh apapun. Didalam *game RPG The Treasure Hunter*, *player* akan menghadapi 2 jenis musuh, musuh biasa ketika *player* menghadapi makhluk-makhluk buas dan liar dan jenis musuh yang kedua adalah ketika *player* menghadapi 7 musuh besar. 7 musuh besar yang dimaksud didalam *game The Treasure Hunter* ini adalah para Bos yang melindungi senjata dan armor legenda yang pernah dipakai pahlawan yg telah mengalahkan bos monster.

Konsep *metode* ini akan berjalan pada saat *player* bertemu dengan musuh, saat sebelum musuh dan *player* bertemu di *Medan Field* (arena pertarungan), *sprite* dari kedua karakter dikenali oleh *sistem*. *Sprite* adalah representasi *grafik* dari karakter. Setelah *sistem* mengenali *sprite* dari proses *inisialisasi*. *Grid Squere* berfungsi sebagai tempat dimana untuk meletakkan posisi tiap-tiap karakter. Penempatan karakter *player* ditentukan oleh pemain. Selain penentuan posisi awal, juga berfungsi sebagai penentu posisi karakter saat melakukan serangan pada musuh. Setelah penentuan posisi dengan *Position Squere Ability* memiliki fungsi yang berbeda Grid Squere. *Position Squere Ability* berfungsi untuk meletakkan karakter pada setian posisi. *Turn* System berfungsi sebagai penghitungan waktu untuk melakukan serangan terlibih dahulu, barulah *player* dan musuh dapat bertemu dan bertarung di *Medan Field* (arena pertarungan) yang sudah ditentukan.

Ketiga konsep tersebut berfungsi sebagai media pertarungan. Dengan menggunakan *Grid Squere*, *Position Squere Ability*, dan *Turn* System sehingga *player* dan musuh bergerak berputar bervariasi dan bergantian.

4.2. Penentuan Karakter

Dalam *game RPG (Role Playing Game)*, sosok karakter sangat diperlukan untuk merepresentasikan seseorang pada alur cerita. Karakter pada sebuah *game RPG (Role Playing Game)* dibagi menjadi 2 kategori, yaitu *Playable Character (PC)* dan *Non Playable Character (NPC)*. *Playable Character (PC)* adalah sosok karakter yang dapat dimainkan dan digerakkan oleh *player* dan dapat melakukan interaksi atau kontak secara langsung terhadap sistem melalui sebuah aksi. Hal tersebut ditunjukkan dengan karakter yang dimainkan oleh *player* mampu berinteraksi dengan lingkungan disekitarnya, memahami pola dan sifat dari apa yang baru dikenalnya, menentukan sebuah pilihan yang akan diambil, dan berperan penting didalam sebuah pertarungan.

Sedangkan *NPC (Non Playable Character)* adalah karakter sekunder yang memiliki fungsi sebagai pelengkap alur cerita dan sebagai pendamping interaksi bagi *PC (Playable Character)* di dalam *game*. *NPC (Non Playable Character)* memiliki peran yang tidak kalah penting dengan *PC (Playable Character)*, diantaranya adalah sebagai pembimbing *player* yang dapat memberikan informasi berupa petunjuk didalam suatu *game*, dan tentang semua hal apa-apa saja yang harus dilakukan oleh *player* nantinya didalam *game*. *NPC (Non Playable Character)* tidak dapat dimainkan oleh *player*, tetapi *player* mampu berinteraksi penuh dengan *NPC (Non Playable Character)* didalam *game*. Tanpa adanya *NPC (Non Playable Character)*, sebuah *game* akan terasa hambar dan *game* tersebut tidak dapat berjalan dengan baik jika hal yang dituju adalah jenis permainan *RPG (Role Playing Game)*.

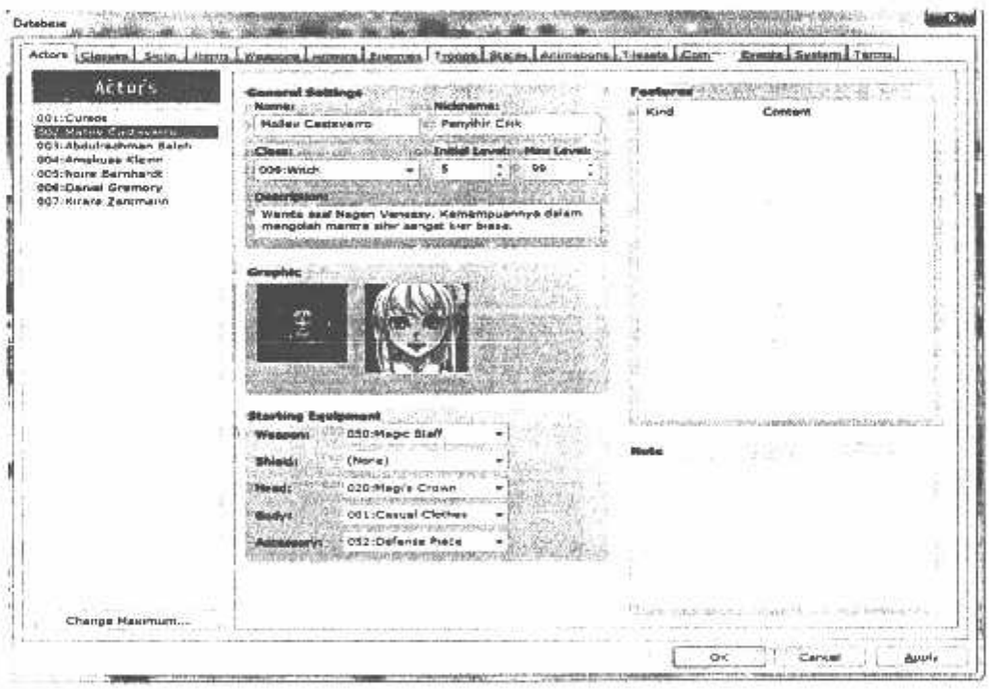
4.2.1. Menentukan karakter di dalam *RPG Maker VX Ace*

Sebelum memainkan sebuah *game RPG (Role Playing Game)*, langkah pertama yang harus diperhatikan adalah membuat dan menentukan karakter terlebih dahulu yang akan dimainkan. Karakter karakter tersebut nantinya akan dimainkan oleh *player* dan merepresentasikan sosok *player* didalam sebuah *game*. Karakter yang dibuat dapat ditentukan jenis *Class* nya. *Class* adalah jati diri yang ditanamkan pada karakter di dalam *RPG Maker VX Ace*. *Class* juga dapat disebut sebagai tingkatan kemampuan yang membedakan kemampuan masing-masing karakter. Selain *Class*, pemberian nama karakter

adalah hal utama yang dapat dilakukan. Langkah selanjutnya adalah pemberian *Level* pada karakter. *Level* disini dimaksudkan untuk memberikan sebuah nilai tingkatan *numerik* saat pertama kali karakter tersebut dimainkan. Pemberian *Level* dapat dimulai dari *Level* 1 sebagai start dan awal mula, atau dapat dimulai pada *level* 5.

Selain penentuan nama, *Class*, dan *level*, karakter yang telah dibuat tentunya harus dilengkapi dengan sistem persenjataan. Sistem persenjataan tersebut mendukung tiap-tiap karakter untuk beraksi didalam pertarungan yang akan terjadi nantinya. Senjata dan material yang terdapat pada karakter berbeda-beda, hal tersebut ditunjukkan pada pemberian *Class* yang telah dilakukan sebelumnya.

Karakter–karakter tersebut dapat dibuat dan diatur sedemikian rupa pada sebuah menu *Database* yang terdapat pada *RPG Maker VX Ace*. Menu *Database sistem* ditunjukkan pada gambar 4.1.



Gambar 4.1 Menu *Database* pada *RPG Maker VX Ace*.

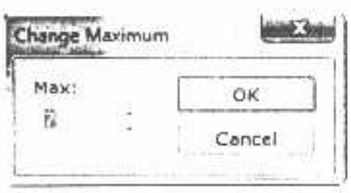
Pada menu *Database* yang ditunjukkan dalam gambar 4.1, penentuan karakter yang nantinya akan dimainkan oleh *player* dapat dilakukan pada kolom-kolom yang sudah tersedia pada sub menu *Actors* dari menu

Database. Pada sub menu *Actors* yang terletak pada kolom sebelah kiri, jumlah karakter yang ingin dibuat dan dimainkan dapat ditentukan melalui option *Change Maximum* yang terletak pada posisi paling bawah.

1. Kolom *Actors* pada sub menu *Actors*

Gambar 4.2 Kolom *Actors* pada sub menu *Actors RPG Maker VX Ace*.

Dalam gambar 4.2 ditunjukkan sebuah kolom yang berfungsi untuk menambah maupun mengurangi jumlah karakter yang nantinya akan dimainkan. Untuk melakukan proses *edit*/pergantian nama suatu karakter, caranya adalah dengan menyeleksi karakter yang sudah tersedia pada kolom dan kemudian menggantinya. Sedangkan untuk menambah maupun mengurangi jumlah karakter yang akan digunakan, digunakanlah tombol *Change Maximum* yang terletak paling bawah pada kolom *Actors*.

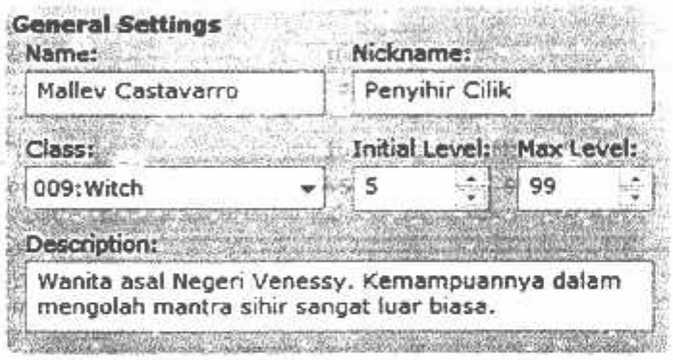


Gambar 4.3 Tombol *Change Maximum* pada *RPG Maker VX Ace*.

Jumlah karakter yang nantinya akan dipergunakan didalam *game* dapat ditentukan dengan cara mengganti nilai *Max* yang terdapat didalam tombol *Change Maximum*. Jumlah karakter dapat ditentukan sesuai dengan karakter didalam *game* yang akan dimainkan. Karakter yang akan dimainkan bisa meliputi Tokoh utama yang nantinya akan digerakkan oleh *player*, dan karakter lainnya adalah musuh / bos musuh.

2. Kolom *General Settings* pada sub menu *Actors*

Kolom *General Settings* pada sub menu *Actors* digunakan untuk menentukan identitas dari karakter yang nantinya akan dimainkan oleh *player* didalam *game*. Pemberian identitas seorang karakter harus lebih rinci dan jelas untuk menunjukkan status karakter tersebut didalam *game* nantinya. Kolom *General Settings* ditunjukkan pada gambar 4.4.



Gambar 4.4 Kolom *General Settings* pada *RPG Maker VX Ace*.

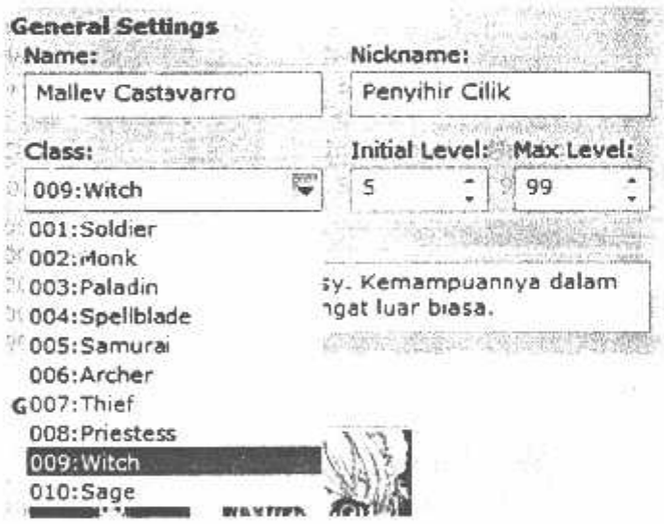
Pada gambar 4.4 diambil sebuah contoh karakter bernama Mallev Castavarro. Karakter ini berperan penting sebagai rekan pertama Tokoh utama yang bertugas membantu Tokoh utama didalam petualangan dan mendampingi Tokoh utama kemanapun dan melakukan hal apapun. Karakter Mallev Castavarro ditentukan sebagai karakter yang mampu mendukung Tokoh utama bertarung di dalam *Battle Sistem* saat melawan musuh. Penjelasan mengenai fungsi dari *General Settings* dapat dijabarkan sebagai berikut:

a. *Name* : fungsi *name* adalah sebagai pemberian sebuah identitas berupa nama terhadap karakter yang nantinya akan dimainkan dalam *game*. Sebuah karakter yang tidak memiliki identitas berupa

nama, tidak dapat digolongkan sebagai karakter pendamping tokoh utama. Karakter yang tidak memiliki nama bisa disebut juga sebagai *NPC (Non Playable Character)*. Tujuan pemberian nama terhadap karakter adalah agar sistem dapat mengenali dengan mudah suatu karakter yang terlibat didalam suatu *event game*.

b. *Nickname* : fungsi *nickname* adalah pemberian gelar atau julukan yang biasanya dimiliki oleh karakter-karakter dalam *game RPG (Role Playing Game)*.

c. *Class* : fungsi *class* adalah sebagai jati diri dari suatu karakter yang membedakan kemampuan tiap-tiap karakter. Kemampuan tiap-tiap karakter dibedakan menurut jati diri yang diberikan kepadanya, hal ini menyebabkan adanya karakter yang memiliki kemampuan tidak sama dengan karakter lain secara mayoritas. Didalam *RPG Maker VX Ace* terdapat 10 macam *Class* yang membedakan karakter lain, namun fungsi *Class* juga dapat ditambah maupun dikurangi sesuai dengan kebutuhan didalam *game*.



Gambar 4.5 Macam-macam *Class* pada *RPG Maker VX Ace*.

