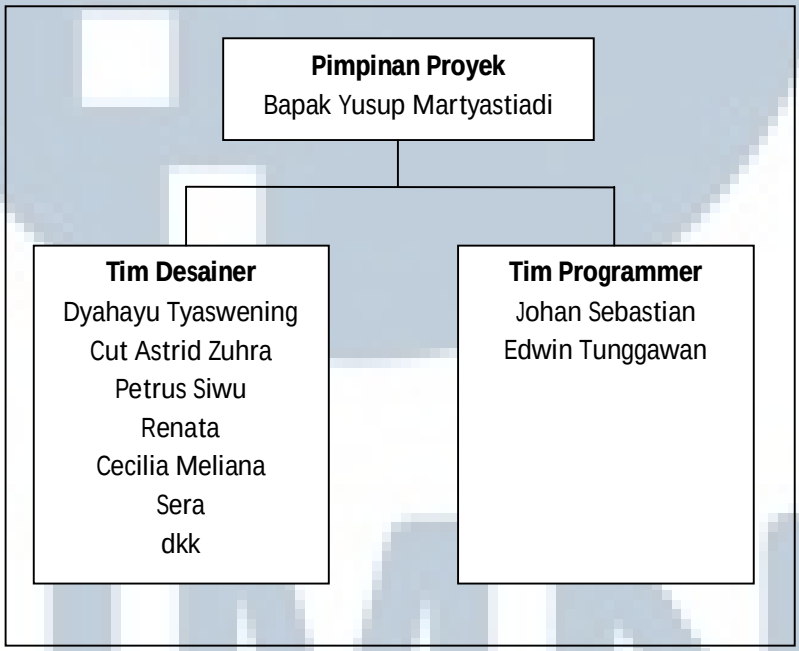
BAB III
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama periode magang, peserta magang berperan sebagai *game programmer* yang bertugas memprogram JavaScript untuk *battle system* dalam *game* yang dibuat dengan menggunakan *game engine* Unity. Pembuatan model dan perancangan *scene* untuk *battle system* dilakukan oleh Saudari Dyahayu Tyaswening, sebagai desainer yang bertanggung jawab atas *battle system* dari game yang bersangkutan.

Secara skema, kedudukan dan koordinasi peserta magang selama periode magang yaitu sebagai berikut.



Gambar 3.1 Skema Kedudukan

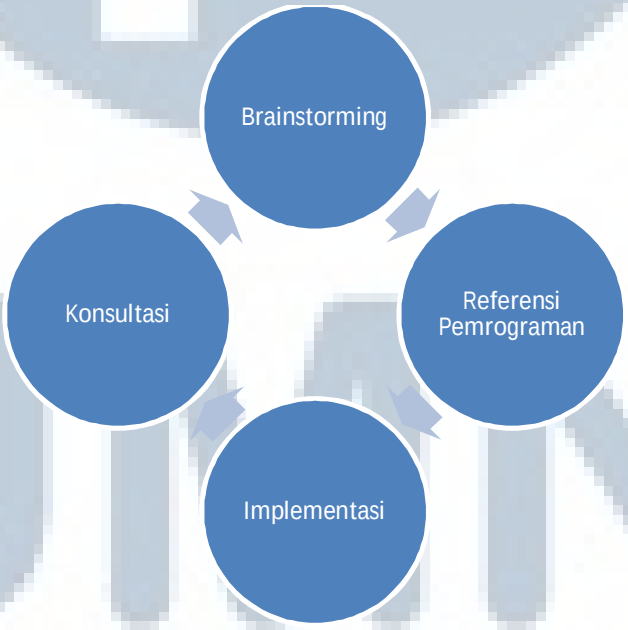
3.2 Tugas yang Dilakukan

Selama periode magang, peserta magang bertugas untuk membuat *battle system* sesuai dengan desain yang didefinisikan oleh para desainer yang mendesain *game* yang bersangkutan, dan juga membantu Saudari Cut Astrid Zuhra, desainer yang bertugas membuat perancangan *gameplay* untuk *game* ini, antara lain dalam pembuatan *flow*, logika, perhitungan, dan *balancing*. Peserta magang ikut membantu dalam perancangan *game*, terutama di bagian-bagian yang banyak membutuhkan perhitungan dan logika.

