BAB V

IMPLEMENTASI KARYA

Pada bab ini membahas tentang bagaimana penerapan rancangan pemrograman terhadap pengembangan *game tower defense* ini. Berikut merupakan penjelasan secara rincian tahap produksi dan tahap pasca produksi.

5.1 Produksi

Tahap produksi memiliki beberapa pemrograman yang penting untuk dikerjakan. Pemrograman ini mempunyai keterkaitan satu sama lain, apabila salah satu tidak ada maka *game* tidak akan berjalan dengan baik. Pemrograman menggunakan program *Game Maker Studio Professional.* Berikut adalah pemrograman dari pembuatan *game tower defense* ini sebagai berikut:

5.1.1 Pembuatan *Assets*

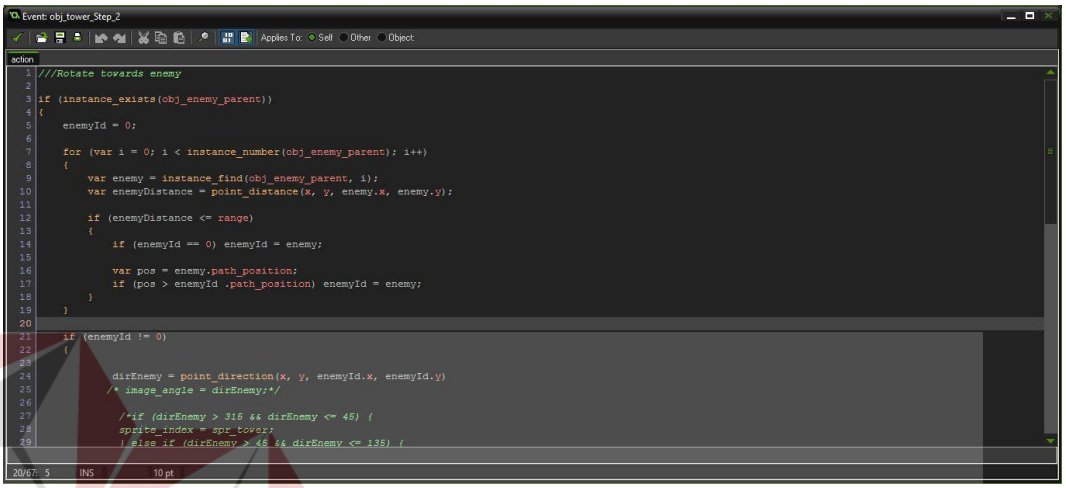
Game tower defense ini memerlukan *assets* agar *game* bisa dimainkan dengan baik. *Assets* yang berada di dalam *game* ini terdiri dari:

1. ***Character Programming***

   a. *Character players*

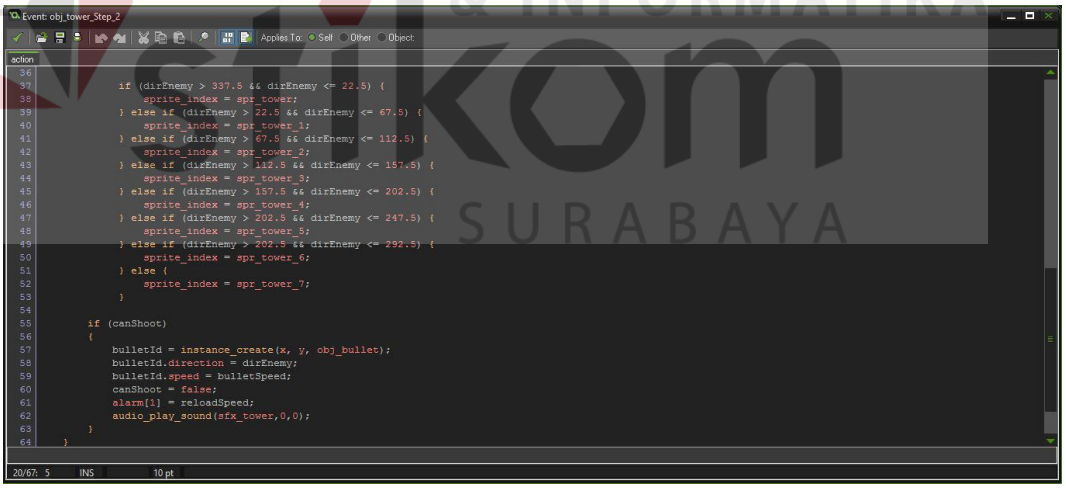
      Pemrograman karakter meluputi penentuan variabel *health point,* harga pembelian tower, kekuatan serangan, *upgrade character* dan rotasi *tower.* Karakter *tower* dibagi menjadi 3 bagian sesuai dengan senjata yang digunakan yaitu bambu runcing, senapan mesin dan senapan jarak jauh. Setiap karakter memiliki kemampuan masing-masing sehingga pemain

dapat memilih sesuai strategi yang digunakan. Berikut gambar pemrograman karakter yang sudah dibuat.



Gambar 5.1 Pemrograman Rotasi Karakter

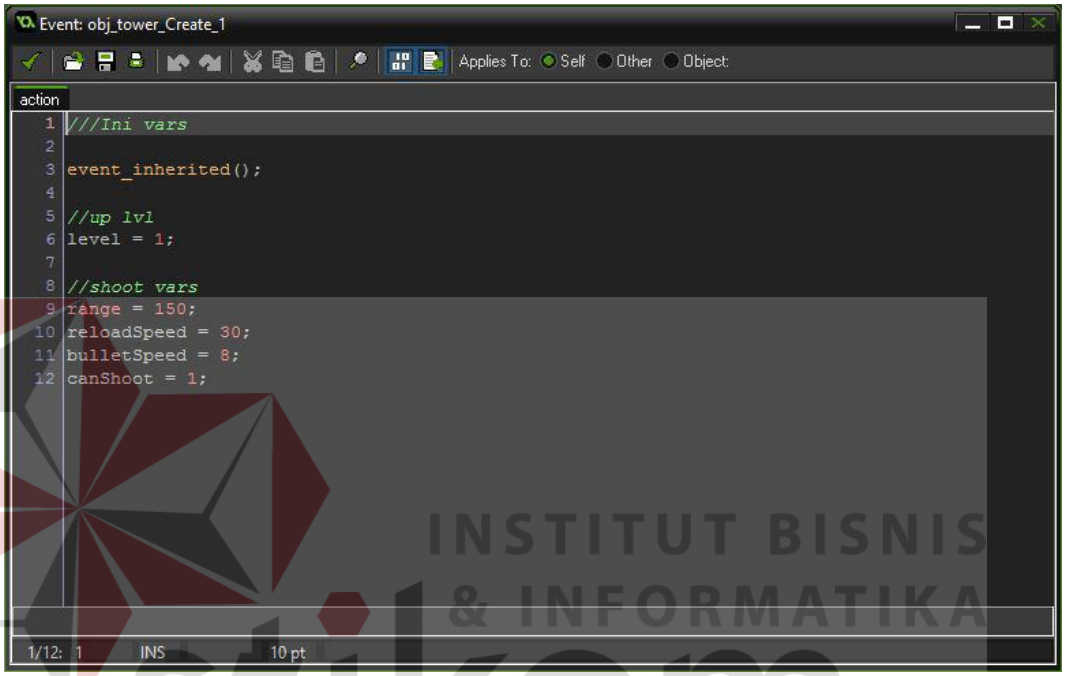
(Sumber : Olahan Penulis)