Dalam gambar 4.5 ditunjukkan macam-macam *Class* yang menjadi jati diri suatu karakter. Macam-macam *Class* tersebut menjadi pembeda dalam hal kemampuan bertempur yang dimiliki masing-masing karakter. Dalam contoh *game RPG (Role Playing Game) Sword Master Of Destiny*, Mallev Castavarro dibekali dengan *Class Witch*. *Witch* memiliki arti Penyihir.

d. *Initial level* : fungsi *initial level* adalah sebagai nilai *level* awal yang dimiliki oleh suatu karakter. Pada awal permainan sebuah *game*, tidak mungkin *player* langsung menemui karakter dengan *level* tinggi. Tentunya *player* akan menemui karakter dengan *level* rendah dan kemudian *player* dituntut untuk mengembangkan karakternya untuk mencapai *level* yang diinginkan. Pada permainan *RPG (Role Playing Game)* pada umumnya, *initial level* dimulai dengan *level* 1. Untuk contoh karakter Mallev, *initial level* diisi dengan angka 5 dikarenakan *player* akan bertemu dengan karakter Mallev pada pertengahan cerita *game*.

e. *Max level* : fungsi *max level* disini adalah sebagai nilai maksimal sebuah *level* yang dapat dicapai oleh suatu karakter. Dalam *game Sword Master Of Destiny*, nilai maksimal *level* yang dapat dicapai oleh suatu karakter adalah 99.

f. *Description* : fungsi dari *description* adalah menjelaskan dengan detail tentang deskripsi yang dimiliki suatu karakter. Deskripsi dapat berisikan tentang tempat karakter tersebut berasal, kemampuan yang dimiliki oleh suatu karakter, dan bahkan sifat yang dimiliki oleh karakter tersebut.

3. Kolom *Graphic* pada sub menu *Actors*

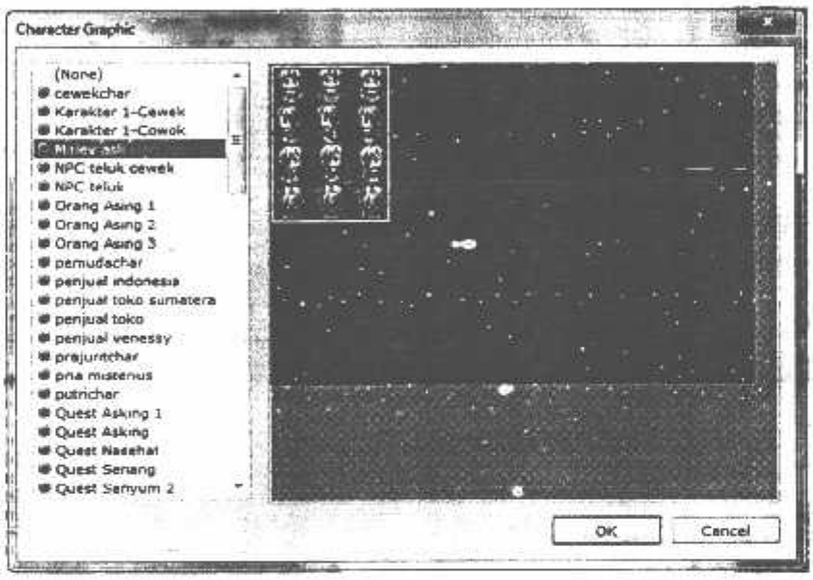
Kolom *Graphic* pada sub menu *Actors* digunakan untuk membuat karakter yang akan dimainkan. Pada kolom *Graphic* ada 2 hal yang harus diperhatikan, yaitu *Character Graphic* dan *Face Graphic*.

Gambar 4.6 Kolom *Graphic* pada *RPG Maker VX Ace*.

Dalam gambar 4.6 ditunjukkan kolom *Graphic* yang berfungsi sebagai pilihan untuk membuat suatu karakter yang akan dimainkan. Terdapat 2

tampilan dalam kolom *Graphic*, kotak sebelah kiri menunjukkan *Character Graphic* dan kotak sebelah kanan menunjukkan *Face Graphic* dari suatu karakter.

a. Character Graphic : *character graphic* adalah desain *2D* (2 Dimensi) dari suatu karakter yang didesain berupa *sprite*. *Sprite* adalah representasi grafik berupa *objek* dari suatu karakter yang didesain agar wujud dari karakter yang akan dimainkan dapat terlihat. Untuk memilih *sprite* karakter, langkah yang harus dilakukan adalah klik kotak disebelah kiri pada kolom *Graphic*. Kemudian akan muncul *window Character Graphic*, dan karakter berupa *sprite* dapat dipilih sesuka hati. Tampilan *window Character Graphic* ditunjukkan pada gambar 4.7.

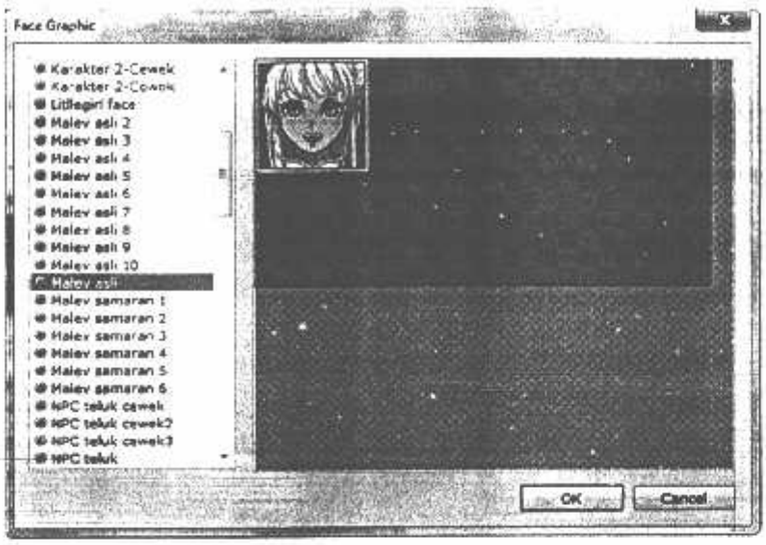


Gambar 4.7 *Window Character Graphic* pada *RPG Maker VX Ace*.

Pada gambar 4.7 ditunjukkan macam-macam *sprite* dari karakter yang dapat digunakan. *Sprite* yang diseleksi pada gambar 4.7 adalah *sprite* Mallev Castavarro sebagai rekan pertama Tokoh utama dalam *game Sword Master Of Destiny*. *Sprite* yang digunakan haruslah sesuai dengan *Face Graphic* dari karakter tersebut.

b. Face Graphic : *face graphic* adalah representasi wajah atau ekspresi dari suatu karakter. Didalam sebuah game *RPG (Role Playing Game)*, *face graphic* dari setiap karakter sangat dibutuhkan dalam

menunjukkan suatu ekspresi karakter yang sedang melakukan interaksi. Tampilan *window Face Graphic* ditunjukkan dalam gambar 4.8.

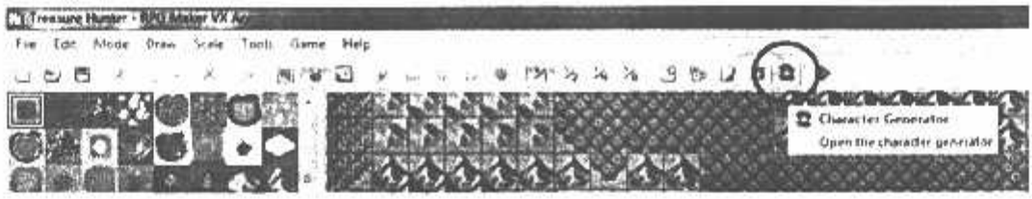


Gambar 4.8 *Window Face Graphic* pada *RPG Maker VX Ace*.

4.2.2. Pembuatan karakter dengan *Tools Character Generator*

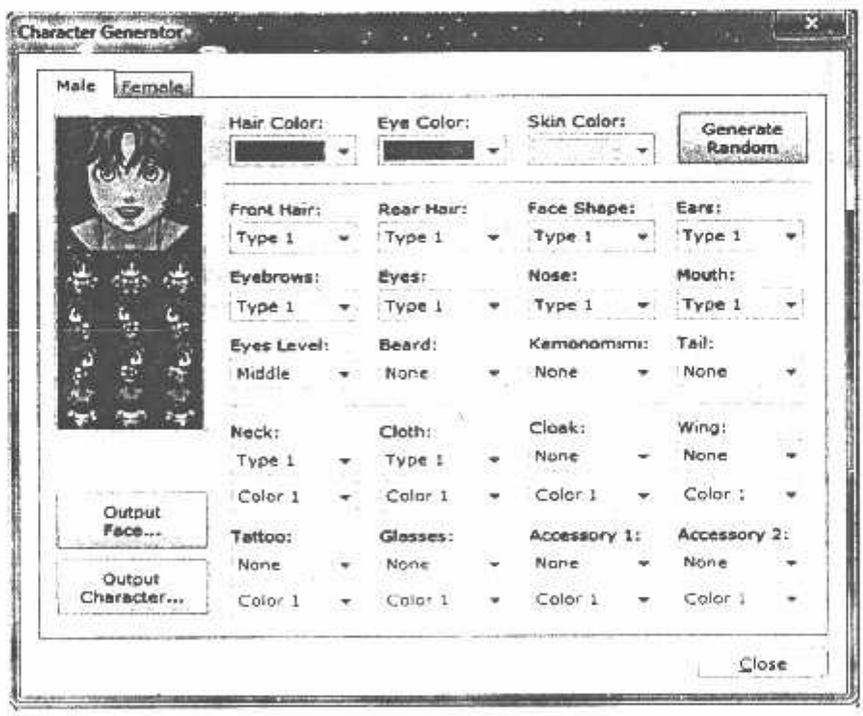
Pada *engine game RPG Maker VX Ace*, sang *game developer* (pembuat *game*) tidak perlu lagi susah-susah untuk membuat karakter yang nantinya akan dimainkan oleh *player*. *RPG Maker VX Ace* sudah memberikan *fitur* yang memuaskan bagi para *game developer*. *Fitur* yang memuaskan tersebut adalah sudah tersedianya *charset* tiap-tiap karakter yang siap digunakan oleh *game developer*. *Charset* yang sudah tersedia adalah *Character graphic* tiap-tiap karakter dalam bentuk *sprite* dan *Face Graphic*.

Namun *fitur* tersebut memiliki keterbatasan yang terkadang kurang memuaskan bagi seorang *game developer* (pembuat *game*) dalam hal penentuan karakter. Hal tersebut dibuktikan dengan terbatasnya *Face Graphic* dari suatu karakter yang mampu mengekspresikan hal-hal tertentu yang dialami oleh karakter dalam suatu kondisi. Sebagai penanggulangan akan hal tersebut, *engine game RPG Maker VX Ace* memberikan *fitur Character Generator* sebagai penambah keleluasaan bagi *game developer* (pembuat *game*) untuk membuat karakter yang mereka inginkan baik itu dari segi desain *sprite* untuk *character graphic* maupun ekspresi wajah untuk *face graphic*. Didalam *engine RPG Maker VX Ace*, pilihan *Character Generator* ditunjukkan pada gambar 4.9



Gambar 4.9 Pilihan *Character Generator*.

Dengan bantuan pilihan *Character Generator* seperti yang ditunjukkan pada gambar 4.11, seorang *game developer* (pembuat *game*) mampu memaksimalkan kinerja *game* dengan benar. Dengan *fitur* tambahan ini, karakter dapat dibuat semenarik mungkin dari segi penampilan. Baik itu desain *sprite* untuk *character graphic* dan ekspresi wajah yang berbeda-beda dari satu karakter saat berada dalam suatu kondisi tertentu yang mendukung *face graphic* karakter. Dengan *fitur character generator*, karakter wanita dan laki-laki bisa dibuat sedemikian rupa sesuai keinginan. Gambar untuk *window character generator* ditunjukkan dalam gambar 4.10.



Gambar 4.10 Window *Character Generator*.

Character generator memiliki berbagai macam pilihan yang dapat dipergunakan untuk membuat karakter wanita dan laki-laki. Untuk membuat karakter laki-laki, tekan saja pilihan *Male* pada pojok kiri atas yang terdapat

pada *character generator*, dan untuk membuat karakter wanita, tekan saja pilihan *Female* seperti yang ditunjukkan pada gambar 4.10. berikut akan dijelaskan berbagai macam pilihan dari *character generator*:

a. *Hair color* : pilihan ini digunakan untuk memilih warna rambut karakter.
b. *Eye color* : pilihan ini digunakan untuk memilih warna mata karakter.
c. *Skin color* : pilihan ini digunakan untuk memilih warna kulit karakter.
d. *Front hair* : pilihan ini digunakan untuk memilih jenis rambut karakter.
e. *Rear hair* : pilihan ini digunakan untuk memilih jenis rambut karakter.
f. *Face shape* : pilihan *face shape* digunakan untuk memilih bentuk wajah.
g. *Ears* : pilihan ini digunakan untuk memilih bentuk telinga karakter.
h. *Eyebrows* : pilihan ini digunakan untuk memilih alis mata karakter.
i. *Eyes* : pilihan ini digunakan untuk memilih bentuk mata karakter.
j. *Nose* : pilihan ini digunakan untuk memilih bentuk hidung karakter.
k. *Mouth* : pilihan ini digunakan untuk memilih bentuk mulut karakter.
l. *Eyes level* : pilihan *eyes level* adalah menentukan jarak pandang dari letak mata karakter. Letak mata tersebut berada di atas, tengah, atau bawah.
m. *Beard* : pilihan ini digunakan untuk menambahkan kumis pada karakter.
n. *Kemonomimi*: pilihan ini digunakan untuk menambahkan bentuk telinga hewan pada karakter.
o. *Tail* : pilihan ini digunakan untuk menambahkan bentuk ekor hewan pada karakter.

p. *Neck* : pilihan ini digunakan untuk merubah tampilan kerah baju yang dipakai oleh karakter. Akan terlihat pada *face graphic*.

q. *Cloth* : pilihan ini digunakan untuk memilih pakaian karakter.

r. *Cloak* : pilihan ini digunakan untuk menambahkan penutup bahu pada karakter. Dapat dipakai jika karakter adalah prajurit elit atau prajurit kerajaan.

s. *Wing* : pilihan ini digunakan untuk menambahkan sayap pada karakter.

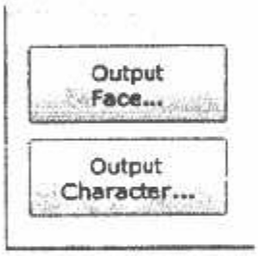
t. *Tattoo* : pilihan ini digunakan untuk menambahkan hiasan tato pada wajah karakter.

u. *Glasses* : pilihan ini digunakan untuk menambahkan kacamata.

v. *Accessory 1*: pilihan ini digunakan untuk memberikan bermacam bentuk aksesoris pada karakter.

w. *Accessory 2*: pilihan pilihan ini digunakan untuk memberikan bermacam bentuk aksesoris pada karakter.