Pada awalnya, peserta magang hanya bertugas untuk membuat *script battle system* sesuai dengan desain yang telah ditetapkan oleh desainer. Tetapi karena adanya kendala-kendala dalam proses desain, maka peserta magang ikut membantu dalam proses desain game.

3.3 Tahapan Pelaksanaan Kerja Magang

Tahapan pelaksanaan kerja magang yang dilakukan oleh peserta magang selama periode magang di Universitas Multimedia Nusantara dapat digambarkan sebagai berikut.



Gambar 3.2 Tahap Pelaksanaan Kerja Magang

3.3.1 *Brainstorming*

Pada tahap ini, tim pengembang *game* akan berkumpul dan mengemukakan ide-ide mereka. Di sini akan dipilih ide-ide yang akan dimasukkan ke dalam game dan akan dilihat apakah ide tersebut bisa direalisasikan.

Peserta magang juga membantu memberikan saran-saran dan menambahkan ide-ide yang berkaitan pengembangan *battle system* dan mendiskusikannya dengan Saudari Dyahayu Tyaswening dan Saudari Cut Astrid Zuhra, terutama dalam hal-hal yang menyangkut perhitungan matematis dan yang berkaitan langsung dengan logika *scripting*, seperti rumus perhitungan *battle damage*, yang akan diimplementasikan oleh peserta magang.

3.3.2 Referensi Pemrograman

Masing-masing *programmer* akan mempelajari bagaimana cara mengimplementasikan hasil *brainstorming* ke dalam *game* yang sedang dibuat. Karena tim *programmer*, termasuk peserta magang, belum pernah menggunakan *game engine* Unity yang digunakan dalam proyek ini sebelumnya, maka pencarian referensi pemrograman dan *self-tutorial* menjadi salah satu tahap yang krusial dalam pengembangan *game* ini.

Unity adalah sebuah *game engine* yang bersifat *cross-platform* yang dikembangkan oleh Unity Technologies. *Game* yang dibuat dengan menggunakan *game engine* Unity dapat dirilis untuk *platform desktop*, *mobile*, dan web. Dalam *game engine* ini juga disertakan IDE MonoDevelop sebagai IDE *default* yang digunakan untuk melakukan *scripting* pada Unity.

Berdasarkan input dari Saudara Jason Christian dari Game Development Club UMN, peserta magang melakukan instalasi IDE UnityScript Editor untuk *scripting* karena UnityScript Editor memiliki fitur *autocomplete* yang dapat memudahkan peserta magang dalam menulis dan mempelajari fungsi-fungsi yang telah disediakan oleh Unity. Fitur *autocomplete* tersebut tidak dimiliki oleh IDE MonoDevelop, sehingga peserta magang memilih menggunakan IDE UnityScript

Editor dibandingkan dengan MonoDevelop yang sudah disertakan dalam *game engine* Unity.

Peserta magang mengacu pada buku berjudul "Unity 3.x Game Development Essentials: Game development with C# and JavaScript" yang ditulis oleh Will Goldstone sebagai buku panduan utama dalam proses pembelajaran. Peserta magang juga mendapatkan bantuan berupa penjelasan dan demo mengenai fitur-fitur dan *scripting* JavaScript dengan Unity dari Saudara Jason Christian, yang memudahkan peserta magang dalam mempelajari *game engine* Unity.

3.3.3 Implementasi

Pada tahap ini peserta magang akan mengimplementasikan desain *game* dan *requirements* ke dalam *game*. Dalam tahap ini, peserta magang akan menempatkan model *player character*, musuh, dan kamera dalam *battle scene* menggunakan *game engine* Unity. Setelah model-model ditempatkan, peserta magang akan melakukan *scripting* untuk memprogram gerakan-gerakan komponen yang telah diletakkan di dalam *battle scene*.

Scripting akan dilakukan dengan menggunakan JavaScript, sebagai salah satu bahasa yang didukung oleh *game engine* Unity, dengan menggunakan IDE UnityScript Editor.