Gambar 5.2 Pemrograman Rotasi Karakter

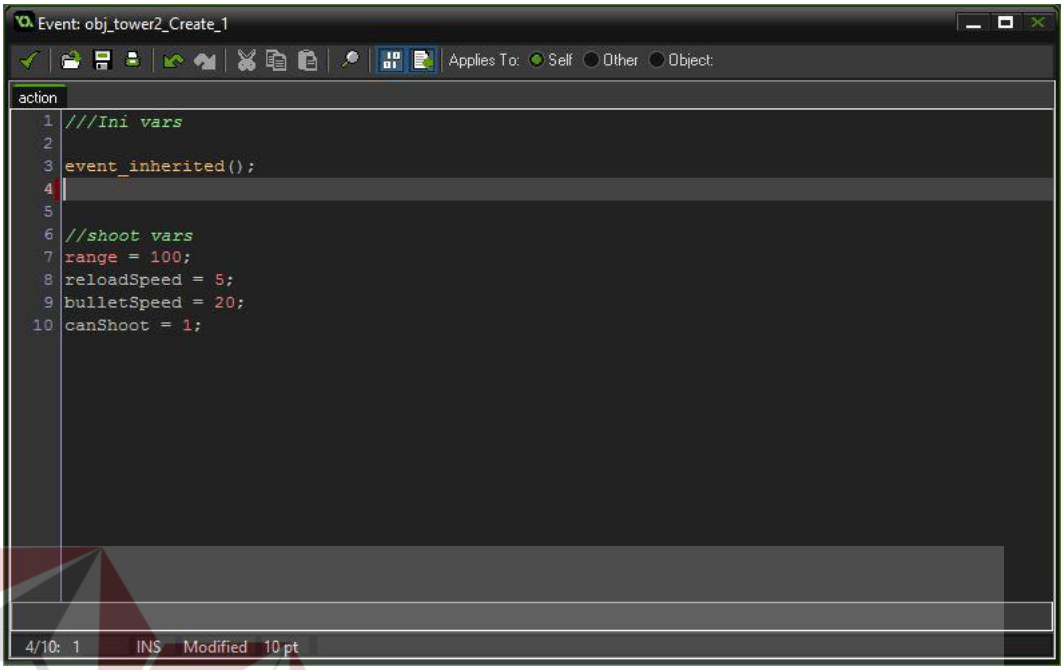
(Sumber : Olahan Penulis)

Gambar 5.2 adalah proses perancangan rotasi karakter *tower* yang berfungsi untuk membuat karakter *tower* secara otomatis menghadap kearah *enemy* yang lewat dan kemudian menyerang secara otomatis.



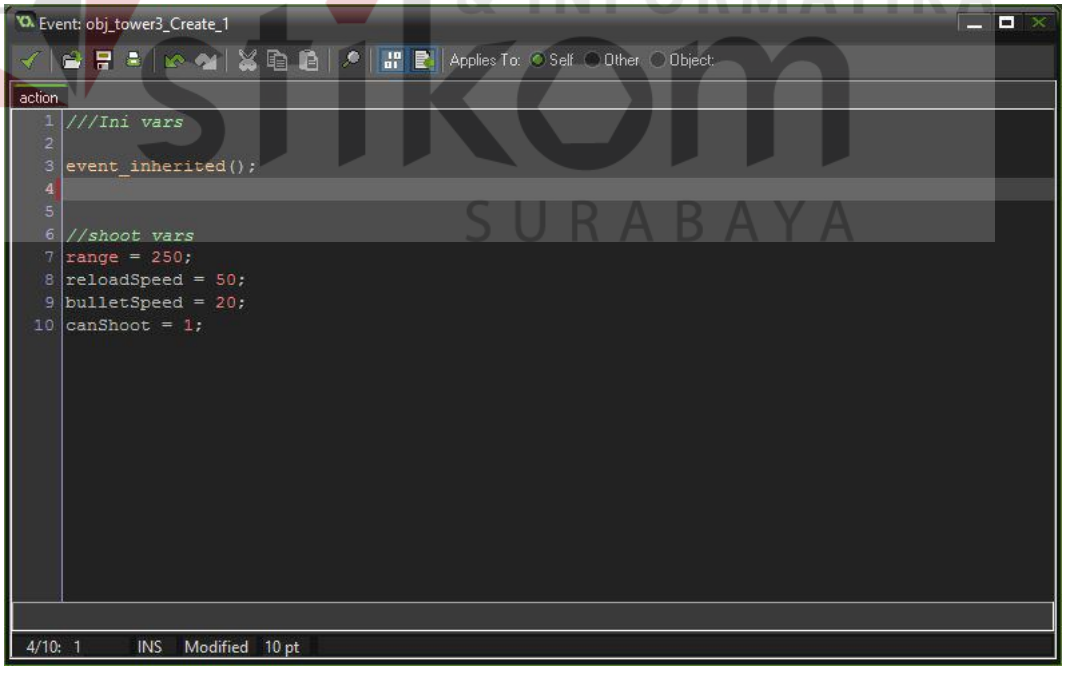
Gambar 5.3 Pemrograman Variabel Karakter Bambu Runcing

(Sumber : Olahan Penulis)



Gambar 5.4 Pemrograman Variabel Karakter Senapan Mesin

(Sumber : Olahan Penulis)

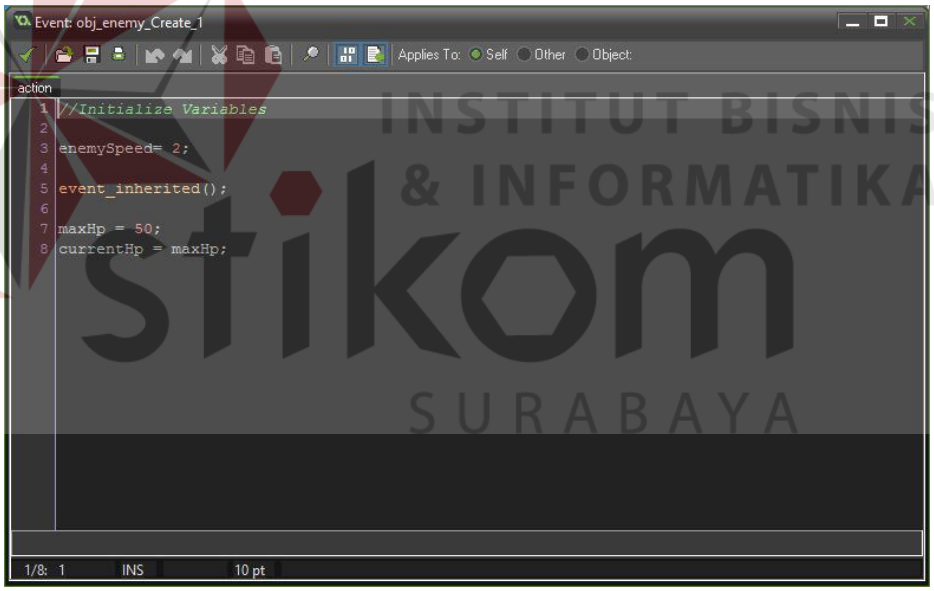


Gambar 5.5 Pemrograman Variabel Karakter Senapan Jauh

(Sumber : Olahan Penulis)