Karakter yang telah dibuat tersebut harus disimpan terlebih dahulu agar dapat digunakan didalam permainan. Gambar 4.11 menunjukkan cara untuk menyimpan grafik karakter yang telah dibuat.

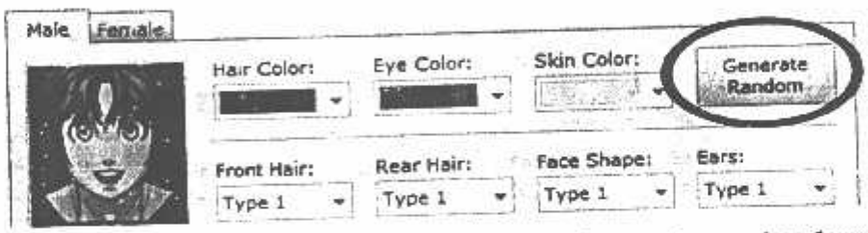


Gambar 4.11 Pilihan *Save* untuk menyimpan karakter.

Pada gambar 4.11 diperlihatkan sebuah *option* untuk menyimpan karakter yang telah dibuat. Pada *Character Generator*, *option* tersebut dapat ditemukan di pojok kiri bawah pada menu *Character Generator*. Untuk menyimpan *face graphic* yang menunjukkan ekspresi wajah karakter, dapat ditekan tombol *Output Face* seperti yang ditunjukkan pada gambar 4.13 pada menu *Character Generator*. Hasil keluaran *face graphic* tersebut akan disimpan pada folder aplikasi *engine game* yang telah diinstal, yaitu pada folder *Documents/RPGVXAce Sword Master Of Destiny/Graphics/Faces*.

Sedangkan hasil keluaran untuk desain *character graphic* berupa *sprite*, cara yang sama dapat dilakukan dengan menekan tombol *Output Character* pada menu *Character Graphic*. Hasil keluaran *character graphic* berupa desain *sprite* tersebut akan disimpan pada folder aplikasi *engine game* yang telah diinstal, yaitu pada folder *Documents/RPGVXAce/Sword Master Of Destiny/Graphics/Characters*.

Seorang *game developer* (pembuat game) terkadang dibuat bingung tentang bagaimana cara membuat karakter original sendiri yang nantinya akan dimainkan. Untuk menyiasati hal itu, menu *character generator* menyediakan option *Generate Random*. *Generate Random* berfungsi membantu seorang *game developer* (pembuat game) membentuk karakter secara acak dari berbagai macam pilihan. Gambar option *Generate Random* ditunjukkan pada gambar 4.12.

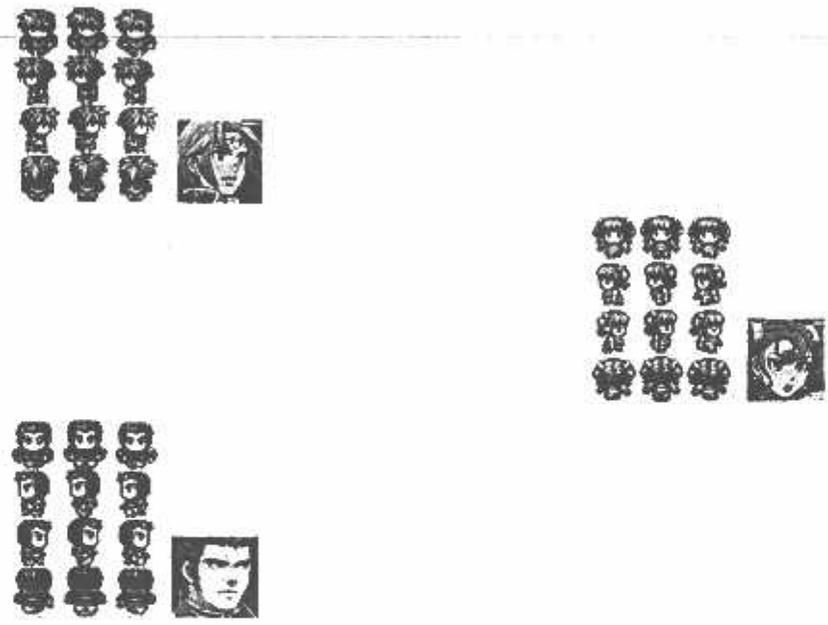


Gambar 4.12 Pilihan *Generate Random* untuk menyimpan karakter.

Option Generate Random dapat ditemukan di pojok kanan atas pada menu *Character Generator* seperti yang ditunjukkan pada gambar 4.14. Dengan *option* tersebut, seorang *game developer* (pembuat game) tidak perlu direpotkan menentukan satu persatu pilihan yang tersedia didalam menu *Character Generator*. Namun *option Generate Random* memiliki kelemahan didalam membuat karakter. Hal tersebut dibuktikan dengan hasil yang didapat terkadang tidak sesuai dengan apa yang diinginkan, karena sistem memilihkan karakter secara acak dari setiap pilihan yang tersedia. Tetapi memiliki keuntungan mempermudah proses pembuatan dan menyingkat waktu bagi seorang *game developer* (pembuat game).

4.2.3. Karakter Utama yang dimainkan *player*

Didalam sebuah permainan *RPG (Role Playing Game)* pada umumnya, karakter yang nantinya akan dimainkan oleh *player* harus dibuat terlebih dahulu. Karakter yang dimainkan terdiri dari karakter laki-laki dan karakter perempuan. *Player* diberi fasilitas untuk memilih karakter kesukaan mereka. Pada *game Sword Master Of Destiny*, *player* diberikan 3 karakter yang hanya bisa dipilih 1. Masing-masing terdiri dari 2 karakter laki-laki dan 1 karakter perempuan. Gambar desain *sprite* dan *face graphic* dari karakter utama ditunjukkan pada gambar 4.13.

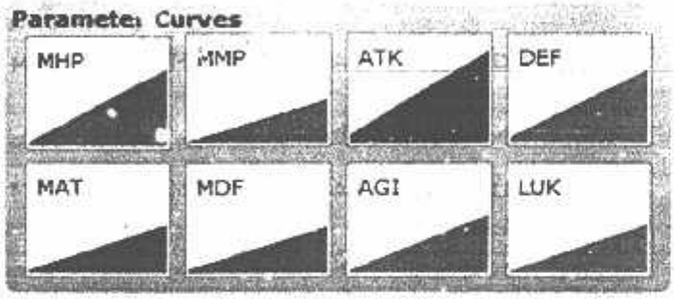


Gambar 4.13 Desain *sprite* dan *face graphic* karakter utama.

Pada gambar 4.13 ditunjukkan bahwa setiap karakter memiliki *Class* yang berbeda. Fungsi *class* adalah membedakan kemampuan antar karakter dari jenis *weapon* (senjata) dan *Skill* (jurus) yang akan digunakan didalam pertarungan. Sebelum memulai *game*, *player* akan diberikan fasilitas memilih 1 karakter yang akan dimainkan dari 3 karakter yang sudah tersedia.

4.2.4. *Parameter Curve* pada *RPG Maker VX Ace*

Parameter Curve adalah atribut yang berisi keterangan tentang status suatu karakter. Status tersebut menjelaskan keadaan dari suatu karakter yang nantinya akan dimainkan didalam *game*. Didalam *RPG Maker VX Ace*, ada 8 atribut yang berfungsi menjelaskan status karakter. Diantaranya yaitu, *MHP (Maximum Hit Point), MMP (Maximum Magic Point), ATK (Attack power), DEF (Defense power), MAT (Magic Attack power), MDF (Magic Defense power), AGI (Agility), dan LUK (Luck).* Tampilan *Parameter Curve* ditunjukkan pada gambar 4.14.



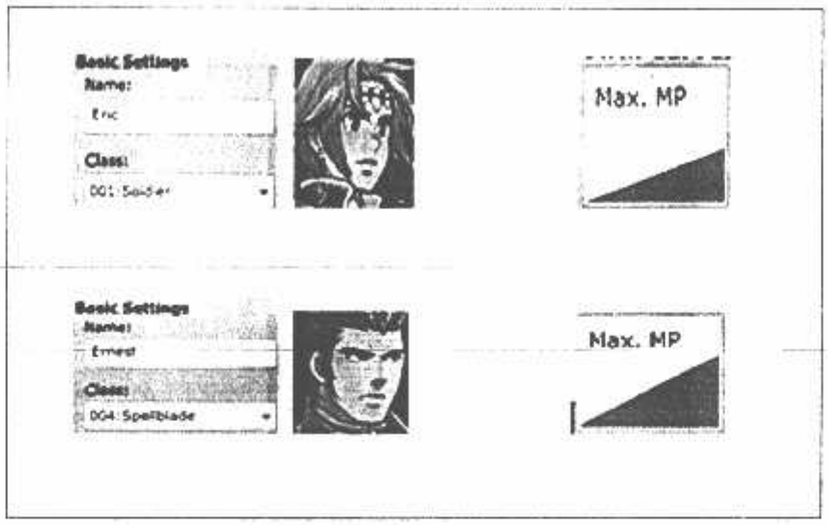
Gambar 4.14 *Parameter Curves.*

Pada gambar 4.14 terdapat 8 atribut penting yang menegaskan status suatu karakter. Berikut adalah penjelasan dari 8 atribut pada *Parameter Curves*:

a. *MHP (Maximum Hit Point)* : atribut ini menegaskan mengenai nyawa yang dimiliki oleh karakter. Bisa disebut juga sebagai *HP (Hit Point)* yang dimiliki oleh karakter. *Maximum Hit Point* adalah nyawa suatu karakter yang berupa nilai maksimal yang dapat ditentukan pada karakter.

b. *MMP (Maximum Magic Point)* : atribut ini menegaskan mengenai status kemampuan sihir yang dapat digunakan oleh karakter. Atribut *MMP (Maximum Magic Point)* tidak bisa diterapkan pada semua karakter. Hal ini disebabkan, pengaturan *Parameter Curves* juga diperhatikan dari jenis *Class* yang sudah ditetapkan pada karakter sebelumnya. Dalam contoh kasus adalah karakter utama Amakusa Kleinn dan karakter pendamping Mallev Castavarro memiliki nilai *MMP (Maximum Magic Point)* yang berbeda yang diperlihatkan pada

Parameter Curves. Hal ini disebabkan *Class* yang ditetapkan pada Amakusa Kleinn adalah *Soldier* (Prajurit), dan *Class* yang ditetapkan pada Mallev Castavarro adalah *Witch* (Penyihir).



Gambar 4.15 Perbedaan nilai *MMP*.

Pada gambar 4.15 terdapat perbedaan nilai *MMP (Maximum Magic Point)* yang mencolok antara karakter utama *Eric* dengan karakter pendamping *Ernest*. Hal ini dikarenakan faktor *Class* yang menyebabkan perbedaan tersebut. Amakusa Kleinn ditetapkan memiliki *Class Soldier* (Prajurit). *Class Soldier* (Prajurit) memiliki kekuatan fisik yang besar, namun memiliki kekuatan sihir yang minim atau bisa disebut juga tidak memiliki kekuatan sihir. Sedangkan *Ernest* termasuk dalam *Class Spellblade* (Penyihir). *Class Witch* (Penyihir) sesuai namanya adalah seorang penyihir, tentu dalam *Parameter Curves* nilai *MMP (Maximum Magic Point)* milik Mallev Castavarro ditetapkan secara maksimum dan nilai *MMP (Maximum Magic Point)* milik Amakusa Kleinn ditetapkan secara minimum atau hampir tidak memiliki kekuatan sihir.

c. *ATK (Attack Power)* : atribut ini menegaskan mengenai besarnya kekuatan serangan fisik yang dimiliki oleh karakter. Besar serangan yang ditetapkan pada karakter membantu karakter menang mudah didalam pertarungan.

d. *DEF (Defense Power)* : atribut ini menegaskan mengenai besarnya kekuatan pertahanan yang dimiliki oleh karakter dari serangan fisik. Nilai *DEF (Defense Power)* yang besar membantu karakter tidak mudah mati didalam pertarungan, karena besarnya nilai *DEF (Defense Power)* juga berpengaruh pada nilai *HP (Hit Point)* karakter.

e. *MAT (Magic Attack Power)* : atribut ini menegaskan mengenai besarnya kekuatan serangan sihir yang dimiliki oleh karakter. Nilai *MAT (Magic Attack)* berhubungan erat dengan nilai *MMP (Maximum Magic Point)*. Hal ini disebabkan *MAT (Magic Attack)* tidak akan bisa berfungsi jika elemen bagian *MMP (Maximum Magic Point)* tidak terisi terlebih dahulu pada pertarungan. Didalam pertarungan, karakter yang mampu menggunakan *Magic* (sihir) tidak akan bisa mengeluarkan kekuatannya jika nilai *MMP (Maximum Magic Point)* tidak terpenuhi. Jika nilai *MMP (Maximum Magic Point)* terpenuhi walaupun hanya sebagian, karakter tersebut mampu menggunakan kekuatan sihir.

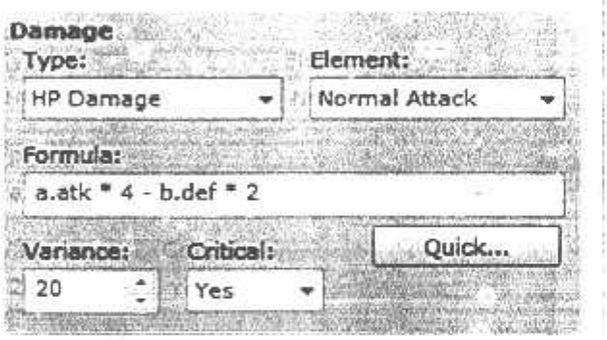
f. *MDF (Magic Defense Power)* : atribut ini menegaskan mengenai besarnya kekuatan pertahanan yang dimiliki oleh karakter dari serangan sihir. *MDF (Magic Defense Power)* berbeda dengan *DEF (Defense Power)* meskipun memiliki arti yang sama yaitu pertahanan. *DEF (Defense Power)* adalah kemampuan bertahan karakter dari serangan fisik, sedangkan *MDF (Magic Defense Power)* adalah kemampuan bertahan karakter dari serangan sihir.

g. *AGI (Agility)* : atribut ini menegaskan mengenai besarnya kecepatan yang dimiliki oleh suatu karakter. Kecepatan ini mengacu pada perintah menyerang secara bergantian bagi karakter dalam putaran permainan. Karakter yang memiliki nilai *AGI (Agility)* lebih tinggi dari karakter lain, memiliki kesempatan menyerang karakter lain terlebih dahulu. Sedangkan karakter lain yang memiliki nilai *AGI (Agility)* lebih rendah, mendapatkan kesempatan menyerang pada putaran berikutnya. Didalam *RPG Maker VX Ace*, nilai *AGI (Agility)* menjadi nilai acuan utama bagi karakter untuk mendapatkan kesempatan menyerang pertama kali.

h. LUK (Luck) : atribut ini menegaskan mengenai pengkalkulasian kesempatan bagi karakter untuk menerima / memberikan *status effect* dari / pada karakter lain. Sebagai contoh jika nilai *LUK (Luck) player* lebih tinggi dari *enemy*, maka *player* memiliki kesempatan besar mempengaruhi *enemy* dengan *status effect* yang dimiliki, dan juga memungkinkan bagi *player* untuk tidak terkena *status effect* dari serangan yang dikeluarkan oleh *enemy*.