Dalam proses *scripting*, peserta magang menggunakan beberapa fungsi yang menjadi fitur dalam Unity. Fungsi-fungsi tersebut adalah sebagai berikut.

1. Start()

   Fungsi ini dijalankan secara otomatis oleh Unity satu kali saja, pada saat Unity selesai memuat komponen-komponen yang akan digunakan untuk *game* pada saat *game* tersebut dimainkan. Berguna untuk inisialisasi variabel untuk objek-objek di dalam *game*.
2. Update()

   Fungsi ini dijalankan secara otomatis oleh Unity setiap kali terjadi pergantian *frame* pada tampilan *game*.

3. OnGUI()

   Fungsi ini berguna untuk menampilkan teks, tombol, dan lain-lain ke layar monitor.

4. Random.Range()

   Berguna untuk mendapatkan sebuah bilangan acak di antara dua buah bilangan yang dijadikan parameter untuk fungsi yang bersangkutan.

5. GameObject.Find()

   Digunakan untuk mendapatkan objek-objek di dalam *game* untuk digunakan di dalam proses. Menerima parameter berupa *string* yang berisi nama yang diberikan kepada suatu objek dalam *game* dan mengembalikan objek tersebut.

6. GetComponent()

   Merupakan *method* dari *class GameObject*, menerima parameter berupa *string* yang berisi nama *script* yang di-*assign* untuk objek dari *class GameObject*. Berfungsi untuk menjalankan fungsi yang berada dalam *script* yang di-*assign* ke objek lain yang berada di dalam *game*.

7. animation.Play()

   Digunakan untuk menjalankan animasi-animasi yang telah dibuat ke dalam model yang digunakan untuk objek yang bersangkutan. Menerima parameter berupa *string* yang berisi nama animasi yang sudah ditetapkan.

8. transform.Translate()

   Mentranslasikan suatu objek ke arah yang diinginkan. Menerima tiga buah parameter berupa bilangan dengan tipe data *float* yang berisi informasi koordinat x, y, dan z yang menentukan ke arah mana objek yang bersangkutan akan ditranslasikan. Perhitungan translasi relatif terhadap objek yang ditranslasikan, yang mana objek yang ditranslasikan akan dianggap berada di posisi titik koordinat (0, 0, 0).

3.3.4 Konsultasi

Peserta magang akan melaporkan hasil kerja kepada Bapak Yusup Martyastiadi, yang berlaku sebagai pimpinan proyek, dan kepada Saudari

Dyahayu Tyaswening sebagai desainer *battle system*. Jika terdapat kesalahan dalam pengerjaan, hasil implementasi akan diperbaiki.

Untuk pengerjaan *battle stats* karakter-karakter, peserta magang melakukan koordinasi dan juga konsultasi dengan Saudari Cut Astrid Zuhra dan Saudara Johan Sebastian yang mengerjakan perhitungan poin *upgrade* untuk senjata dan *armor*.

3.4 Uraian Pelaksanaan Kerja Magang

Berikut adalah uraian pelaksanaan kerja magang yang dilakukan oleh peserta magang selama membangun *battle system* untuk *game* RPG ini.

3.4.1 Terminologi Dasar

Berikut ini terminologi-terminologi dasar yang banyak digunakan dalam permainan *video game*.

1. *Player character*, yaitu karakter yang dapat dimainkan oleh pemain dalam sebuah *game*.
2. *Non-player character*, yaitu karakter yang tidak dapat dimainkan oleh pemain dalam sebuah *game*. *Non-player character* dapat berupa figuran, teman-teman dari *player character* yang dikendalikan oleh komputer, atau musuh yang harus dikalahkan.
3. *Stats*, yaitu nilai-nilai yang diberikan ke variabel-variabel tertentu yang ada di dalam suatu objek dari sebuah karakter yang terdapat di dalam *game*. Nilai-nilai ini menentukan kekuatan-kekuatan yang dimiliki oleh karakter-karakter tersebut dalam *game* yang bersangkutan.
4. *Base stats*, yaitu nilai-nilai *stats* yang terdapat dalam suatu karakter tanpa ditambahkan dengan nilai-nilai bonus yang dapat ditambahkan kepada karakter tersebut.
5. *Default stats*, yaitu nilai-nilai *stats* yang dimiliki oleh karakter secara *default* pada saat *game* dimulai. Nilai *stats* karakter dapat berubah seiring dengan meningkatnya level dari karakter tersebut nantinya.