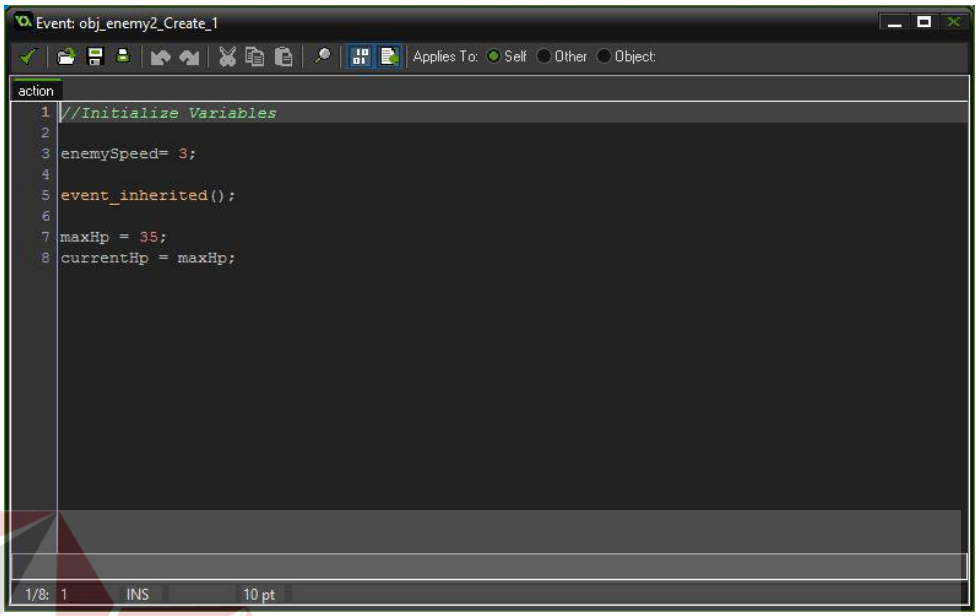
b. *Character Enemy*

Pemrograman *enemy* meluputi penentuan variabel *health point* dan kecepatan bergerak. Karakter *enemy* dibagi menjadi 3 *normal enemy* dan 2 *enemy boss*. Perbedaannya adalah *normal enemy* memiliki *health point* yang rendah dan kecepatan bergerak yang bervariasi sedangkan *enemy boss* memiliki *health point* yang tinggi namun kecepatan bergerak yang pelan. Karakter ini nantinya akan berjalan menyusuri jalan yang telah ditentukan menggunakan *path finding* dan menyerang benteng pertahanan pemain. Berikut gambar pemrograman karakter *enemy* yang sudah dibuat.



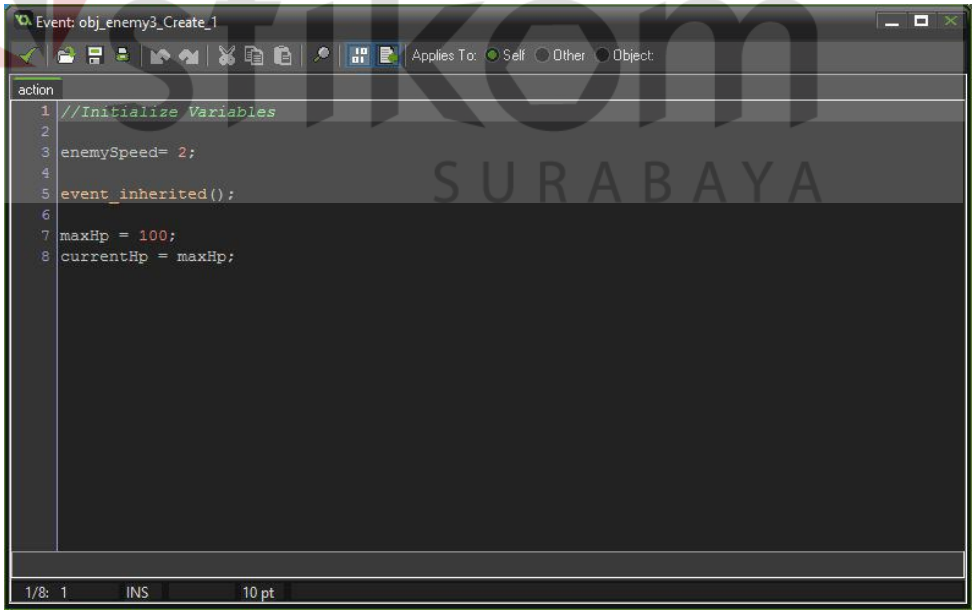
Gambar 5.6 Pemrograman Variabel Karakter *Enemy* 1

(Sumber : Olahan Penulis)



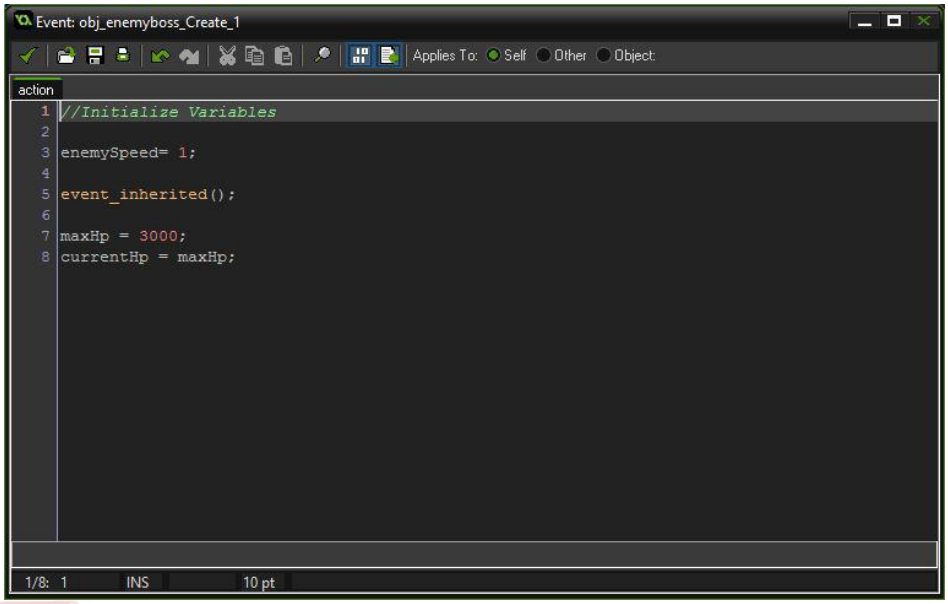
Gambar 5.7 Pemrograman Variabel Karakter *Enemy* 2

(Sumber : Olahan Penulis)



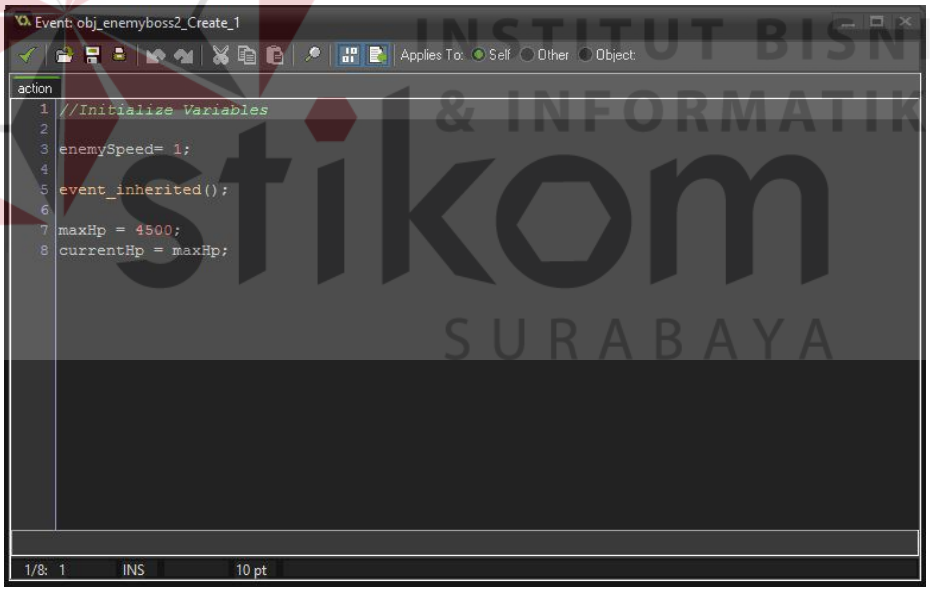
Gambar 5.8 Pemrograman Variabel Karakter *Enemy 3*

(Sumber : Olahan Penulis)