4.2.5. Penentuan *Formula* pada *RPG Maker VX Ace*

Didalam *RPG Maker VX Ace*, penentuan *formula* sangat diperhatikan pada sistem pertarungan antara *player* dengan *enemy*. Dengan *formula*, proses pengitungan untuk pengurangan nilai *HP (Hit Point)* antara *player* dengan *enemy* dapat ditentukan dengan mudah. Hal tersebut diambil berdasarkan penghitungan *formula* pada kolom *damage*. Gambar penghitungan *formula* pada *RPG Maker VX Ace* ditunjukkan pada gambar 4.16.



Gambar 4.16 Penentuan *formula* pada kolom *Damage*.

Pada gambar 4.16 diperlihatkan ada beberapa unsur yang penting didalam menentukan sebuah *formula*. Berkurangnya *HP (Hit Point)* pada karakter, besarnya jumlah *Attack* (serangan) suatu karakter, kuat dan lemahnya *Defense* (pertahanan) yang dimiliki karakter, semuanya diatur dalam tabel *formula*. Keterangan mengenai *formula* dapat dilihat pada tabel 4.1.

Tabel 4.1 Keterangan yang dipakai dalam *formula*.

Formula	Keterangan
A	Player
B	Enemy

V	Variable
	Variable berisi keterangan mengenai penghitungan formula yang dilakukan

Variable formula seperti yang ditunjukkan pada tabel 4.1 berisi keterangan mengenai besarnya nilai serangan, besarnya nilai bertahan dan cara berkurangnya nilai *HP (Hit Point)* dari suatu karakter. Keterangan mengenai *formula* secara lengkap dapat dilihat pada tabel 4.2.

Tabel 4.2 Keterangan *formula* secara lengkap.

Formula	Keterangan
A	Player
B	Enemy
V	Variable
a.v	a.(atk*4)
b.v	b.(def*2)
a.atk*4	Player memiliki kekuatan serangan (atk) yang nilai kekuatannya dikalikan 4.
b.def*2	Enemy memiliki kekuatan bertahan (def) yang nilai kekuatannya dikalikan 2.
Formula	a.atk*4-b.def*2
a.atk*4-b.def*2	Nilai formula adalah kekuatan serangan (atk) player dikalikan 4 dikurangi dengan kekuatan bertahan (def) musuh dikalikan 2.

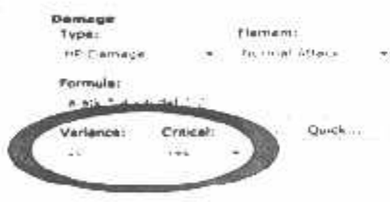
Pada tabel 4.2 dijelaskan bahwa contoh *formula* berdasarkan pada nilai serangan dari *player*, dan juga berdasarkan pada nilai bertahan dari *enemy*. Besarnya jumlah serangan serta bertahan ditentukan pada jumlah perkalian yang mengikuti *variable*. Sebagai penjelas lebih lanjut mengenai *formula*, berikut akan ditampilkan contoh lain dari beberapa *formula* yang juga dipakai didalam *game* yang ditunjukkan pada tabel 4.3.

Tabel 4.3 Contoh *Formula* lain yang dipakai didalam *game*.

Formula	Keterangan
b.mhp / 4	Nilai formula adalah hasil penghitungan dari magic hit point (mhp) milik musuh dibagi 4

	Formula seperti ini diterapkan pada player yang memiliki kemampuan untuk menyerap kekuatan sihir musuh. Hasil penghitungan dari formula ini akan mengakibatkan kekuatan sihir musuh berkurang.
1000 + a.mat * 2	Nilai formula adalah nilai magic attack (mat) milik player dikali 2 dan ditambah 1000. Formula seperti ini diterapkan pada player yang memiliki kekuatan untuk mengembalikan nyawanya.
400 + a.mat * 2 - b.mdf * 2	Nilai formula adalah kekuatan serangan sihir milik player (mat) dikali 2 dan ditambah 400 kemudian dikurangi dengan kekuatan bertahan dari serangan sihir milik musuh (mdf) dikali 2. Formula seperti ini diterapkan pada karakter yang memiliki kekuatan sihir baik dalam posisi menyerang maupun bertahan, hanya atribut sihir saja yang dimasukkan didalam perhitungan formula. Sebagai contoh adalah mat (magic attack) dan mdf (magic defense).
a.atk * 4 + a.def * 2 - b.def * 2	Nilai formula adalah kekuatan serangan milik player (atk) dikali 4 ditambah kekuatan bertahan milik player (def) dikali 2 kemudian dikurangi dengan kekuatan bertahan milik musuh dikali 2. Formula seperti ini diterapkan pada jenis serangan besar yang mampu melukai musuh.

Selain *formula* yang digunakan didalam penghitungan nilai untuk pertarungan, hal lain yang perlu diperhatikan adalah *Variance* dan *Critical* pada kolom *Damage* yang ditunjukkan pada gambar 4.17.



Gambar 4.17 *Variance* dan *Critical* sebagai tambahan *formula*.

Pada gambar 4.17 diperlihatkan sebuah nilai dari *Variance* dan opsi pilihan dari *Critical*. Fungsi *variance* adalah memberikan nilai kisaran variasi *damage* dari perhitungan *formula*, dan *Critical* adalah opsi untuk menentukan nilai suatu serangan mencapai nilai puncak atau tidak berdasarkan opsi "*YES*" atau "*NO*".

Berikut adalah cara kerja *Variance* berdasarkan penghitungan dari *formula* pada kolom *damage RPG Maker VX Ace*:

a. Sebuah *formula* bernilai *a.atk*4 - b.def*2*.
   Dimana a = *player* dan b = *enemy*. *Variable* yang mengikuti adalah *atk*4* dan *def*2*.
b. Kemudian dimasukkan nilai *atk (attack)* dari *player* sebesar 200. Dan *def (defense)* dari *enemy* sebesar 150.
c. *Atk = 200, def = 150.*
d. Penghitungan untuk nilai *formula* adalah:
   = (*a.atk*4 - b.def*2*)
   = (a.200*4 - b.150*2)
   = a.800 - b.300 = 500
   Hasil keluaran *damage* adalah sebesar 500.
e. Nilai 500 adalah jumlah *damage* yang nantinya akan diterima oleh *enemy*.
f. Setelah proses penghitungan *formula* selesai, langkah selanjutnya adalah proses *variance*.
g. *Variance* adalah nilai kisaran variasi *damage* dari penghitungan *formula*, seperti yang ditunjukkan pada gambar 4.19, nilai *variance* diisi dengan 20. Maka perhitungan fungsi *variance* dapat dilihat pada tabel 4.4.

Tabel 4.4 Nilai kisaran *variance*.

Variance rendah	**Variance tinggi**
= Damage – Variance	= Damage + Variance
= 500 – 20	= 500 + 20
= 480	= 520

Dari tabel 4.4 dapat dijelaskan bahwa nilai kisaran *damage* yang diterima oleh *enemy* nantinya berkisar antara 480 hingga 520, dari jumlah normal *damage* yang berasal dari *formula* sebesar 500.

h. Jika didalam kolom *damage* pilihan *Critical* berisi "*YES*", maka *damage* dari serangan *player* akan bernilai tinggi dari proses *variance* yaitu 520 setiap *player* menyerang. Jika pilihan *Critical* berisi "*NO*", maka *damage* bernilai 520 akan jarang ditemui. Namun nilai *damage* tetap berkisar antara nilai 480 hingga 520.

4.3. Pengujian Aplikasi

4.3.1. Pengujian Menu Utama

Pada saat pertama kali *player* menjalankan *game Sword Master Of Destiny*, tampilan awal yang dijumpai oleh *player* adalah menu utama. Menu utama berisi 4 macam pilihan yang dapat diakses oleh *player* untuk memainkan *game*. Menu utama *game Sword Master Of Destiny* ditunjukkan pada gambar 4.18.

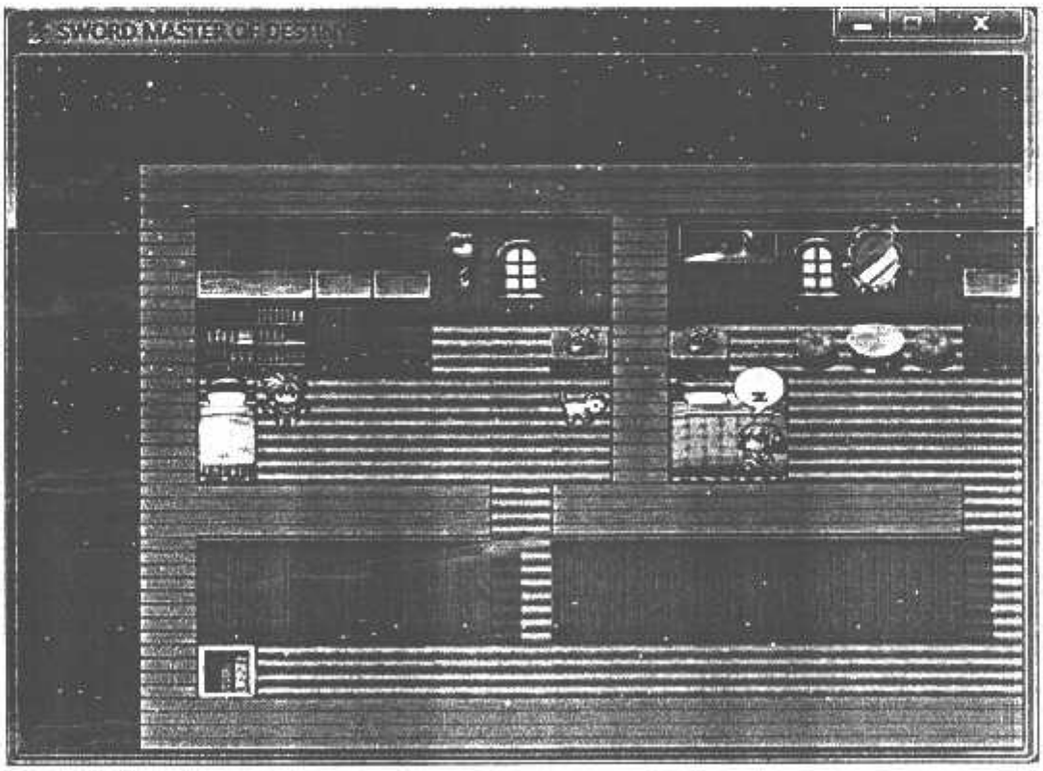
Gambar 4.18 Menu utama *game Sword Master Of Destiny*.

Pada gambar 4.18 diperlihatkan tampilan menu utama yang berisi 4 pilihan menu. Fungsi dari tiap-tiap pilihan menu dijelaskan sebagai berikut:

a. Permainan Baru : pilihan Mulai Baru adalah pilihan menu dimana *player* akan memulai memainkan *game*.
b. Lanjutkan : pilihan Lanjutkan adalah pilihan menu dimana *player* melanjutkan lagi permainannya dari simpanan data *game* yang telah disimpan sebelumnya.
c. Keluar : pilihan Keluar adalah pilihan menu dimana *player* dapat keluar dari *game*.

4.3.2. Pengujian Menu Mulai Baru

Pada menu mulai baru, *player* dapat mulai memainkan *game* . Namun sebelum *player* dapat memainkan *game*, *player* akan disuguhkan terlebih dahulu sejarah *game* yang menjadi inti cerita dari *Sword Master Of Destiny*. *Player* juga diberikan pilihan untuk melihat sejarah tersebut atau tidak. Tampilan sejarah awal mula *game* ditunjukkan pada gambar 4.19.

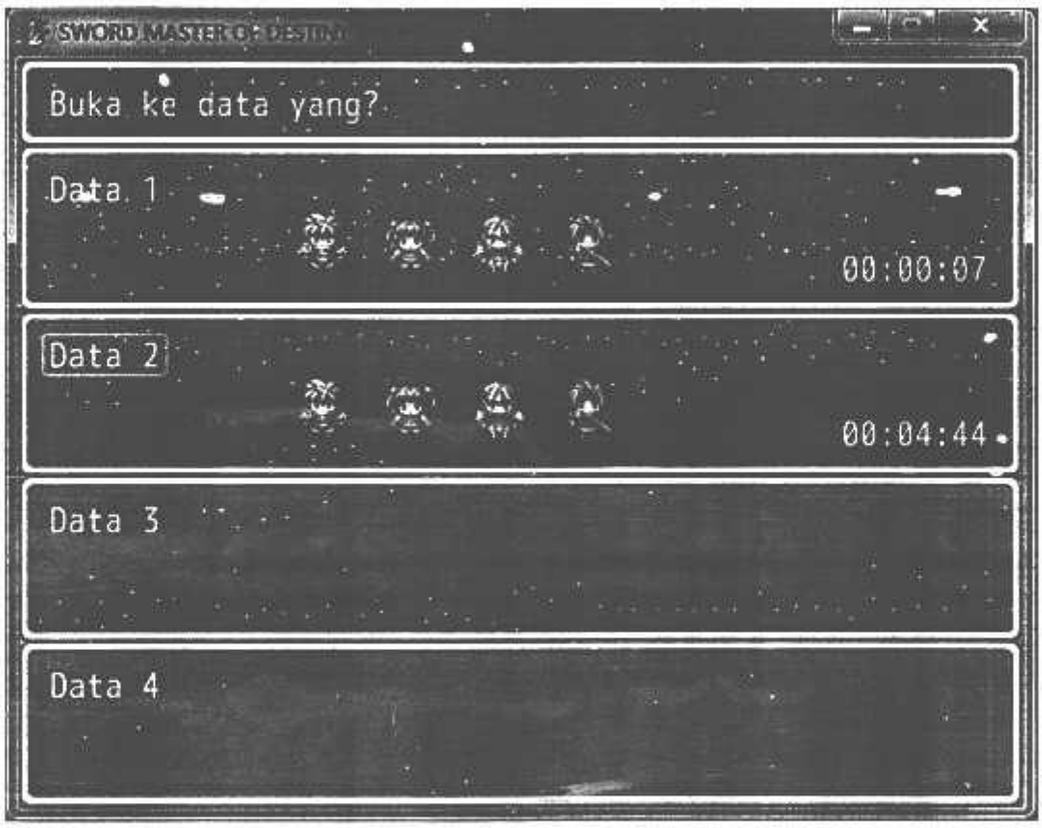


Gambar 4.19 Tampilan awal mula *game*.