6. *Random battle,* yaitu suatu keadaan di mana pemain akan bertemu dengan lawan secara acak ketika pemain menjelajahi area di dalam *game*.
7. *Boss battle,* yaitu suatu keadaan di mana pemain akan bertemu dengan lawan yang harus dikalahkan untuk melanjutkan cerita. Lawan yang ditemui dalam *boss battle* umumnya lebih sulit dikalahkan jika dibandingkan dengan lawan-lawan yang ditemui oleh pemain dalam *random battle.*
8. *Final boss,* yaitu lawan yang harus dikalahkan oleh pemain di dalam *boss battle* terakhir di dalam permainan *video game*. Setelah *final boss* dikalahkan, maka pemain dapat melihat *ending* dari *game* yang dimainkannya dan *game* tersebut pun tamat.
9. *Active time battle,* atau biasa disingkat ATB, adalah suatu sistem perhitungan untuk menentukan karakter mana yang mendapatkan giliran untuk menyerang dalam suatu *battle scene.*
10. *Status ailments,* yaitu status-status tambahan yang bisa diberikan kepada suatu karakter dan dapat menyulitkan karakter tersebut dalam *battle.*
11. *Item drop,* yaitu benda-benda yang dijatuhkan oleh musuh-musuh yang dikalahkan oleh pemain dalam *battle*. Benda-benda ini dapat digunakan untuk membantu pemain selama dalam permainan.
12. *Side quest,* yaitu misi-misi tambahan yang terdapat di dalam *game* yang dapat diselesaikan oleh pemain untuk mendapatkan suatu *item* ataupun potongan-potongan cerita tambahan. Misi-misi ini tidak wajib diselesaikan oleh pemain untuk menyelesaikan *game*, tetapi pemain bisa mendapatkan *reward* yang dapat membantunya dalam permainan jika diselesaikan. Dalam beberapa *game*, senjata dan perlengkapan terkuat dari *player character* hanya bisa didapatkan dengan menyelesaikan *side quest* yang berhubungan dengan karakter tersebut.

3.4.2 *Game Design*

Game berjudul "Emendation" ini adalah sebuah *game* bergenre RPG, dengan gaya *Japanese RPG* dan *setting* fantasi yang futuristik. Secara umum,

desain *game* untuk "Emendation" bisa dibagi menjadi dua bagian utama: eksplorasi kota dan *battle*. Dalam bagian eksplorasi kota, pemain dapat menggerakkan *player character* unuk menjelajahi kota di mana *player character* berada dan mengumpulkan *item* yang terdapat pada lokasi-lokasi tertentu dalam kota tersebut.

Pada saat menjelajahi kota, pemain dapat menemukan beberapa *non-player character* yang meminta bantuan kepada *player character*. Jika pemain menyelesaikan misi yang diberikan, maka pemain bisa mendapatkan *reward* dari *side quest* tersebut.

Player character akan menemui musuh secara acak pada saat menjelajahi kota, dan adegan akan dipindahkan dari adegan kota ke *battle scene*. Dalam *battle scene, player character* akan menghadapi musuh-musuh yang memiliki tipe yang berbeda-beda. Jika pemain berhasil mengalahkan musuh yang muncul, maka pemain akan mendapatkan poin *experience* yang berguna untuk menaikkan level dan kekuatan dari *player character*. Dari *battle*, pemain juga akan mendapatkan uang yang bisa digunakan untuk membeli *item* dan memperkuat senjata dan perlengkapan-perlengkapan lain yang dimiliki oleh *player character*. Setiap musuh memberikan jumlah poin *experience* dan uang yang berbeda, dengan kemungkinan untuk menjatuhkan *item* yang berguna bagi pemain.

Tim desainer mengharapkan *battle system* dari "Emendation", selain dapat dimainkan dengan lancar tanpa bug, nantinya juga dapat memiliki tampilan *user interface* yang baik dan menarik. Dari *game-game* bergaya *Japanese RPG* yang telah popular, tampaknya *user interface battle system* yang paling mendekati harapan dari tim desainer adalah tampilan *user interface battle system* dari *game* "Persona 4" yang dirilis oleh Atlus.