Gambar 5.9 Pemrograman Variabel Karakter *Enemy Boss 1*

(Sumber : Olahan Penulis)

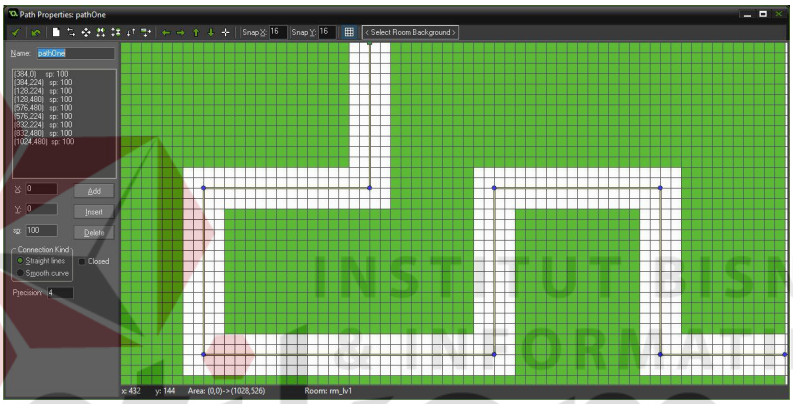


Gambar 5.10 Pemrograman Variabel Karakter *Enemy Boss 2*

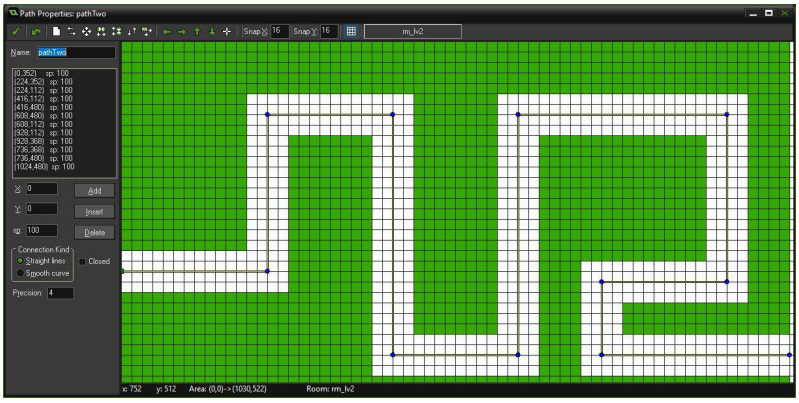
(Sumber : Olahan Penulis)

2. *Path Finding*

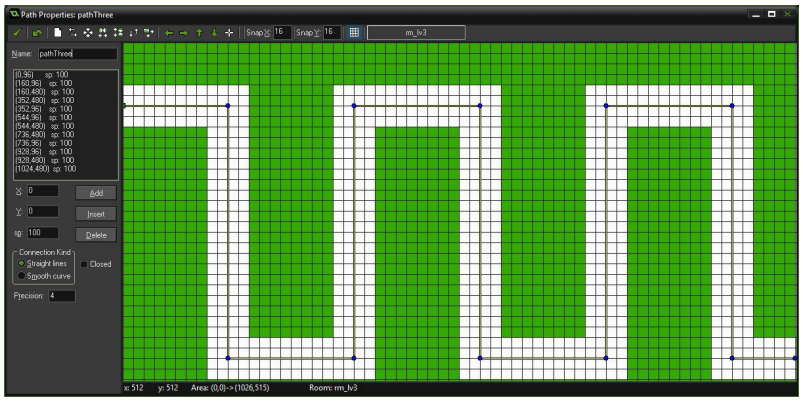
Pembuatan *path finding* pada *game* ini bertujuan untuk membuat *enemy* bergerak melawati jalur yang telah ditentukan. Berikut pembuatan *path finding* di setiap *level* akan berbeda mengikuti desain *map* sehingga tampak tidak monoton.



Gambar 5.11 Pemrograman *Path Finding Level* 1
(Sumber : Olahan Penulis)



Gambar 5.12 Pemrograman *Path Finding Level* 2
(Sumber : Olahan Penulis)



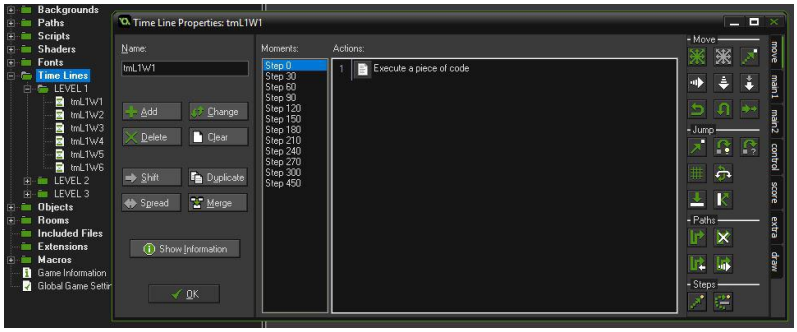
Gambar 5.13 Pemrograman *Path Finding Level* 3

(Sumber : Olahan Penulis)

Setiap *level* memiliki perbedaan pada dimana semakin panjang jalur *path finding* maka semakin banyak *enemy* yang bermunculan.

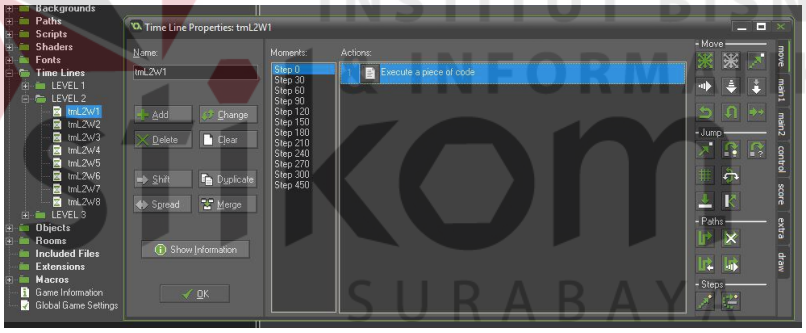
3. Wave System

Pembuatan *wave system* berguna untuk menjalankan pasukan *enemy* yang akan bergerak mengikuti *path finding* yang telah ditentukan. Setiap *level* memiliki total *wave* yang berbeda-beda. Pada *level* 1 memiliki 6 *wave* sedangkan *level* 2 memiliki 8 *wave* dan pada *level* 3 memiliki 10 *wave*. Pemrograman *wave system* menggunakan *tools* yang telah disediakan oleh *Game Maker Studio Professional* yaitu *timeline.* Berikut adalah gambar pemrograman *wave system* menggunakan *timeline.*



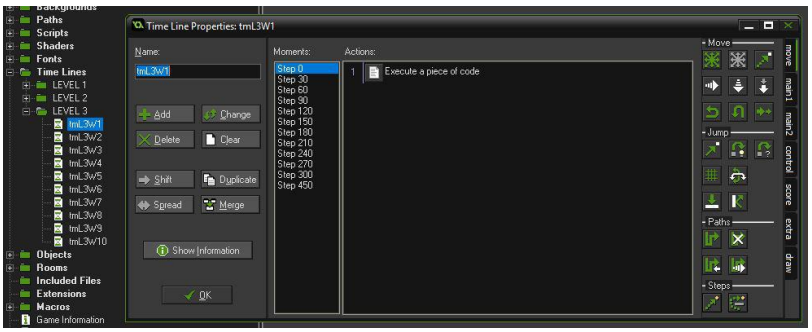
Gambar 5.14 Pemrograman *Wave Level 1*

(Sumber : Olahan Penulis)