Pada gambar 4.19 *Player* akan langsug disajikan *prolog* sejarah awal mula pada *game Sword Master Of Destiny*.Setelah *player* dihadapkan pada tampilan sejarah awal mula, *player* diberikan kesempatan untuk memilih karakter yang sudah tersedia didalam *game* sebelum *player* memainkan *game Sword Master Of Destiny*. Terdapat 3 pilihan karakter yang tersedia yang dapat dipilih oleh *player*.

4.3.3. Pengujian Menu Lanjutkan

Menu Lanjutkan adalah pilihan menu yang dapat dipilih oleh *player* jika *player* ingin memainkan kembali *game* yang sudah dimainkan sebelumnya, dan data simpanan *game* sudah tersimpan didalam berkas simpanan. Tampilan Menu Lanjutkan ditunjukkan pada gambar 4.20.

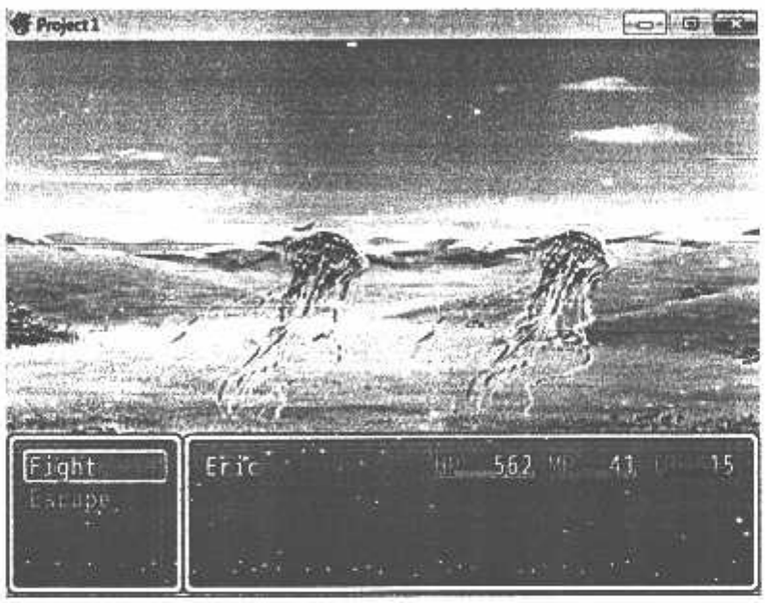


Gambar 4.20 Tampilan Menu Lanjutkan.

Pada Menu Lanjutkan, *player* diberikan *slot* (tempat) untuk menyimpan simpanan data *game* sebanyak 16 *slot*. Hal tersebut bertujuan untuk memudahkan *player* menyimpan data simpanan *game* yang dimainkan dalam jumlah besar.

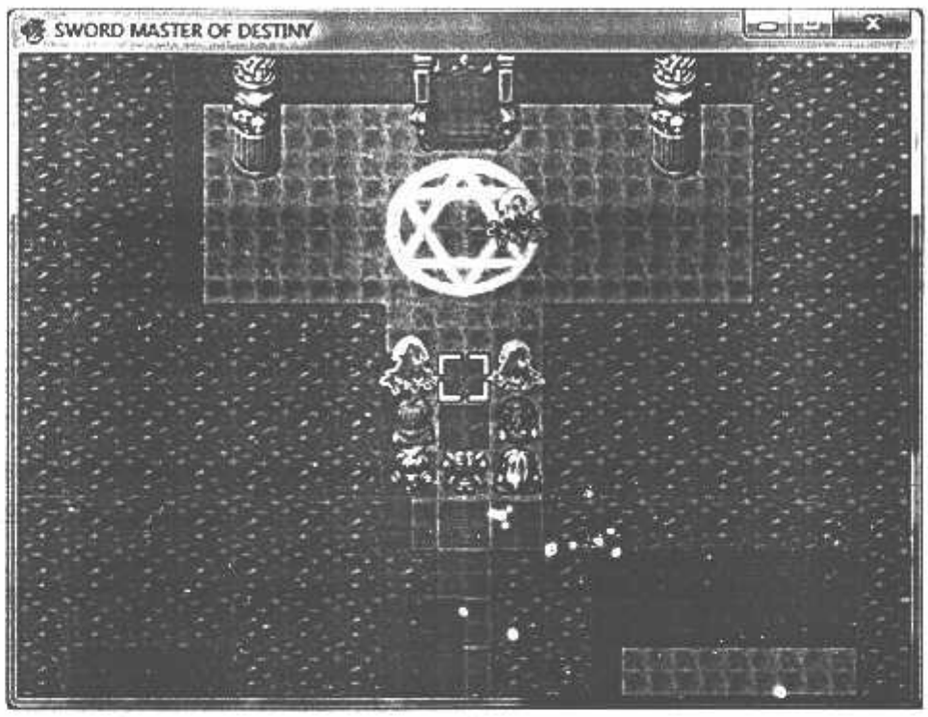
4.3.5. Pengujian Metode *Tactical Battle*

Metode *Tactical Battle* digunakan untuk merubah tampilan pertarungan antar karakter yang terjadi. Penentuan Metode *Tactical Battle* dengan menggunakan Grid squere, position Ability, dan Turn system sebagai media pendukung aplikasi. Tampilan pertarungan antar karakter sebelum ditambah dengan *Tactical Battle* ditunjukkan pada gambar 4.21.



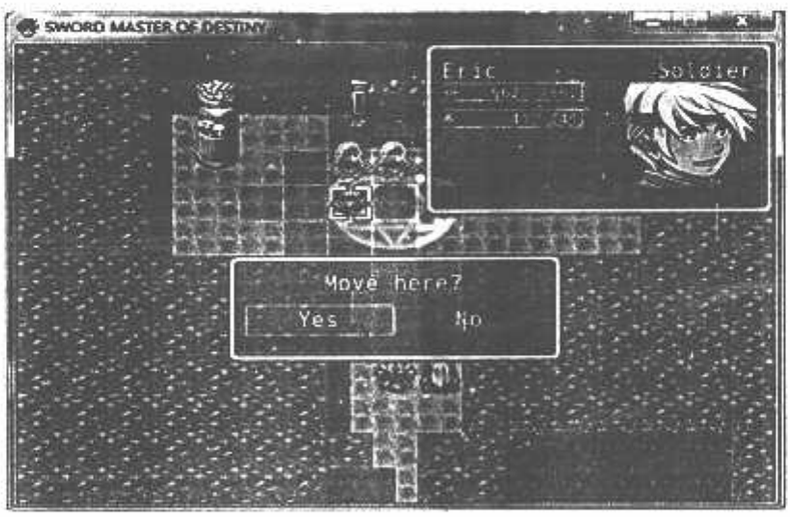
Gambar 4.21 Tampilan pertarungan sebelum ditambah *Tactical Battle*.

Pada gambar 4.21 sebelum ditambah media *Grid squere, position Ability, dan Turn system* pertarungan antar karakter terlihat biasa dan membosankan. Karakter dalam hal ini adalah *player* tidak terlihat saat bertarung melawan musuh. *Player* hanya dipresentasikan dengan tampilan nama beberapa atribut yang menyertainya. Sedangkan apabila ditambah dengan *Grid squere, position Ability, dan Turn system, player* dipresentasikan dengan *sprite* dan terlihat jelas serta ditambah dengan beberapa atribut yang diperlengkap. Tampilan *battle system* setelah ditambah *Metode Tactical Battle* ditunjukkan pada gambar 4.22.

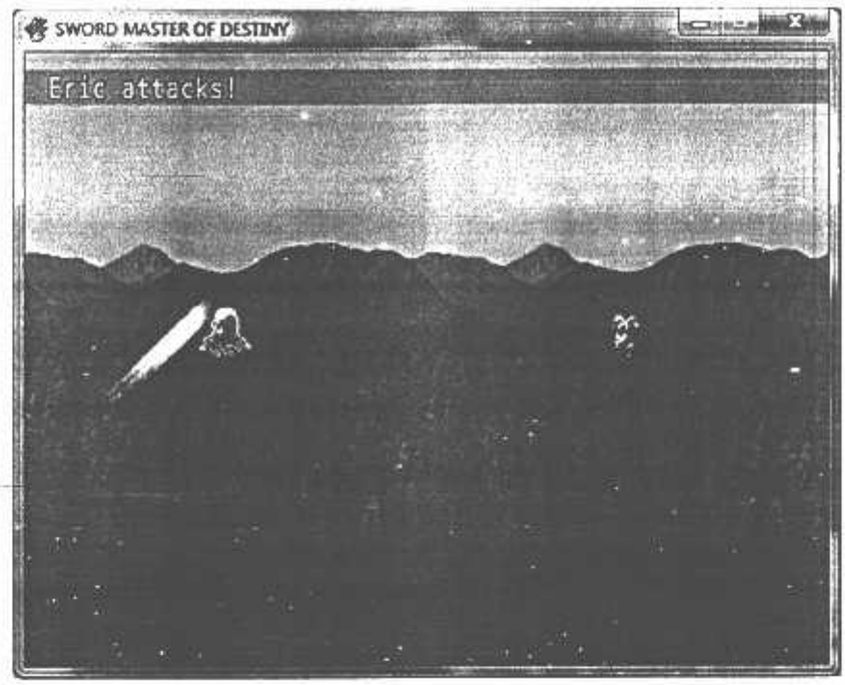


Gambar 4.22 Tampilan pertarungan setelah ditambah *Metode Tactical Battle*.

Pada gambar 4.22 *sprite* karakter *player* ditampilkan dan terlihat dengan jelas. *Sprite* sebagai bentuk representasi dari *player* ditampilkan dan dapat bergerak. Bukan hanya player saja yang dapat bergerak, karakter musuh juga dapat melakukan hal yang sama. Tampilan lain ditunjukkan pada gambar 4.23 dan gambar 4.24.



Gambar 4.23 Karakter *player* bergerak mendekati musuh.



Gambar 4.24 Karakter *player* menyerang musuh.

Pada gambar 4.24 diperlihatkan bahwa karakter *player* menyerang musuh dibantu dengan bantuan *8D Battler* sebagai penentu gerakan / aksi yang dilakukan. Aksi yang dilakukan oleh *player* adalah menebas musuh dengan serangan biasa. Kemudian setelah selesai melakukan sebuah aksi, posisi *player* ditentukan kembali pada tempatnya semula dengan *Holder*. Selain *player*, musuh juga memiliki prinsip yang sama. Musuh juga bergerak maju dan mundur sama seperti yang dilakukan oleh *player*.

4.3.6. Metode Pengujian

Metode Pengujian adalah salah satu metode yang digunakan untuk menguji keberhasilan dari game *Sword Master Of Destiny* ini. Metode pengujian yang digunakan untuk menguji *game Sword Master Of Destiny* ini yaitu menggunakan metode pengujian *blackbox* yang berarti pengujian aplikasi berdasarkan fungsi. Selain menggunakan Metode *Blackbox*, metode lain yang digunakan untuk pengujian *game* adalah dengan kuisioner. Kuisioner berfungsi sebagai tanggapan masyarakat terhadap sistem *game* yang sudah dibuat dan dikembangkan, baik itu baik maupun buruk. Hasil pengujian yang diperoleh melalui metode *blackbox* dapat dilihat pada tabel 4.5.

Tabel 4.5 *Tabel Pengujian melalui Metode Blackbox*

Pengujian Kesesuaian Fungsi dalam System Game				Keakuratan Dalam (%)
No	Menu	Proses	Hasil	
1	Splash Screen	Fadein, Fadeout	Ya	100 %
2	Menu Mulai Baru	Link to Mulai Baru	Ya	100 %
3	Menu Lanjutkan	Link to Lanjutkan	Ya	100 %
4	Menu Nilai tertinggi	Link to Nilai	Ya	80 %
5	Menu Keluar	Keluar	Ya	100 %
6	Sejarah Cerita	Story Game	Ya	100 %
7	Pemilihan Karakter	Select Character	Ya	100 %
8	Battle System	Battle Screen	Ya	100 %
9	Metode Dice Roll	System	Ya	100 %
10	Simpan Data	Link to Berkas	Ya	100 %
11	Pause	Game Stop	Ya	100 %
12	Time System	Show Minute	Ya	100 %

Keterangan dalam pengujian:

1. Ya = Berhasil.
2. Tidak = Gagal.

Berdasarkan hasil pengujian dari tabel 4.5 dapat dijelaskan bahwa tingkat keakuratan *game Sword Master Of Destiny* ini sudah mencapai 98,333 %. Tingkat keakuratan data tersebut diambil dari besarnya tiap nilai keakuratan sistem dalam persen (%) dibagi dengan jumlah menu sistem yang sedang diuji.

Didalam pengujian terdapat salah satu fungsi yang kurang bekerja maksimal. Yaitu pada bagian Nilai Tertinggi. Nilai tertinggi bernilai sebesar 80 %, hal tersebut dikarenakan saat *player* selesai mengalahkan Bos Musuh dan menang, akumulasi *score* (nilai) yang didapatkan tidak langsung masuk pada Papan Nilai. Agar nilai dapat masuk pada Papan Nilai, hal yang harus dilakukan adalah me-*refresh game Sword Master Of Destiny* tersebut. Setelah proses *refresh* selesai, nilai akumulasi kemenangan masuk dan tampak pada Papan Nilai.

Selain dengan metode *Blackbox*, pengujian juga dilakukan terhadap responden dengan cara menyebarkan kuisioner. Hasil tersebut dapat dilihat pada tabel 4.6.