Gambar 3.3 *Battle System* dalam *Game* "Persona 4"

Desain yang ditetapkan oleh para *game designer* dan peserta magang adalah sebagai berikut:

1. *Random battle*, dengan lawan yang muncul berjumlah 1-2 musuh. *Random battle* akan muncul secara acak pada saat *player character* melangkah saat menjelajahi area permainan.
2. *Player character* dapat dikendalikan melalui *battle menu* yang berisi pilihan-pilihan sebagai berikut.
   a. *Attack*

      Menyerang dengan serangan standar. *Damage* yang disebabkan oleh serangan ini akan dihitung dengan menggunakan kalkulasi *damage* untuk *physical attack*.
   b. *Skill*

      Menggunakan kemampuan khusus milik *player character*.
   c. *Item*

      Menggunakan *item* yang terdapat dalam *inventory* untuk membantu dalam *battle*.

3. *Camera view* pada saat *battle scene* mengambil gambar dari belakang *player character*.
4. *Battle system* bersifat *turn based* dengan menggunakan sistem *Active Time Battle* (ATB) untuk menentukan karakter mana yang mendapat giliran untuk menyerang pada saat itu. Masing-masing karakter akan memiliki *ATB counter* yang akan terus bertambah. Pada saat nilai *ATB counter* dari karakter tersebut mencapai nilai maksimum, maka karakter tersebut mendapatkan giliran untuk menyerang.

Setiap karakter yang terlibat dalam *battle* memiliki parameter-parameter yang memiliki pengaruh dalam jalannya *battle system*, yaitu:

1. *Strength*

   Merupakan parameter yang menentukan seberapa besar *damage* yang dapat disebabkan oleh suatu karakter dengan menggunakan serangan yang termasuk ke dalam kategori *physical attack*.
2. *Vitality*

   Merupakan parameter yang menentukan seberapa besar reduksi *damage* yang didapatkan oleh suatu karakter pada saat diserang oleh musuh dalam *battle*. *Vitality* berperan besar dalam pengurangan *physical damage*, dan berperan dalam pengurangan *magical damage* meskipun tidak terlalu besar pengaruhnya.
3. *Intelligence*

   Merupakan parameter yang menentukan seberapa besar damage yang dapat disebabkan oleh suatu karakter dengan menggunakan serangan yang termasuk ke dalam kategori *magical attack*.
4. *Wisdom*

   Merupakan parameter yang menentukan seberapa besar reduksi *damage* yang didapatkan oleh suatu karakter pada saat diserang oleh musuh dalam *battle*. *Wisdom* berperan besar dalam pengurangan *magical damage*, dan berperan dalam pengurangan *physical damage* meskipun tidak terlalu besar pengaruhnya.

5. *Hit Point*

   Dapat dikatakan sebagai "nyawa" dari suatu karakter. *Damage* yang diterima oleh suatu karakter dalam *battle* akan mengurangi nilai *hit point* dari karakter tersebut. Ketiga nilai *hit point* mencapai nol, maka karakter tersebut akan kalah dan menghilang dari *battle*.

6. *Skill Point*

   Parameter yang menentukan seberapa sering suatu karakter bisa menggunakan kemampuan-kemampuan khususnya. Setiap kali suatu karakter menggunakan *skill*, maka *skill point* dari karakter tersebut akan berkurang sejumlah konsumsi SP yang dibutuhkan oleh *skill* tersebut. Jika *skill point* tidak mencukupi, maka karakter yang bersangkutan tidak dapat menggunakan *skill* itu.

7. *Max ATB Counter*

   Menentukan panjang interval di antara tindakan-tindakan yang dapat diambil oleh suatu karakter dalam *battle*. Semakin besar nilai *max ATB counter* dari suatu karakter, maka semakin panjang pula interval di antara tindakan-tindakan yang dapat diambil oleh karakter tersebut dalam *battle.*

8. *Weapon Attack Bonus*

   Bonus poin yang didapatkan oleh *player character* dari senjata yang digunakan. Nilai ini akan ditambahkan ke jumlah *damage* yang dikenakan kepada lawan yang diserang oleh *player character* dalam *battle*. Untuk *non-player character*, nilai *weapon attack bonus* adalah nol.