Gambar 5.15 Pemrograman *Wave Level 2*

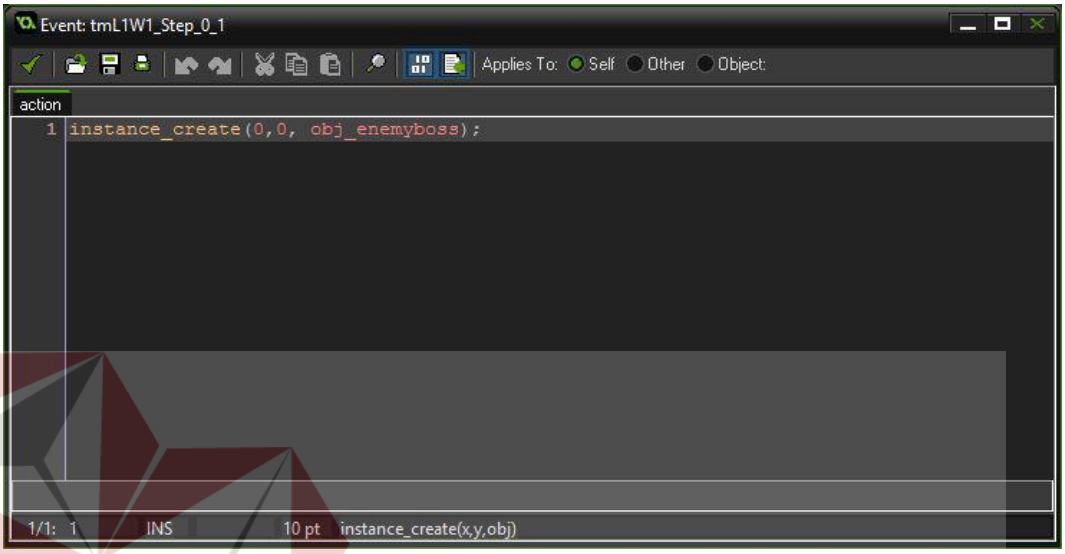
(Sumber : Olahan Penulis)



Gambar 5.16 Pemrograman *Wave Level 3*

(Sumber : Olahan Penulis)

Pada setiap wave terdiri dari 10 pasukan *enemy* dan 1 *enemy boss* pada akhir setiap *level.* Gambar berikutnya pemrograman *enemy* dalam setiap *wave.*

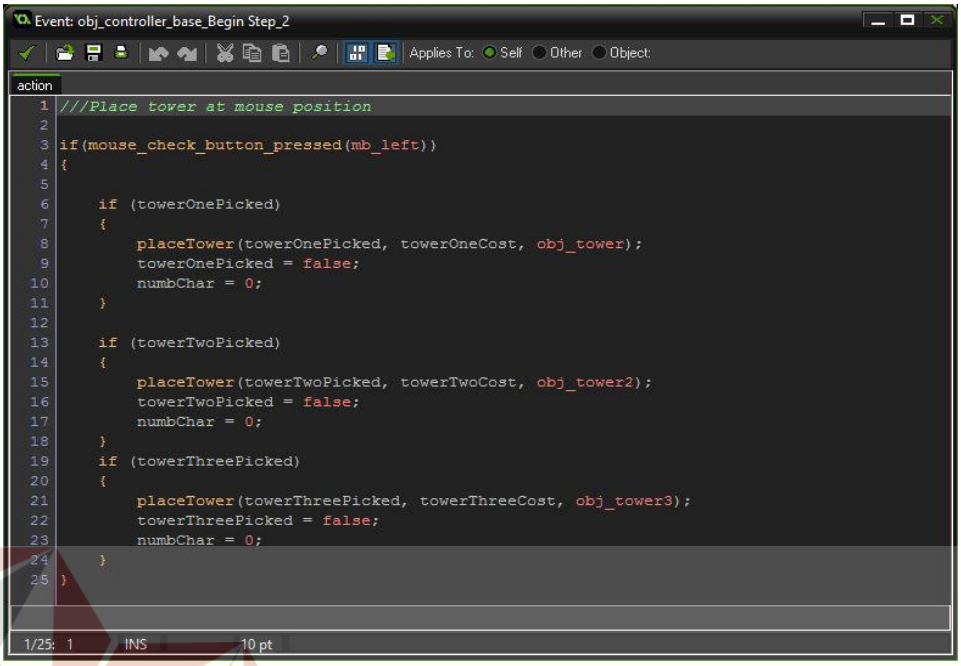


Gambar 5.17 Pemrograman *Enemy Wave*

(Sumber : Olahan Penulis)

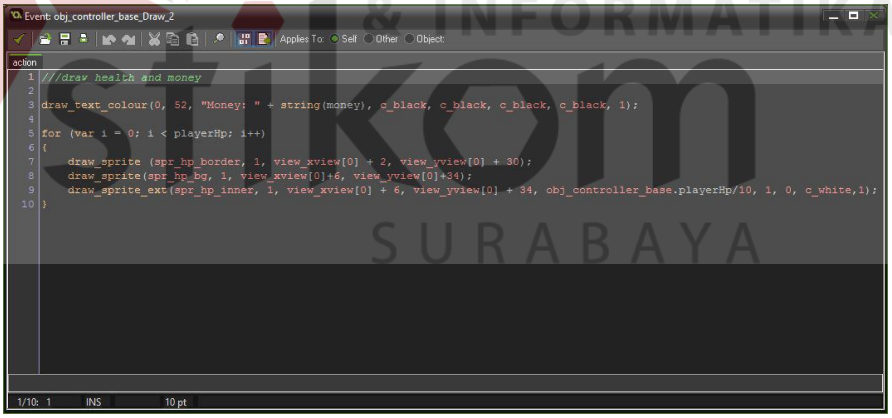
4. Controller

Pemrograman *controller* berfungsi untuk mengatur penempatan posisi *tower,* menunjukkan informasi *health bar* dan *point* yang telah dikumpulkan menunjukkan informasi *tower* dan untuk menjalankan *wave* disetiap *level.* Berikut adalah gambar pemrograman yang telah dibuat.



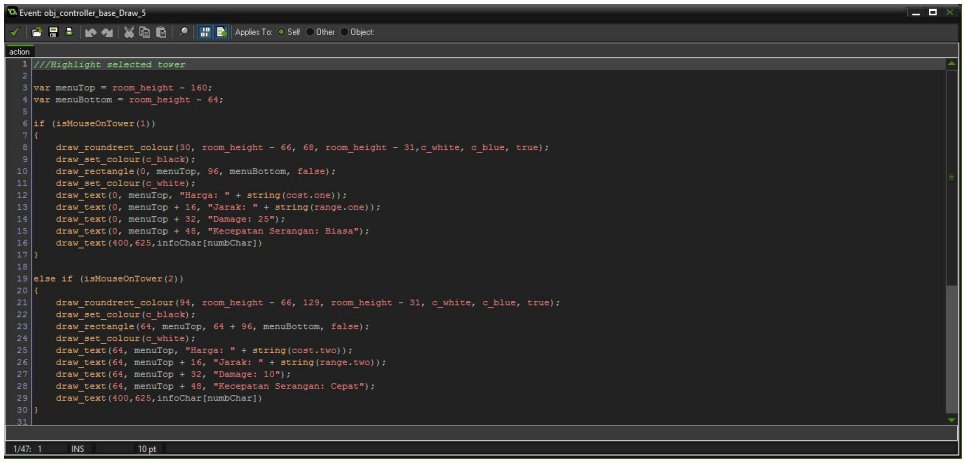
Gambar 5.18 Pemrograman *Controller* Penempatan *Tower*

(Sumber : Olahan Penulis)