Tabel 4.6 *Tabel Pengujian terhadap 20 responden.*

No	Kriteria Penilaian	Keterangan			
		SB	B	C	K
1	Layout antar muka game	8	10	2	-
2	Jalan cerita game	17	3	-	-
3	Kinerja game	17	2	1	-
4	Tingkat kemudahan game	18	2	-	-

Keterangan dalam pengujian:

1. SB = Sangat Baik.
2. B = Baik.
3. C = Cukup.
4. K = Kurang.

Berdasarkan pada tabel 4.6, hasil kesimpulan yang dapat diambil dari 20 orang responden adalah sebagai berikut:

1. *Layout* Antar Muka *Game*.

a. SB : 8 responden 40 %
b. B : 10 responden 50 %
c. C : 2 responden 10 %
d. K : 0 responden 0 %

2. Jalan Cerita *Game*.

a. SB : 17 responden 85 %
b. B : 3 responden 15 %
c. C : 0 responden 0 %
d. K : 0 responden 0 %

3. Kinerja *Game*.

a. SB : 17 responden 85 %

b. B : 2 responden 10 %

c. C : 1 responden 5 %

d. K : 0 responden 0 %

4. Tingkat Kemudahan *Game*.

a. SB : 18 responden 90 %

b. B : 2 responden 10 %

c. C : 0 responden 0 %

d. K : 0 responden 0 %

Dari data kuisioner diatas dapat disimpulkan bahwa:

1. 40% dari jumlah responden berpendapat bahwa *layout game* ini Sangat Baik, 50% responden berpendapat Baik, dan 10% responden berpendapat Cukup.
2. 85% dari jumlah responden berpendapat bahwa jalan cerita *game* ini Sangat Baik dan 15% responden berpendapat Baik.
3. 85% dari jumlah responden berpendapat bahwa kinerja dari *game* ini Sangat Baik, 10% responden berpendapat Baik, dan 5% responden berpendapat Cukup.
4. 90% dari jumlah responden berpendapat bahwa tingkat kemudahan *game* ini Sangat Baik dan 10% berpendapat Baik.

4.3.7. Pengujian Pada *Operating System*

Pengujian juga dilakukan terhadap Sistem Operasi yang digunakan. Pengujian terhadap Sistem Operasi dimaksudkan untuk mengetes apakah *game Sword Master Of Destiny* tersebut *kompatibel* dengan berbagai macam Sistem Operasi atau tidak. Terdapat 5 Sistem Operasi yang digunakan untuk proses pengujian, kelima Sistem Operasi tersebut ditunjukkan pada tabel 4.7.

Tabel 4.7*Tabel Pengujian Game pada Sistem Operasi yang digunakan*

Pengujian Game Pada Jenis Operating Sistem			Kesuksesan Dalam (%)
No	Operating Sistem	Keterangan	
1	Windows XP 2	Berjalan	100 %
2	Windows XP 3	Berjalan	100 %
3	Windows 7	Berjalan	100 %
4	Windows 8	Berjalan	100 %
5	Mac OS	Berjalan	100 %

Pada Tabel Pengujian *Game* berdasarkan Jenis sistem operasinya, *game Sword Master Of Destiny* diuji pada beberapa sistem operasi yang ada. Apakah *game Sword Master Of Destiny* ini dapat berjalan baik atau tidak di beberapa sistem operasi komputer. Hasil pengujian ditunjukkan pada tabel 4.7.

Pada hasil pengujian terhadap sistem operasi, *game Sword Master Of Destiny* berjalan dengan baik pada semua sistem operasi baik itu pada *Windows 7, Windows 8, Windows XP,* dan *Mac OS.*

4.3.8. Pengujian Resolusi *Screen Game*

Pengujian resolusi *screen game* pada tingkat resolusi komputer diperlukan untuk membuktikan apakah *game* dapat berjalan atau tidak pada *screen* resolusi tinggi maupun rendah. Hasil pengujian terhadap resolusi *screen game* ditunjukkan pada tabel 4.8.

Tabel 4.8*Tabel Pengujian Game pada Screen Resolusi yang digunakan.*

Resolusi Sistem Operasi	Resolusi 800 x 600	Resolusi 1024 x 728	Resolusi 1200 x 768
Windows XP	Berjalan	Berjalan	Berjalan Baik
Windows 7	Berjalan	Berjalan	Berjalan Baik
Windows 8	Berjalan	Berjalan	Berjalan Baik
Mac OS	Berjalan	Berjalan	Berjalan Baik

Pada tabel 4.8 ditunjukkan hasil pengujian *game* dari berbagai ukuran *screen resolution*. Pada *screen* resolusi ukuran 800 x 600 dan 1024 x 728 *game* dapat berjalan, namun ukuran piksel dari *game* buram dan tidak jelas.

Sedangkan pada ukuran normal screen resolution 1200 x 768 game dapat berjalan dengan baik. Game tidak pecah dan tampilan tidak buram.

BAB V
PENUTUP

5.1 Kesimpulan

Game Sword Master Of Destiny dengan Metode *Tactical Battle* ini menggunakan *Grid squere, position Ability, dan Turn system* sebagai media *battle* (pertarungan) antar karakter. Grid squere adalah tempat yang disediakan untuk penempatan karakter yang akan melakukan pertarungan *Position Squere Ability* memiliki fungsi yang berbeda Grid Squere. *Position Squere Ability* berfungsi untuk meletakkan karakter pada setian posisi. *Turn* System berfungsi sebagai penghitungan waktu untuk melakukan serangan terlibih dahulu. *8D Battler* berfungsi sebagai penentu gerakan karakter dalam menyerang.

Ketiga konsep dan *8D Battler* yang diterapkan didalam sistem *battle*, Japat membuat karakter baik itu *player* maupun musuh bergerak bervariasi. Gerakan yang bervariasi ini membuat karakter bergerak untuk menyerang lawan.

Setelah melalui beberapa tahapan perancangan, implementasi, dan pengujian, *game Sword Master Of Destiny* ini dapat disimpulkan sebagai berikut :

1. *Game Sword Master Of Destiny* dengan Metode *Tactical Battle* ini dapat diterima dengan sangat baik oleh semua kalangan masyarakat sebagai media hiburan dan penarik minat masyarakat terhadap *Classical RPG (Role Playing Game)* yang sudah dilupakan dan hampir ditinggalkan oleh sebagian orang.
2. *Game Sword Master Of Destiny* ini dapat dijalankan pada semua media komputer dan Laptop dengan Sistem Operasi yang berbeda.
3. Dengan Metode *Tactical Battle* dapat disimpulkan bahwa sebagian besar responden sangat menyukai tampilan *battle system* pada *game RPG (Role Playing Game)* jenis *Classical*. Hal tersebut dibuktikan dengan hasil pengujian pada sebagian besar responden. Dari 20 responden 40 % responden menyukai tampilan awal *game*, 85 % responden menyukai jalan cerita *game*, 85 % responden menyukai kinerja dari *game*, dan 90 % responden menyukai tingkat kemudahan *game* yang dimainkan.
4. Ukuran file *game Sword Master Of Destiny* ini adalah 50,0 Mb.

5.2 Saran

Dalam pembuatan *game Sword Master Of Destiny* ini masih jauh dari sempurna. Karena masih banyaknya kekurangan - kekurangan yang perlu untuk ditambahkan antara lain :

1. *Game Sword Master Of Destiny* ini masih perlu ditambahkan sistem peta *(World Map)*. Karena *player* hanya dapat melihat peta *(World Map)* saat *player* hanya berada di penginapan. Dan sistem peta didalam *game Sword Master Of Destiny* ini belum dapat menunjukkan letak / lokasi *player* berada.
2. Didalam *game Sword Master Of Destiny* ini perlu ditambahkan beberapa *notifikasi* (pemberitahuan) tentang simpanan data *game* yang sudah dilakukan oleh *player* saat bermain sebelumnya. Hal tersebut perlu dilakukan dikarenakan saat *player* menyimpan simpanan data *game*, dalam sistem masih belum terdapat *notifikasi* "Data Tersimpan".

DAFTAR PUSTAKA

[1] Jasson. 2009. *Role Playing Game (RPG) Maker*. Yogyakarta: CV ANDY OFFSET.

[2] Wahana Komputer. 2014. *Tips and Trik RPG Maker VX Ace*. Andi Publisher.

[3] Duggan,Michael. 31/05/2011.*RPG Maker For Teens*. Jakarta: CV ANDY OFFSET.

[4] Alfian, Theolius. 2012. 24 *Jam Pinter Desain RPG Maker*. Jakarta: GRAMEDIA.

[5] Yami Engine Ace. 2012. *Battle System Symphony*. Jakarta: CV ANDY OFFSET.

[6] Subangkit, Teguh. S. 2011. *Sejarah-sejarah Dunia dan Misteri yang Tersembunyi*. Surabaya: Dian Ilmu.

[7] Permana, Rizal. 2012. *Dasar-dasar Pemrograman Bahasa Ruby dalam RPG Maker*.Semarang: GRAMEDIA.

[8] RPG Maker VX Community. 2012. *Tutorial dan Dasar-dasar membuat Game dengan RPG Maker VX Ace*.

Source Code System Game RPG Maker VX Ace

```
#==============================================================================
=======
# ** Game_System
#------------------------------------------------------------------------------
#  This class handles system data. It saves the disable state of saving and
# menus. Instances of this class are referenced by $game_system.
#==============================================================================
=======

class Game_System
  #--------------------------------------------------------------------------
  # * Public Instance Variables
  #--------------------------------------------------------------------------
  attr_accessor :save_disabled          # save forbidden
  attr_accessor :menu_disabled          # menu forbidden
  attr_accessor :encounter_disabled     # encounter forbidden
  attr_accessor :formation_disabled     # formation change forbidden
  attr_accessor :battle_count           # battle count
  attr_reader   :save_count             # save count
  attr_reader   :version_id             # game version ID
  #--------------------------------------------------------------------------
  # * Object Initialization
  #--------------------------------------------------------------------------
  def initialize
    @save_disabled = false
    @menu_disabled = false
    @encounter_disabled = false
    @formation_disabled = false
    @battle_count = 0
    @save_count = 0
    @version_id = 0
    @window_tone = nil
```

```
    @battle_bgm = nil
    @battle_end_me = nil
    @saved_bgm = nil
  end
  #--------------------------------------------------------------------------
  # * Determine if Japanese Mode
  #--------------------------------------------------------------------------
  def japanese?
    $data_system.japanese
  end
  #--------------------------------------------------------------------------
  # * Get Window Color
  #--------------------------------------------------------------------------
  def window_tone
    @window_tone || $data_system.window_tone
  end
  #--------------------------------------------------------------------------
  # * Set Window Color
  #--------------------------------------------------------------------------
  def window_tone=(window_tone)
    @window_tone = window_tone
  end
  #--------------------------------------------------------------------------
  # * Get Battle BGM
  #--------------------------------------------------------------------------
  def battle_bgm
    @battle_bgm || $data_system.battle_bgm
  end
  #--------------------------------------------------------------------------
  # * Set Battle BGM
  #--------------------------------------------------------------------------
  def battle_bgm=(battle_bgm)
    @battle_bgm = battle_bgm
  end
```

```
#--------------------------------------------------------------------------
# * Get Battle End ME
#--------------------------------------------------------------------------
def battle_end_me
 @battle_end_me || $data_system.battle_end_me
end
#--------------------------------------------------------------------------
# * Set Battle End ME
#--------------------------------------------------------------------------
def battle_end_me=(battle_end_me)
 @battle_end_me = battle_end_me
end
#--------------------------------------------------------------------------
# * Pre-Save Processing
#--------------------------------------------------------------------------
def on_before_save
 @save_count += 1
 @version_id = $data_system.version_id
 @frames_on_save = Graphics.frame_count
 @bgm_on_save = RPG::BGM.last
 @bgs_on_save = RPG::BGS.last
end
#--------------------------------------------------------------------------
# * Post-Load Processing
#--------------------------------------------------------------------------
def on_after_load
 Graphics.frame_count = @frames_on_save
 @bgm_on_save.play
 @bgs_on_save.play
end
#--------------------------------------------------------------------------
# * Get Play Time in Seconds
#--------------------------------------------------------------------------
def playtime
```

```
  Graphics.frame_count / Graphics.frame_rate
end
#--------------------------------------------------------------------------
# * Get Play Time in Character String
#--------------------------------------------------------------------------
def playtime_s
  hour = playtime / 60 / 60
  min = playtime / 60 % 60
  sec = playtime % 60
  sprintf("%02d:%02d:%02d", hour, min, sec)
end
#--------------------------------------------------------------------------
# * Save BGM
#--------------------------------------------------------------------------
def save_bgm
  @saved_bgm = RPG::BGM.last
end
#--------------------------------------------------------------------------
# * Resume BGM
#--------------------------------------------------------------------------
def replay_bgm
  @saved_bgm.replay if @saved_bgm
end
end
```