9. *Armor Defense Bonus*

   Bonus poin yang didapatkan oleh *player character* dari *armor* yang digunakan. Nilai ini akan ikut menambahkan nilai reduksi *damage* yang diterima oleh *player character* pada saat *player character* diserang oleh lawan dalam *battle.* Untuk *non-player character*, nilai *armor defense bonus* adalah nol.

Rumus kalkulasi *damage* yang digunakan dalam *battle system* untuk *game* ini adalah sebagai berikut.

1. Perhitungan *damage* untuk *physical attack* dari suatu karakter:

*Damage = 7*Strength + Weapon Attack Bonus*

2. Perhitungan *damage* untuk *magical attack* dari suatu karakter:
   *Damage = 7*Intelligence + Weapon Attack Bonus*
3. Reduksi *damage* untuk *physical attack* yang diarahkan ke suatu karakter:
   Damage = Damage – (Vitality + Wisdom/2 + Armor Defense Bonus)
4. Reduksi *damage* untuk *magical attack* yang diarahkan ke suatu karakter:
   Damage = Damage – (Wisdom + Vitality/2 + Armor Defense Bonus)

Hanya terdapat satu *player character* dalam *game* ini, yaitu seorang wanita muda yang bernama Vla. Di dalam permainan ini, Vla akan menemui tiga jenis robot yang berlaku sebagai musuh dalam *random battle* dan seorang bernama Vincent sebagai *final boss* yang akan dihadapi dalam *boss battle.*

Gambar 3.4 Desain Karakter Vla oleh Saudari Renata

Berikut ini *base stats* yang dimiliki oleh Vla pada level 1.

Tabel 3.1 *Base Stats* Vla

Nama	Vla
HP	500
SP	200
Strength	10
Vitality	10
Intelligence	10
Wisdom	10

Berikut ini perhitungan kenaikan status untuk HP dan SP setiap kali Vla mengalami kenaikan level dalam *game*.

- Level 1-10 → 20 + level – 1
- Level 11-20 → 40 + level – 1
- Level 21-30 → 60 + level – 1
- Level 31-40 → 80 + level – 1
- Level 41-50 → 100 + level – 1
- Level 51-60 → 120 + level – 1
- Level 61-70 → 140 + level – 1
- Level 71-80 → 160 + level – 1
- Level 81-90 → 180 + level – 1
- Level 91-100 → 200 + level – 1

Dengan demikian, pada level 5 Vla akan memiliki nilai HP maksimum sebagai berikut.

MaxHP = 500 + (20 + 2 – 1) + (20 + 3 – 1) + (20 + 4 – 1) + (20 + 5 – 1)

MaxHP = 570

Sementara itu, untuk kenaikan nilai SP, setiap kali Vla mengalami kenaikan level maka kenaikan nilai SPnya adalah sebagai berikut.

- Level 1-10 → 10
- Level 11-20 → 20
- Level 21-30 → 30
- Level 31-40 → 40
- Level 41-50 → 50
- Level 51-60 → 60
- Level 61-70 → 70
- Level 71-80 → 80
- Level 81-90 → 90
- Level 91-100 → 100

Dengan demikian, pada level 5 Vla akan memiliki nilai HP maksimum sebagai berikut.

MaxSP = 200 + 10 + 10 + 10 + 10

MaxSP = 240

Setiap kali mengalami kenaikan level, Vla akan mendapatkan 5 *ability point* yang dapat dialokasikan untuk menambahkan *stats* yang dia miliki. Di awal *game*, Vla akan memiliki 20 *ability point* yang secara *default* telah didistribusikan secara merata ke poin *Strength*, *Vitality*, *Intelligence*, dan *Wisdom.* Berikut ini *default stats* Vla pada awal *game*.