Gambar 5.19 Pemrograman *Controller* Informasi *Health Bar* dan *Point*

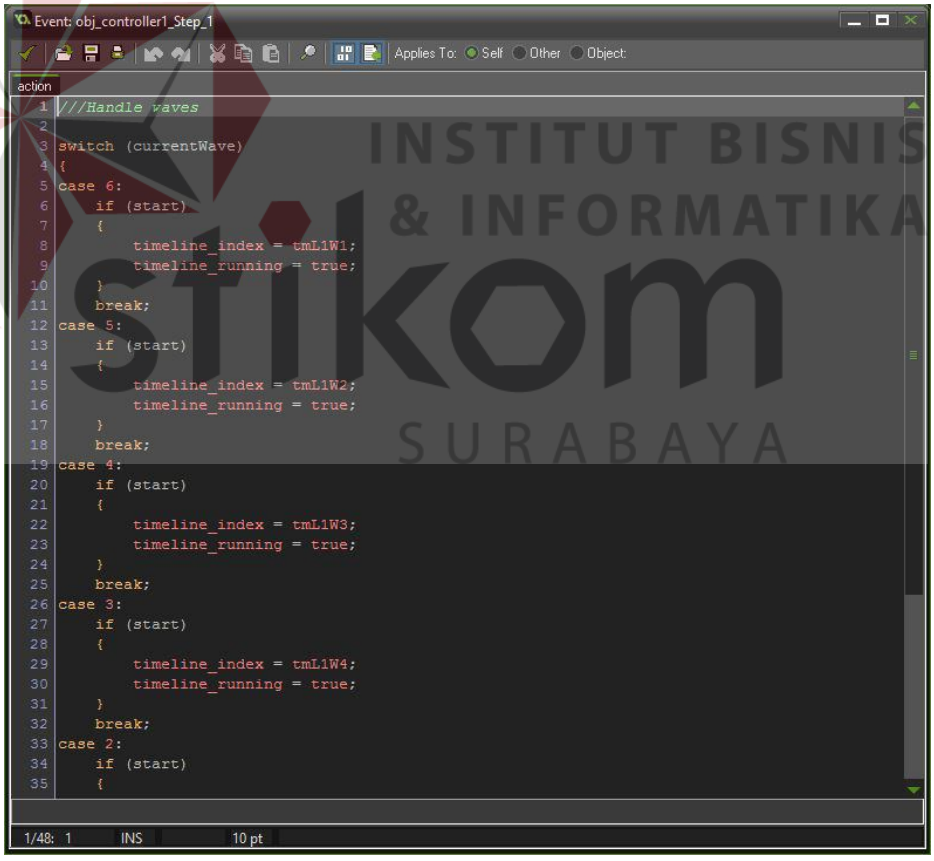
(Sumber : Olahan Penulis)



```
///Highlight selected tower

var menuTop = room_height - 160;
var menuBottom = room_height - 64;

if (isMouseOnTower(1))
{
    draw_roundrect_colour(30, room_height - 66, 68, room_height - 31,c_white, c_blue, true);
    draw_set_colour(c_black);
    draw_rectangle(0, menuTop, 96, menuBottom, false);
    draw_set_colour(c_white);
    draw_text(0, menuTop, "Harga: " + string(cost.one));
    draw_text(0, menuTop + 16, "Jarak: " + string(range.one));
    draw_text(0, menuTop + 32, "Damage: 25");
    draw_text(0, menuTop + 48, "Kecepatan Serangan: Biasa");
    draw_text(400,625,infoChar[numbChar])
}

else if (isMouseOnTower(2))
{
    draw_roundrect_colour(94, room_height - 66, 129, room_height - 31, c_white, c_blue, true);
    draw_set_colour(c_black);
    draw_rectangle(64, menuTop, 64 + 96, menuBottom, false);
    draw_set_colour(c_white);
    draw_text(64, menuTop, "Harga: " + string(cost.two));
    draw_text(64, menuTop + 16, "Jarak: " + string(range.two));
    draw_text(64, menuTop + 32, "Damage: 10");
    draw_text(64, menuTop + 48, "Kecepatan Serangan: Cepat");
    draw_text(400,625,infoChar[numbChar])
}
```

Gambar 5.20 Pemrograman *Controller* Informasi *Tower*

(Sumber : Olahan Penulis)



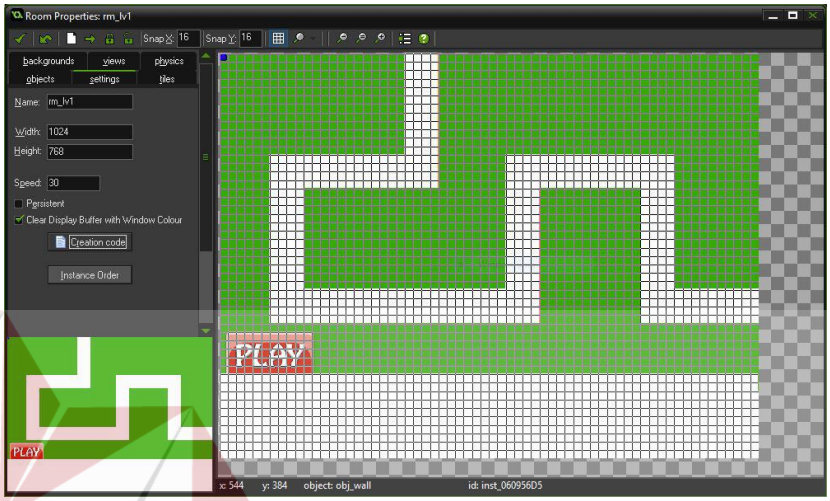
```
///Handle waves

switch (currentWave)
{
case 6:
    if (start)
    {
        timeline_index = tmL1W1;
        timeline_running = true;
    }
    break;
case 5:
    if (start)
    {
        timeline_index = tmL1W2;
        timeline_running = true;
    }
    break;
case 4:
    if (start)
    {
        timeline_index = tmL1W3;
        timeline_running = true;
    }
    break;
case 3:
    if (start)
    {
        timeline_index = tmL1W4;
        timeline_running = true;
    }
    break;
case 2:
    if (start)
    {
```

Gambar 5.21 Pemrograman *Controller Wave*

(Sumber : Olahan Penulis)

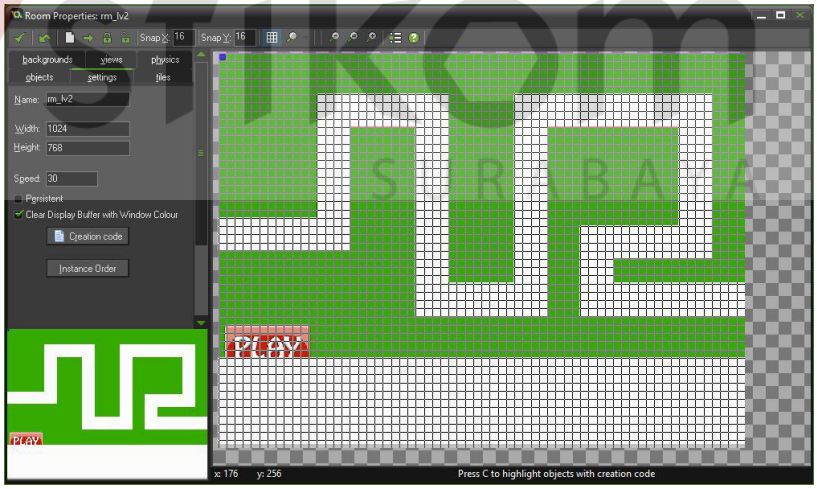
5. Pembuatan *Level*

Pembuatan *level* merupakan pembaruan pada *level* sebelumnya menjadi *level* berikutnya. Pembuatan *level game* ini dibuat menjadi pembuatan 3 *level.*