Source Code Holder sebagai Dice Roll

```
#==========================================================================
#
# Add-on: Holder Battlers
#
#==========================================================================

$imported = {} if $imported.nil?
$imported["BattleSymphony-HB"] = true

#==========================================================================
# ▼ Compatibility
# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
#
#==========================================================================

$imported = {} if $imported.nil?
$imported["BattleSymphony-HB"] = true

#==========================================================================
# ■ Direction - Advanced Configuration
#==========================================================================

module Direction

 #--------------------------------------------------------------------
 # self.index_hb
 #--------------------------------------------------------------------
 def self.index_hb(pose)
  case pose
  # array = [row, frames]
  # frames is optional, default is 15
  when :idle
   array = [0, 12]
  when :struck
   array = [3, 10]
  when :woozy
   array = [2, 12]
  #---
  when :victory
   array = [10]
  when :defend
   array = [1, 5]
  when :dead
   array = [12]
```

```
    #---
    when :attack
      array = [4, 4]
    when :skill
      array = [6, 6]
    when :magic
      array = [7, 6]
    when :item
      array = [5, 6]
    #---
    when :advance
      array = [8, 5]
    when :retreat
      array = [9, 5]
    else; array = [0, 12]
    end
    return array
  end

  #--------------------------------------------------------------------------
  # self.auto_pose_hb
  #--------------------------------------------------------------------------
  def self.auto_pose_hb(battler)
    return :dead if battler.dead?
    return :woozy if battler.hp < battler.mhp / 4
    return :idle
  end

end # Direction

#==============================================================================
# ■ BattleManager
#==============================================================================

module BattleManager

  #--------------------------------------------------------------------------
  # alias method: process_victory
  #--------------------------------------------------------------------------
  class <<self; alias bes_hb_process_victory process_victory; end
  def self.process_victory
    $game_party.alive_members.each { |battler| battler.force_pose_hb(:victory) }
    return bes_hb_process_victory
  end
```

```
 #--------------------------------------------------------------------------
 # alias method: process_defeat
 #--------------------------------------------------------------------------
 class <<self; alias bes_hb_process_defeat process_defeat; end
 def self.process_defeat
  $game_troop.alive_members.each { |battler| battler.force_pose_hb(:victory) }
  return bes_hb_process_defeat
 end

end # BattleManager

#==============================================================================
# ■ Regular Expression
#==============================================================================

module REGEXP
 module SYMPHONY

  HOLDERS_BATTLER = /<(?:HOLDERS_BATTLER|holders battler):[ ]*(.*)>/i

 end
end

#==============================================================================
# ■ DataManager
#==============================================================================

module DataManager

 #--------------------------------------------------------------------------
 # alias method: load_database
 #--------------------------------------------------------------------------
 class <<self; alias load_database_bes_hb load_database; end
 def self.load_database
  load_database_bes_hb
  load_notetags_bes_hb
 end

 #--------------------------------------------------------------------------
 # new method: load_notetags_bes_hb
 #--------------------------------------------------------------------------
 def self.load_notetags_bes_hb
  groups = [$data_actors, $data_enemies]
  groups.each { |group|
   group.each { |obj|
```

```
      next if obj.nil?
      obj.battle_symphony_holders_battler
    }
  }
 end

end # DataManager

#==============================================================================
# ■ RPG::BaseItem
#==============================================================================

class RPG::BaseItem

 #--------------------------------------------------------------------------
 # * Public Instance Variables
 #--------------------------------------------------------------------------
 attr_accessor :holders_name

 #--------------------------------------------------------------------------
 # new method: battle_symphony_holders_battler
 #--------------------------------------------------------------------------
 def battle_symphony_holders_battler
  self.note.split(/[\r\n]+/).each { |line|
   case line
   when REGEXP::SYMPHONY::HOLDERS_BATTLER
    @holders_name = $1.to_s
   end
  }
 end

end # RPG::BaseItem

#==============================================================================
# ■ Game_Battler
#==============================================================================

class Game_Battler < Game_BattlerBase

 #--------------------------------------------------------------------------
 # new method: use_hb?
 #--------------------------------------------------------------------------
 def use_hb?
  self.actor? ? !actor.holders_name.nil? : !enemy.holders_name.nil?
 end
```

```
#--------------------------------------------------------------------------
# new method: holders_name
#--------------------------------------------------------------------------
def holders_name
  self.actor? ? actor.holders_name : enemy.holders_name
end

#--------------------------------------------------------------------------
# alias method: set_default_position
#--------------------------------------------------------------------------
alias bes_hb_set_default_position set_default_position
def set_default_position
  bes_hb_set_default_position
  set_hb_default_position if self.use_hb?
end

#--------------------------------------------------------------------------
# new method: set_8d_default_position
#--------------------------------------------------------------------------
def set_hb_default_position
  self.pose = Direction.auto_pose_hb(self)
end

#--------------------------------------------------------------------------
# alias method: break_pose
#--------------------------------------------------------------------------
alias bes_hb_break_pose break_pose
def break_pose
  bes_hb_break_pose
  break_pose_hb if self.use_hb?
end

#--------------------------------------------------------------------------
# new method: break_pose_hb
#--------------------------------------------------------------------------
def break_pose_hb
  @pose = Direction.auto_pose_hb(self)
  #---
  return unless SceneManager.scene.spriteset
  return unless self.sprite
  @direction = SYMPHONY::View::PARTY_DIRECTION
  @direction = Direction.opposite(@direction) if self.enemy?
  self.sprite.mirror = [9, 6, 3].include?(@direction)
  #@direction = Direction.opposite(@direction) if self.sprite.mirror
```

```
  end

  #--------------------------------------------------------------------------
  # new method: force_pose_hb
  #--------------------------------------------------------------------------
  def force_pose_hb(pose)
    return unless self.use_hb?
    return unless self.exist?
    #---
    self.break_pose
    self.pose = pose
    @force_pose = true
  end

end # Game_Battler

#==============================================================================
# ■ Game_Actor
#==============================================================================

class Game_Actor < Game_Battler

  #--------------------------------------------------------------------------
  # alias method: use_charset?
  #--------------------------------------------------------------------------
  alias bes_hb_use_charset? use_charset?
  def use_charset?
    return false if use_hb?
    return bes_hb_use_charset?
  end

end # Game_Actor

#==============================================================================
# ■ Game_Enemy
#==============================================================================

class Game_Enemy < Game_Battler

  #--------------------------------------------------------------------------
  # alias method: use_charset?
  #--------------------------------------------------------------------------
  alias bes_hb_use_charset? use_charset?
  def use_charset?
    return false if use_hb?
```

```
    return bes_hb_use_charset?
  end

end # Game_Enemy

#==============================================================================
# ■ Sprite_Battler
#==============================================================================

class Sprite_Battler < Sprite_Base

  #--------------------------------------------------------------------------
  # alias method: update_bitmap
  #--------------------------------------------------------------------------
  alias bes_hb_update_bitmap update_bitmap
  def update_bitmap
    correct_change_pose if @timer.nil?
    @battler.use_hb? ? update_hbset : bes_hb_update_bitmap
  end

  #--------------------------------------------------------------------------
  # alias method: update_origin
  #--------------------------------------------------------------------------
  alias bes_hb_update_origin update_origin
  def update_origin
    bes_hb_update_origin
    update_origin_hb if @battler.use_hb?
  end

  #--------------------------------------------------------------------------
  # new method: update_charset
  #--------------------------------------------------------------------------
  def update_hbset
    @battler.set_default_position unless pose
    #--
    update_hbset_bitmap
    update_src_rect
  end

  #--------------------------------------------------------------------------
  # alias method: correct_change_pose
  #--------------------------------------------------------------------------
  alias bes_hb_correct_change_pose correct_change_pose
  def correct_change_pose
    bes_hb_correct_change_pose unless @battler.use_hb?
```

```
#--------------------------------------------------------------------------
def update_hbset_bitmap
 if hb_graphic_changed?
  @character_name = @battler.holders_name
  set_character_bitmap
 end
end

#--------------------------------------------------------------------------
# alias method: update_src_rect
#--------------------------------------------------------------------------
alias bes_hb_update_src_rect update_src_rect
def update_src_rect
 bes_hb_update_src_rect unless @battler.use_hb?
 return unless @battler.use_hb?
 @timer -= 1
 if @battler.force_pose
  if @timer <= 0 && @pattern < 3
   array = []
   array = Direction.index_hb(pose)
   @pattern += 1
   @timer = array[1].nil? ? 15 : array[1]
  end
 else
  if @timer <= 0
   @pattern += 1
   @pattern = 0 if @pattern > 3
   @timer = 15
  end
 end
 #--
 line_no = Direction.index_hb(pose)[0]
 sx = @pattern * @cw
 sy = line_no * @ch
 self.src_rect.set(sx, sy, @cw, @ch)
end

end # Sprite_Battler

#==============================================================================
#
# END OF FILE
#
#==============================================================================
```

```
  correct_change_pose_hb if @battler.use_hb?
end

#--------------------------------------------------------------------------
# new method: correct_change_pose_hb
#--------------------------------------------------------------------------
def correct_change_pose_hb
  array = Direction.index_hb(pose)
  @pattern = @battler.reverse_pose ? 3 : 0
  @timer = array[1].nil? ? 15 : array[1]
  @last_pose = pose
  @back_step = false
end

#--------------------------------------------------------------------------
# new method: update_charset_origin
#--------------------------------------------------------------------------
def update_origin_hb
  if bitmap
    self.ox = @cw / 2
    self.oy = @ch
  end
end

#--------------------------------------------------------------------------
# new method: hb_graphic_changed?
#--------------------------------------------------------------------------
def hb_graphic_changed?
  self.bitmap.nil? || @character_name != @battler.holders_name
end

#--------------------------------------------------------------------------
# alias method: set_character_bitmap
#--------------------------------------------------------------------------
alias bes_hb_set_character_bitmap set_character_bitmap
def set_character_bitmap
  bes_hb_set_character_bitmap unless @battler.use_hb?
  return unless @battler.use_hb?
  self.bitmap = Cache.character(@character_name)
  @cw = bitmap.width / 4
  @ch = bitmap.height / 14
end

#--------------------------------------------------------------------------
# new method: update_hbset_bitmap
```

Source Code untuk Sprite Karakter

```
#==============================================================================
# ** Sprite_Character
#==============================================================================

class Sprite_Character < Sprite_Base
 #--------------------------------------------------------------------------
 # * Public Instance Variables
 #--------------------------------------------------------------------------
 attr_accessor :character
 #--------------------------------------------------------------------------
 # * Object Initialization
 #    character : Game_Character
 #--------------------------------------------------------------------------
 def initialize(viewport, character = nil)
  super(viewport)
  @character = character
  @balloon_duration = 0
  update
 end
 #--------------------------------------------------------------------------
 # * Free
 #--------------------------------------------------------------------------
 def dispose
  end_animation
  end_balloon
  super
 end
 #--------------------------------------------------------------------------
 # * Frame Update
 #--------------------------------------------------------------------------
 def update
  super
  update_bitmap
  update_src_rect
  update_position
  update_other
  update_balloon
  setup_new_effect
 end
 #--------------------------------------------------------------------------
 # * Get Tileset Image That Includes the Designated Tile
```

```
#--------------------------------------------------------------------------
def tileset_bitmap(tile_id)
 Cache.tileset($game_map.tileset.tileset_names[5 + tile_id / 256])
end
#--------------------------------------------------------------------------
# * Update Transfer Origin Bitmap
#--------------------------------------------------------------------------
def update_bitmap
 if graphic_changed?
  @tile_id = @character.tile_id
  @character_name = @character.character_name
  @character_index = @character.character_index
  if @tile_id > 0
   set_tile_bitmap
  else
   set_character_bitmap
  end
 end
end
#--------------------------------------------------------------------------
# * Determine if Graphic Changed
#--------------------------------------------------------------------------
def graphic_changed?
 @tile_id != @character.tile_id ||
 @character_name != @character.character_name ||
 @character_index != @character.character_index
end
#--------------------------------------------------------------------------
# * Set Tile Bitmap
#--------------------------------------------------------------------------
def set_tile_bitmap
 sx = (@tile_id / 128 % 2 * 8 + @tile_id % 8) * 32;
 sy = @tile_id % 256 / 8 % 16 * 32;
 self.bitmap = tileset_bitmap(@tile_id)
 self.src_rect.set(sx, sy, 32, 32)
 self.ox = 16
 self.oy = 32
end
#--------------------------------------------------------------------------
# * Set Character Bitmap
#--------------------------------------------------------------------------
def set_character_bitmap
 self.bitmap = Cache.character(@character_name)
```

```
  sign = @character_name[/^[\!\$]./]
  if sign && sign.include?('$')
    @cw = bitmap.width / 3
    @ch = bitmap.height / 4
  else
    @cw = bitmap.width / 12
    @ch = bitmap.height / 8
  end
  self.ox = @cw / 2
  self.oy = @ch
end
#--------------------------------------------------------------------------
# * Update Transfer Origin Rectangle
#--------------------------------------------------------------------------
def update_src_rect
  if @tile_id == 0
    index = @character.character_index
    pattern = @character.pattern < 3 ? @character.pattern : 1
    sx = (index % 4 * 3 + pattern) * @cw
    sy = (index / 4 * 4 + (@character.direction - 2) / 2) * @ch
    self.src_rect.set(sx, sy, @cw, @ch)
  end
end
#--------------------------------------------------------------------------
# * Update Position
#--------------------------------------------------------------------------
def update_position
  move_animation(@character.screen_x - x, @character.screen_y - y)
  self.x = @character.screen_x
  self.y = @character.screen_y
  self.z = @character.screen_z
end
#--------------------------------------------------------------------------
# * Update Other
#--------------------------------------------------------------------------
def update_other
  self.opacity = @character.opacity
  self.blend_type = @character.blend_type
  self.bush_depth = @character.bush_depth
  self.visible = !@character.transparent
end
#--------------------------------------------------------------------------
# * Set New Effect
```

```
  #--------------------------------------------------------------------------
  def setup_new_effect
    if !animation? && @character.animation_id > 0
      animation = $data_animations[@character.animation_id]
      start_animation(animation)
    end
    if !@balloon_sprite && @character.balloon_id > 0
      @balloon_id = @character.balloon_id
      start_balloon
    end
  end
  #--------------------------------------------------------------------------
  # * Move Animation
  #--------------------------------------------------------------------------
  def move_animation(dx, dy)
    if @animation && @animation.position != 3
      @ani_ox += dx
      @ani_oy += dy
      @ani_sprites.each do |sprite|
        sprite.x += dx
        sprite.y += dy
      end
    end
  end
  #--------------------------------------------------------------------------
  # * End Animation
  #--------------------------------------------------------------------------
  def end_animation
    super
    @character.animation_id = 0
  end
  #--------------------------------------------------------------------------
  # * Start Balloon Icon Display
  #--------------------------------------------------------------------------
  def start_balloon
    dispose_balloon
    @balloon_duration = 8 * balloon_speed + balloon_wait
    @balloon_sprite = ::Sprite.new(viewport)
    @balloon_sprite.bitmap = Cache.system("Balloon")
    @balloon_sprite.ox = 16
    @balloon_sprite.oy = 32
    update_balloon
  end
```

```
#--------------------------------------------------------------------------
# * Free Balloon Icon
#--------------------------------------------------------------------------
def dispose_balloon
 if @balloon_sprite
  @balloon_sprite.dispose
  @balloon_sprite = nil
 end
end
#--------------------------------------------------------------------------
# * Update Balloon Icon
#--------------------------------------------------------------------------
def update_balloon
 if @balloon_duration > 0
  @balloon_duration -= 1
  if @balloon_duration > 0
   @balloon_sprite.x = x
   @balloon_sprite.y = y - height
   @balloon_sprite.z = z + 200
   sx = balloon_frame_index * 32
   sy = (@balloon_id - 1) * 32
   @balloon_sprite.src_rect.set(sx, sy, 32, 32)
  else
   end_balloon
  end
 end
end
#--------------------------------------------------------------------------
# * End Balloon Icon
#--------------------------------------------------------------------------
def end_balloon
 dispose_balloon
 @character.balloon_id = 0
end
#--------------------------------------------------------------------------
# * Balloon Icon Display Speed
#--------------------------------------------------------------------------
def balloon_speed
 return 8
end
#--------------------------------------------------------------------------
# * Wait Time for Last Frame of Balloon
#--------------------------------------------------------------------------
```

```
  def balloon_wait
    return 12
  end
  #--------------------------------------------------------------------------
  # * Frame Number of Balloon Icon
  #--------------------------------------------------------------------------
  def balloon_frame_index
    return 7 - [(@balloon_duration - balloon_wait) / balloon_speed, 0].max
  end
end
```

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

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama : Mukhammad Dian Amirul Azmi
NIM : 0918065
Jurusan : Teknik Informatika S-1
Judul : Rancang Bangun Game RPG SWORD MASTER OF DESTINY Dengan Metode Tactical Battle

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :
Hari : Kamis
Tanggal : 25 Agustus 2014

PanitiaUjianSkripsi :

KetuaMajelisPenguji

Joseph DedyIrawan, ST, MT
NIP.19740416200501100 2

AnggotaPenguji :

PengujiPertama

ALI MAHMUDI, B.Eng. Phd
NIP. P. 1031000429

PengujiKedua

YOSEP AGUS PRANOTO, ST. MT
NIP. P. 1031000432

LEMBAR KEASLIAN

PROGRAM STUDI TEKNIK INFORMATIKA S1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL
MALANG

LEMBAR PERNYATAAN KEASLIAN

Saya yang bertandatangan di bawah ini :

Nama : MUKHAMMAD DIAN AMIRUL AZMI

NIM : 09.18.065

Program Studi : Teknik Informatika S-1

Fakultas : TeknologiIndustri

Menyatakan bahwa skripsi yang berjudul :

"RANCANG BANGUN GAME RPG SWORD MASTER OF DESTINY DENGAN METODE TACTICAL BATTLE"

Adalah skripsi saya sendiri, bukan duplikat, dan seluruh sumber yang dikutip maupun dirujuk telah saya nyatakan dengan sebenarnya.