Tabel 3.2 *Default Stats* Vla

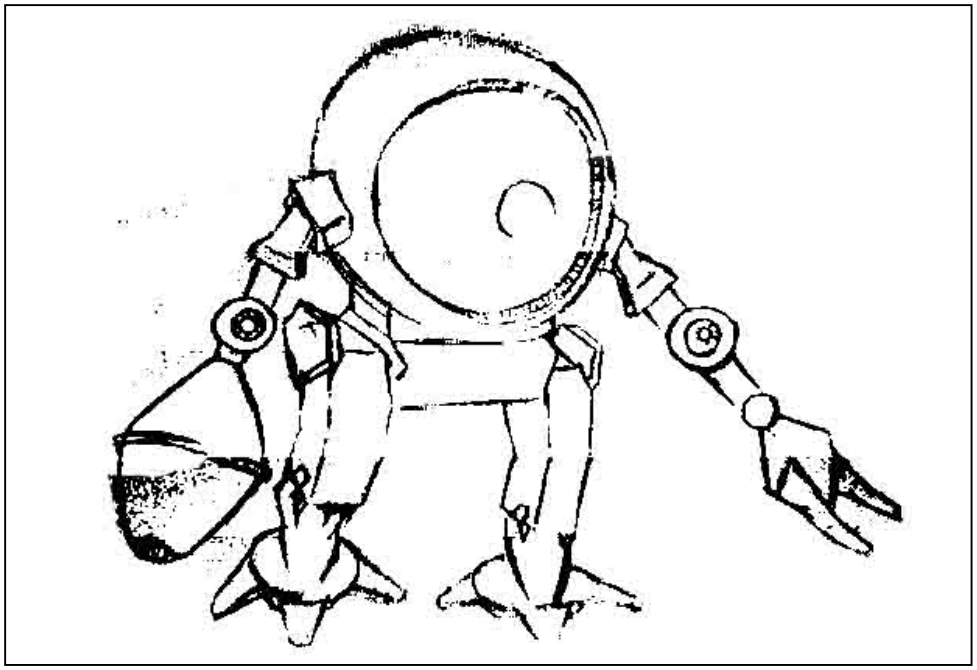
Nama	Vla
HP	570
SP	240
Strength	14
Vitality	14
Intelligence	14
Wisdom	14

Dalam *game* ini, terdapat tiga macam *status ailments* yang dapat menyulitkan *player character* dalam *battle*. Berikut ini keterangan mengenai ketiga *status ailments* tersebut.

Tabel 3.3 Daftar *Status Ailments*

Stun	Membuat karakter yang terkena *status ailment* ini tidak dapat melakukan apa-apa untuk tiga giliran menyerang.
Confuse	Membuat karakter tidak terkendali untuk tiga giliran menyerang. Selama itu karakter yang bersangkutan dapat menyerang dirinya sendiri dengan perhitungan *damage* normal atau menyerang sembarang musuh dengan akurasi yang rendah dan *damage* setengah dari *damage* normalnya.
Poison	Membuat karakter yang terkena *status ailment* ini terkena *damage* senilai 2% dari nilai maksimum HPnya setiap kali karakter tersebut selesai melakukan serangan. *Status ailment* ini adalah satu-satunya *status ailment* yang terus bertahan sampai disembuhkan oleh pemain. Jika tidak disembuhkan, *status ailment* ini akan terus mengurangi HP Vla setelah *battle* usai ketika Vla berjalan.

Seperti yang sudah dijelaskan sebelumnya, Vla akan bertemu dengan tiga jenis robot yang akan berlaku sebagai lawan yang dihadapinya dalam *random battle*. Pada saat ini, masing-masing robot tersebut disebut dengan nama Robot Tipe A, Robot Tipe B, dan Robot Tipe C.



Gambar 3.5 Desain Robot Tipe A oleh Saudari Renata

Robot Tipe A adalah robot dengan kekuatan tingkat menengah yang akan sering ditemui oleh Vla dalam permainan. Robot ini tidak terlalu kuat, tetapi dapat menyulitkan pemain dengan *status ailments* seperti *stun* dan *confuse*. Robot Tipe A dapat muncul sendirian saja, atau berpasangan.