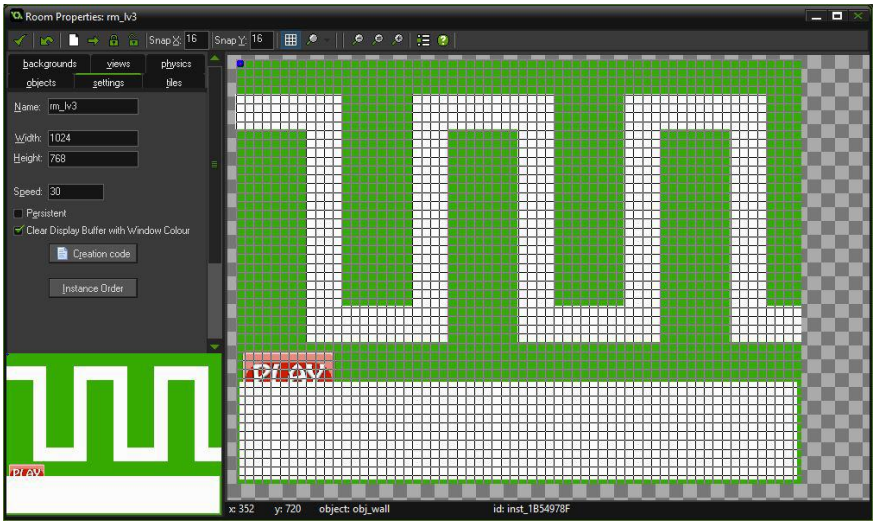
Gambar 5.22 Pemrograman Desain *Game level 1*

(Sumber : Olahan Penulis)



Gambar 5.23 Pemrograman Desain *Game level 2*

(Sumber : Olahan Penulis)



Gambar 5.24 Pemrograman Desain *Game level 3*

(Sumber : Olahan Penulis)

5.1.2 Development

Pada tahap ini, pemrograman *game* mulai dikembangkan. Desain antar muka diterapkan dalam *game engine, sprite* disatukan dengan *behavior* nya, dan *background music*. Berikut bagian-bagian dari *development*.

1. **Program**

   Game ini dibuat dengan menggunakan *game engine Game Maker Professional*. Dikutip dari *website* resmi *YoYo Games* (yoyogames.com) *YoYo Games is the home of GameMaker: Studio™, the fastest and friendliest cross-platform game development technology out there. GameMaker: Studio™ has been developed with usability and efficiency at its core, allowing developers to create games within a single code base. Games are then published to run natively across a multitude of platforms including Android, iOS, OS X, HTML5, Ubuntu, Windows 8, Windows Phone 8 and*

Windows RT. The entire GameMaker family of products has been downloaded more than five million times.

Selain itu *Game Maker Studio* tersedia dalam versi gratis dan berbayar. Versi berbayar menyediakan kelebihan fitur yang dapat meng-*export* data *game* agar bisa dimainkan di *platform mobile* dan *windows* yang tidak dapat digunakan pada versi gratis. *Game* ini menggunakan *Game Maker Studio Professional* versi berbayar karena *game* yang dibuat akan diluncurkan pada *platform mobile* dan *windows.* Berikut merupakan software bisa dilihat pada gambar 5.25.

Gambar 5.25 *Software Game Maker Studio Professional*

(Sumber : Olahan Penulis)

Game Maker Studio Professional merekomendasikan persyaratan sistem yang tersedia untuk *mac* dan *windows sebagai* berikut pada tabel 5.1.

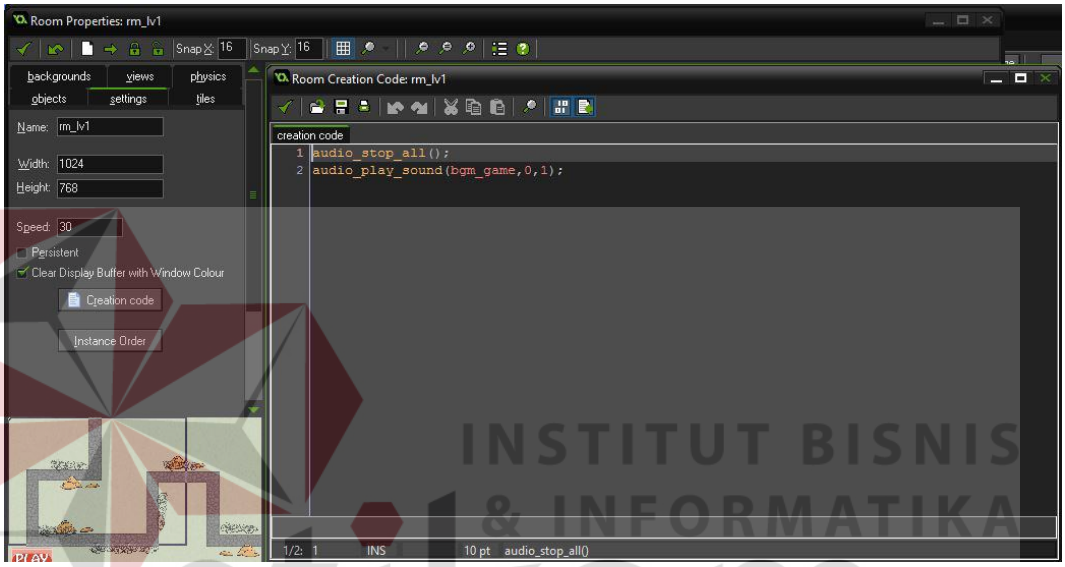
Tabel 5.1 *Game Maker Studio Professional System Requirements*

Mac:	***Windows:***
Mac:OS X 10.8 - 10.10 *OS X 10.9+ (required for publishing)* *Intel Core 2 Duo CPU* *200MB free disk space (for the software only)* *1GB RAM*	*Windows 7 or Windows 8* *OpenGL 2.0* *OpenAL* *Intel Core 2 Duo CPU (or AMD equivalent)* *200MB free disk space (for the software only)* *1GB RAM*

(Sumber: Olahan Penulis)

2. ***Background Music***

Background music dibuat digunakan untuk menunjang suasana pada *game.* Pemrograman *background music* dilakukan melalui *rooms* yang telah dibuat sehingga musik dapat berjalan dengan lancar.



Gambar 5.26 Pemrograman Musik

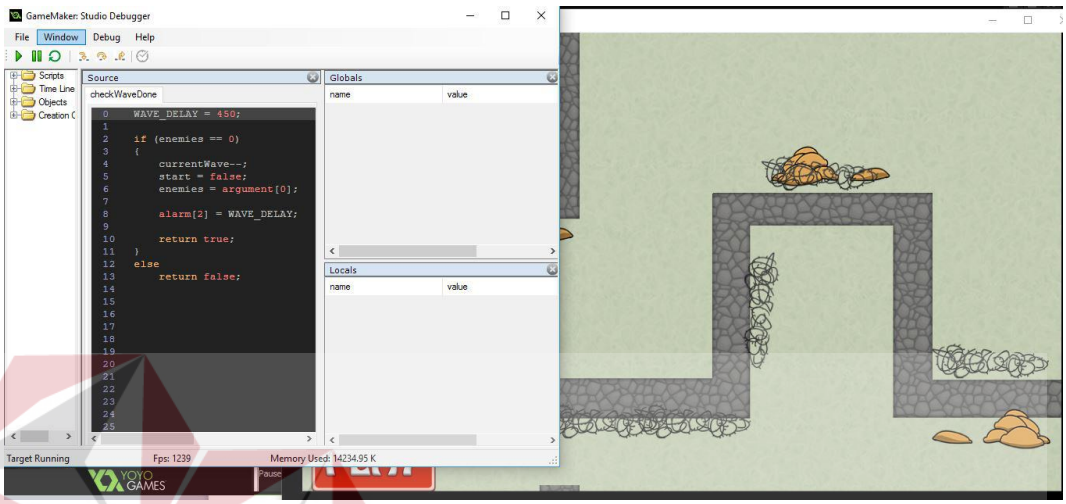
(Sumber: Olahan Penulis)

5.1.3 *Initial Balancing / Debugging*

Untuk mengetahui bekerja atau tidaknya komponen inti dari *game*, diperlukan *initial balancing* yaitu mencoba *game* dari awal hingga akhir sebelum diekspor dan dijadikan .exe, sehingga apabila ada kesalahan bisa segera diperbaiki.