Malang, Agustus2014

Yang membuat pernyataan

METERAI TEMPEL
TEL71ADF569684126
6000
ENAM RIBU RUPIAH

MUKHAMMAD DIAN AMIRUL AZMI

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura-Gura No. 2
MALANG

Malang, 21 Oktober 2013

Lampiran : 1(Satu) berkas

Perihal : Kesediaan sebagai Pembimbing Skripsi

Kepada : Yth. Bpk/Ibu **Ir. Yusuf Ismail Nakhoda, MT**
Dosen Pembina Prodi Teknik Informatika S-1
Institut Teknologi Nasional
MALANG

Yang bertanda tangan dibawah ini:

Nama : MUKHAMMAD DIAN AMIRUL AZMI
Nim : 0918065
Prodi : Teknik Informatika S-1

Dengan ini mengajukan permohonan, kiranya bapak bersedia menjadi Dosen Pembimbing Utama / ~~Pendamping~~ *), untuk penyusunan Skripsi dengan judul (Proposal Terlampir) :

Rancang Bangun Game RPG (Role Play Game) Sword Master Of Destiny Dengan Metode Tactical Battle

Adapun tugas tersebut sebagai salah satu syarat untuk menempuh Ujian Akhir Sarjana Teknik.
Demikian permohonan kami dan atas kesediaan bapak kami sampaikan terima kasih.

Prodi T. Informatika S-1
K e t u a,

Joseph Dedy Irawan,ST., MT.
NIP : 1974041620050210002

Hormat Kami,

MUKHAMMAD DIAN AMIRUL AZMI

Form S-3a

INSTITUT TEKNOLOGI NASIONAL
Jln. Bendungan Sigura-gura No. 2
Jln. Raya Karanglo Km2
M A L A N G

PERNYATAAN KESEDIAAN DALAM PEMBIMBINGAN SKRIPSI

Sesuai permohonan dari mahasiswa/i :

Nama : MUKHAMMAD DIAN AMIRUL AZMI

Nim : 0918065

Program Studi : Teknik Informatika

Dengan ini menyatakan bersedia / tidak bersedia *) membimbing skripsi dari mahasiswa tersebut dengan judul :

Rancang Bangun Game RPG (Role Play Game) Sword Master Of Destiny Dengan Metode Tactical Battle

Demikian Surat Pernyataan ini kami buat agar dipergunakan seperlunya.

Malang, _____

Hormat Kami,

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y.1018800189

Catatan :
Setelah disetujui agar formulir ini diserahkan mahasiswa/i yg bersangkutan kepada Jurusan untuk diproses lebih lanjut
*) coret yang tidak perlu

Form S-3b

INSTITUT TEKNOLOGI NASIONAL
Jln. Bendungan Sigura-gura No. 2
Jln. Raya Karanglo Km2
M A L A N G

PERNYATAAN KESEDIAAN DALAM PEMBIMBINGAN SKRIPSI

Sesuai permohonan dari mahasiswa/i :

Nama : MUKHAMMAD DIAN AMIRUL AZMI

Nim : 0918065

Program Studi : Teknik Informatika

Dengan ini menyatakan bersedia / tidak bersedia *) membimbing skripsi dari mahasiswa tersebut dengan judul :

Rancang Bangun Game RPG (Role Play Game) Sword Master Of Destiny Dengan Metode Tactical Battle

Demikian Surat Pernyataan ini kami buat agar dipergunakan seperlunya.

Malang, _____

Hormat Kami,

Nurlaily Vendyansyah, ST

Catatan :
Setelah disetujui agar formulir ini diserahkan mahasiswa/i yg bersangkutan kepada Jurusan untuk diproses lebih lanjut
*) coret yang tidak perlu

Form S-3b

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura-Gura No. 2
MALANG

Malang, 21 Oktober 2013

Lampiran : 1(Satu) berkas

Perihal : Kesediaan sebagai Pembimbing Skripsi

Kepada : Yth. Bpk/Ibu **Nurlaily Vendyansyah, ST**
Dosen Pembina Prodi Teknik Informatika S-1
Institut Teknologi Nasional
MALANG

Yang bertanda tangan dibawah ini:

Nama : MUKHAMMAD DIAN AMIRUL AZMI
Nim : 0918065
Prodi : Teknik Informatika S-1

Dengan ini mengajukan permohonan, kiranya Bapak/Ibu bersedia menjadi Dosen Pembimbing ~~Utama~~ / Pendamping *), untuk penyusunan Skripsi dengan judul (Proposal Terlampir) :

Rancang Bangun Game RPG (Role Play Game) Sword Master Of Destiny Dengan Metode Tactical Battle

Adapun tugas tersebut sebagai salah satu syarat untuk menempuh Ujian Akhir Sarjana Teknik. Demikian permohonan kami dan atas kesediaan bapak kami sampaikan terima kasih.

Prodi T. Informatika S-1
K e t u a,

Joseph Dedy Irawan,ST., MT.
NIP : 1974041620050210002

Hormat Kami,

MUKHAMMAD DIAN AMIRUL AZMI

Form S 3a

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

I. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 21 Oktober 2013

Nomor : ITN-70/.INF/TA/2013
Lampiran : ---
Perihal : *Bimbingan Skripsi*

Kepada : Yth. Bpk/Ibu **Ir. Yusuf Ismail Nakhoda, MT**
Dosen Pembina Program Studi Teknik Informatika S-1
Institut Teknologi Nasional
Malang

Dengan Hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal skripsi untuk mahasiswa :

Nama : MUKHAMMAD DIAN AMIRUL AZMI
Nim : 0918065
Prodi : Teknik Informatika S-1
Fakultas : Teknologi Industri

Maka dengan ini pembimbingan kami serahkan sepenuhnya kepada Saudara/i selama waktu 6 (enam) bulan, terhitung mulai tanggal :

21 OKTOBER 2013 S/D 21 MARET 2014

Sebagai satu syarat untuk menempuh Ujian Akhir Sarjana Teknik, Program Studi Teknik Informatika S-1.
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Program Studi Teknik Informatika S-1
K e t u a,

Joseph Dedy Irawan,ST., MT.
NIP. 1974041620050210002

Form S-4a

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

T. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 21 Oktober 2013

Nomor : ITN-70/.INF/TA/2013
Lampiran : ---
Perihal : *Bimbingan Skripsi*

Kepada : Yth. Bpk/Ibu **Ir. Yusuf Ismail Nakhoda, MT**
Dosen Pembina Program Studi Teknik Informatika S-1
Institut Teknologi Nasional
Malang

Dengan Hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal skripsi untuk mahasiswa :

Nama : MUKHAMMAD DIAN AMIRUL AZMI
Nim : 0918065
Prodi : Teknik Informatika S-1
Fakultas : Teknologi Industri

Maka dengan ini pembimbingan kami serahkan sepenuhnya kepada Saudara/i selama waktu 6 (enam) bulan, terhitung mulai tanggal :

21 OKTOBER 2013 S/D 21 MARET 2014

Sebagai satu syarat untuk menempuh Ujian Akhir Sarjana Teknik, Program Studi Teknik Informatika S-1.
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Program Studi Teknik Informatika S-1
K e t u a,

Joseph Dedy Irawan,ST., MT.
NIP. 197404162005021002

Form S-4a

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Industri
Program Studi Teknik Informatika S1

(3)

LEMBAR PEMANTAUAN SEMINAR PROGRES SKRIPSI

: Muhhammad Dian A.A.
: 09.18.065
:

Skripsi :
ang Bangun Game RPG Sword Master of Destiny
Metode Tactical Battle System

/ Komentar :

Game tidak dibawa →
Screen shot : Tampilan awal, Start permainan, petu
ran : lengkapi teori dengan penjelasan judul
"Tactical Battle" System.
↳ Muncul di ~~gak~~ Implementasi game atau tidak?

etahui,
m Studi T.Informatika
Ketua

Pembimbing 1

Pembimbing 2

Dedy Irawan, ST.MT
740416 200501 1 002

(.....) (.....)

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Industri
Program Studi Teknik Informatika S1

FORMULIR PERBAIKAN UJIAN SKRIPSI

Dalam pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Informatika, maka perlu adanya perbaikan untuk mahasiswa :

Nama : M. Dian Aminuf Azmi
NIM : 09.17.065
Perbaikan Meliputi :

1) Kaitkan Landasan Teori dgn daftar pustaka.
2) Terlalu rapat. Bab 2 3
3) Perbaiki penomoran gambar lah ada gambar
   4.9, 4.10 → hal 56
   4.22, 4.24
4) Tambahkan Daftar Pustaka.
5) Tambahkan Scoring

Malang, _____

(_____)

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Industri
Program Studi Teknik Informatika S1

FORMULIR PERBAIKAN UJIAN SKRIPSI

Dalam pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Informatika, maka perlu adanya perbaikan untuk mahasiswa :

Nama : Mukhammad Dian Amirul Azmi
NIM : 09.18.065
Perbaikan Meliputi :

1. Daftar Isi + gambar
2. Standarkan penulisan → spasi, paragraf, penomoran, font bab dan subbab
3. perbaiki Struktur Game
4. Nama tabel perlu Icon
5. Isian ke daftar pustaka (tambahkan daftar pustaka)
6. Perbaiki Bab 4 dan Bab 5

Malang, 25 - 9 - 2014

(Yosep Agus P)

KUISIONER

Nama :

Nim / Nip :

Pekerjaan :

Berilah Keterangan Pada kolom nilai yang menurut anda sesuai dengan penilaian aplikasi **"RANCANG BANGUN GAME RPG SWORD MASTER OF DESTINY DENGAN METODE TACTICAL BATTLE"**

No		Kolom Penilaian Responden			
		SB	B	C	K
1	Layout antar muka				
2	Jalan cerita game				
3	Kinerja sistem game				
4	Tingkat kemudahan				

Penilaian berdasarkan kriteria sebagai berikut :

a. SB : Sangat Baik
b. B : Baik
c. C : Cukup
d. K : Kurang

Saran dan kritik :

_____

_____

_____

KUISIONER

Nama :

Nim / Nip :

Pekerjaan :

Berilah Keterangan Pada kolom nilai yang menurut anda sesuai dengan penilaian aplikasi **"RANCANG BANGUN GAME RPG SWORD MASTER OF DESTINY DENGAN METODE TACTICAL BATTLE"**

No		Kolom Penilaian Responden			
		SB	B	C	K
1	Layout antar muka				
2	Jalan cerita game				
3	Kinerja sistem game				
4	Tingkat kemudahan				

Penilaian berdasarkan kriteria sebagai berikut :

a. SB : Sangat Baik
b. B : Baik
c. C : Cukup
d. K : Kurang

Saran dan kritik :

_____

_____

_____

KUISIONER

Nama :

Nim / Nip :

Pekerjaan :

Berilah Keterangan Pada kolom nilai yang menurut anda sesuai dengan penilaian aplikasi **"RANCANG BANGUN GAME RPG SWORD MASTER OF DESTINY DENGAN METODE TACTICAL BATTLE"**

No		Kolom Penilaian Responden			
		SB	B	C	K
1	Layout antar muka				
2	Jalan cerita game				
3	Kinerja sistem game				
4	Tingkat kemudahan				

Penilaian berdasarkan kriteria sebagai berikut :

a. SB : Sangat Baik
b. B : Baik
c. C : Cukup
d. K : Kurang

Saran dan kritik :

_____

_____

_____

KUISIONER

Nama :

Nim / Nip :

Pekerjaan :

Berilah Keterangan Pada kolom nilai yang menurut anda sesuai dengan penilaian aplikasi **"RANCANG BANGUN GAME RPG SWORD MASTER OF DESTINY DENGAN METODE TACTICAL BATTLE"**

No		Kolom Penilaian Responden			
		SB	B	C	K
1	Layout antar muka				
2	Jalan cerita game				
3	Kinerja sistem game				
4	Tingkat kemudahan				

Penilaian berdasarkan kriteria sebagai berikut :

a. SB : Sangat Baik
b. B : Baik
c. C : Cukup
d. K : Kurang

Saran dan kritik :

_____

_____

_____

KUISIONER

Nama :

Nim / Nip :

Pekerjaan :

Berilah Keterangan Pada kolom nilai yang menurut anda sesuai dengan penilaian aplikasi **"RANCANG BANGUN GAME RPG SWORD MASTER OF DESTINY DENGAN METODE TACTICAL BATTLE"**

No		Kolom Penilaian Responden			
		SB	B	C	K
1	Layout antar muka				
2	Jalan cerita game				
3	Kinerja sistem game				
4	Tingkat kemudahan				

Penilaian berdasarkan kriteria sebagai berikut :

a. SB : Sangat Baik
b. B : Baik
c. C : Cukup
d. K : Kurang

Saran dan kritik :

_____

_____

_____