Debugging dilakukan untuk mengetahui adanya *bug* atau permasalahan yang berorientasi pada kode-kode program. Proses *debugging* dilakukan dengan cara meng-*export* program *game* kedalam *windows.* Semua fungsi *game* yang sudah dibuat diuji. Jika terjadi *bug*, *error* atau fungsi yang tidak berjalan sebagaimana

mestinya, maka program *game* akan dikoreksi hingga *game* berfungsi tanpa ada *bug* atau *error*. Seperti pada gambar 5.27.

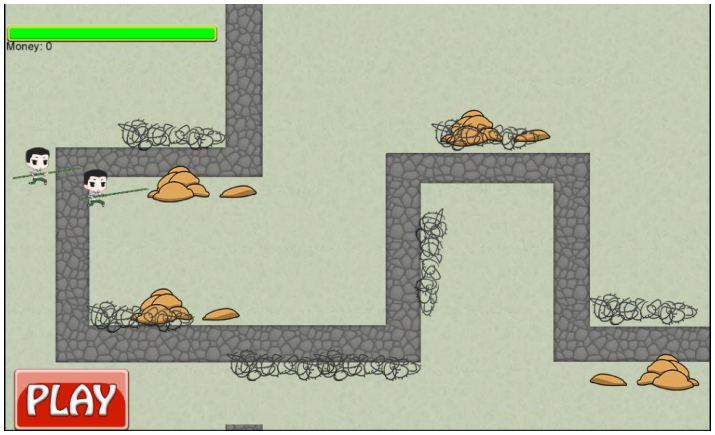


Gambar 5.27 *Testing Debugging*

(Sumber: Olahan Penulis)

5.2 Pasca Produksi

Dalam tahap pasca produksi hal yang dilakukan adalah *publishing*. Publikasi tersebut dilakukan dengan membuat poster dan *merchandise* seperti *sticker*, pin, dan mug serta proses *preview* dari *game* dari awal hingga akhir yang dicek ulang kembali agar semua nya berjalan normal tanpa ada masalah. Seperti pada gambar 5.28.



Gambar 5.28 Proses Pasca Produksi

(Sumber: Olahan Penulis)

5.2.1 *Play Testing*

Tujuan utama dari *play testing* adalah menguji lebih dalam lagi. Pengujian ini dibagi menjadi 3 yaitu *alpha, beta,* dan *gamma.* Dalam tahap *alpha test,* pengujian dilakukan oleh pihak pengembang untuk menemukan *error* atau *bug.* Kemudian pada tahap *beta test*, *game* yang sudah dibuat diujikan ke pihak *external* untuk mendapatkan masukan dalam fitur-fitur permainan. Pada tahap *gamma,* masukan-masukan yang telah didapat kemudian di implementasikan ke *game* yang telah dibuat. Penulis menemukan *error* pada sprite dimana rotasi *sprite* yang sebelumnya hanya memiliki 4 arah dirubah menjadi 8 arah agar pergerakan karakter tidak terlihat kaku.

5.2.2 Publikasi

Kegiatan pasca produksi meliputi tiga aspek yaitu, pembuatan kemasan dan publikasi kepada masyarakat tentang *game* ini.

1. Kemasan

Game yang sudah jadi ini di*burn* pada kepingan CD. Setelah di*burn,* kepingan CD dimasukkan dalam CD-*Case*. Agar kepingan CD dan CD-*Case* tidak terkesan *plain,* maka inilah hasil dari desainnya.

Gambar 5.29 Desain Label CD *Game*
(Sumber: Olahan Penulis)

2. Publikasi

Kegiatan publikasi meliputi pembuatan poster, pembuatan *merchandise* berupa stiker dan gelas serta kegiatan pameran sebagai sarana mengenalkan *game* ini. Berikut adalah hasil jadi dari *merchandise* di pameran.

Gambar 5.30 Desain Poster *Game*

(Sumber: Olahan Penulis)

Gambar 3.31 Desain Sticker
(Sumber: Olahan Penulis)

Gambar 3.32 Desain Pin
Sumber: (Olahan Penulis)

Gambar 3.33 Desain mug
Sumber: (Olahan Penulis)

3. Rilis

Pada tahapan rilis merupakan hak dari pembuat *game*, untuk tahapan rilis yang diharapkan *game* tersebut bisa bersaing dengan *game* dari banyak developer *game-game* lainnya.

4. Realisasi Anggaran

Pada perancangan karya telah terdapat rancangan anggaran untuk pembuatan karya tugas akhir ini. Tabel 5.2 merupakan tabel realisasi anggaran setelah pembuatan karya tugas akhir.Seperti pada gambar tabel 5.2.

Tabel 5.2 Realisasi Anggaran

Uraian	Qty	Harga Satuan	Total
Perangkat Keras			
PC Rakitan	1 set	Rp. 6.000.000,-	Rp. 6.000.000,-
Internet	2 Bulan	Rp. 230.000,-	Rp. 460.000,-

Perangkat Lunak			
Adobe Photoshop CS 6	1 set	Rp. 500.000,-	Rp. 500.000,-
Game Maker Studio Professional	1 set	Rp. 1.000.000,-	Rp. 1.000.000,-
		Grand Total	**Rp. 8.360.000,-**

(Sumber: Olahan Penulis)

5.3 Real Produksi

5.3.1 Kasus

Dalam proses produksi pembuatan game ini, terdapat permasalahan yang dialami aplikasi *Game Maker Studio Professional* yang kadang kala mengalami *error* dan *stuck* sampai membuat pekerjaan tidak bisa di*save* dan harus mengulanginya dari awal. Jadwal yang sudah direncanakan pun menjadi tidak sesuai dan membuat produksi karya menjadi tidak tepat waktu.

5.3.2 Strategi Mengatasinya

Dalam menghadapi aplikasi *Game Maker Studio Profressional* yang kadang kala mengalami *error* dan *stuck*, penulis melakukan *scanning virus* pada komputer agar kinerja komputer membaik sehingga aplikasi *Game Maker Studio Professional* tidak mengalami *error dan stuck.*

5.4 *Screenshot*

Gambar 5.34 *Main Menu*

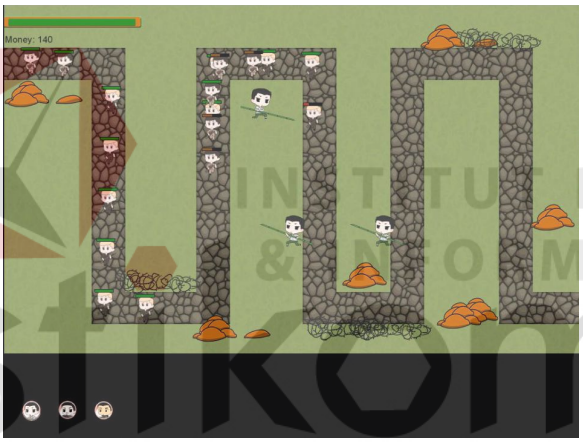
(Sumber: Olahan Penulis)

Gambar 5.35 *Level 1*

(Sumber: Olahan Penulis)

Gambar 5.36 *Level 2*

(Sumber: Olahan Penulis)



Gambar 5.37 *Level 3*

(Sumber: Olahan Penulis)

Gambar 5.38 *Win Condition*

(Sumber: Olahan Penulis)

Gambar 5.39 *Lose Condition*

(Sumber: Olahan Penulis)