Tabel 3.4 *Stats* Robot Tipe A

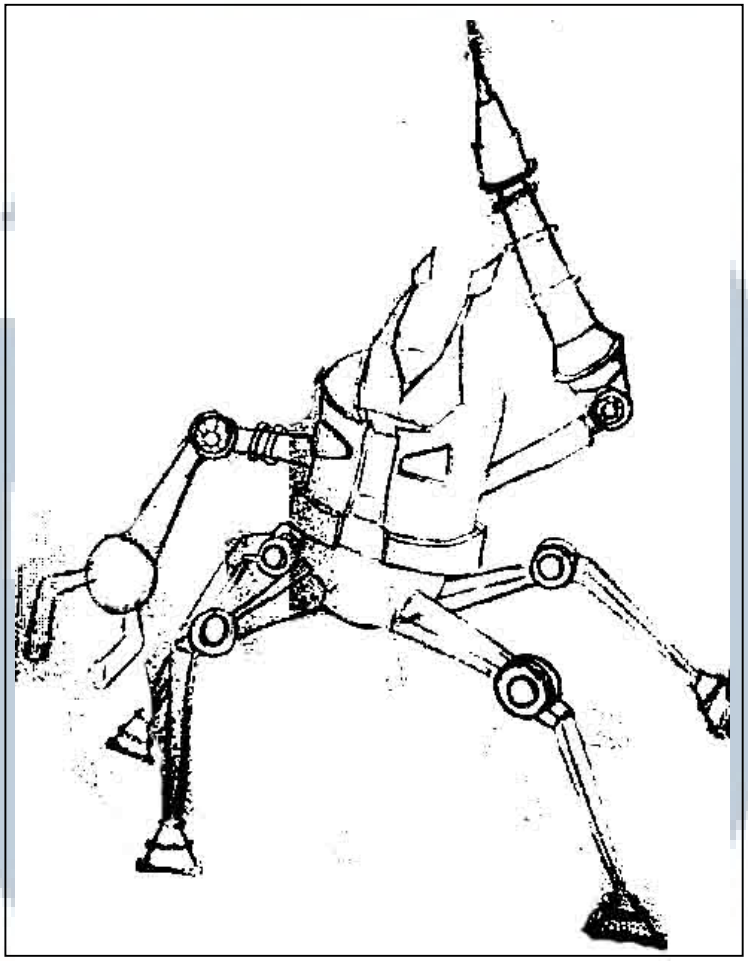
Nama	Robot Tipe A
HP	200
SP	100
Strength	5
Vitality	5
Intelligence	5
Wisdom	5

Tabel 3.5 *Skill* Robot Tipe A

***Skill* Robot Tipe A**	**Kemungkinan Digunakan**	**Akurasi Serangan**	**Konsumsi SP**	***Damage***	**Keterangan**
Normal Attack	60%	80%	0	1x	*Physical attack*, serangan standar.
Stun	20%	40%	12	0.3x	*Magical attack*, membuat target terkena *status ailment stun*.
Confuse	20%	40%	12	0.3x	*Magical attack*, membuat target terkena *status ailment confuse*.

Tabel 3.6 *Item Drop* dari Robot Tipe A

***Item Drop* dari Robot Tipe A**	**Keterangan**
Potion	*Drop rate* 30%.
Ether	*Drop rate* 10%.
Antidote	*Drop rate* 20%.
Elixir	*Drop rate* 3%.
Money	*Amount dropped*: 500-900



Gambar 3.6 Desain Robot Tipe B oleh Saudari Renata

Robot Tipe B adalah robot yang juga sering ditemui oleh Vla selama permainan. Robot ini memiliki *stats* yang rendah dan sangat lemah dalam hal *physical attack*, baik dalam menyerang maupun dalam bertahan. Robot ini dapat muncul sendirian saja atau berpasangan dengan sesama Robot Tipe B. Robot ini juga bisa muncul berpasangan dengan Robot Tipe A. Meskipun lemah, robot ini cukup menyulitkan untuk pemain karena robot ini dapat menyebabkan *status ailment poison,* yang akan terus bertahan pada *player character* meskipun *battle* telah usai selama *player character* tidak menyembuhkan *status ailment* ini dengan menggunakan *item*.

Tabel 3.7 *Stats* Robot Tipe B

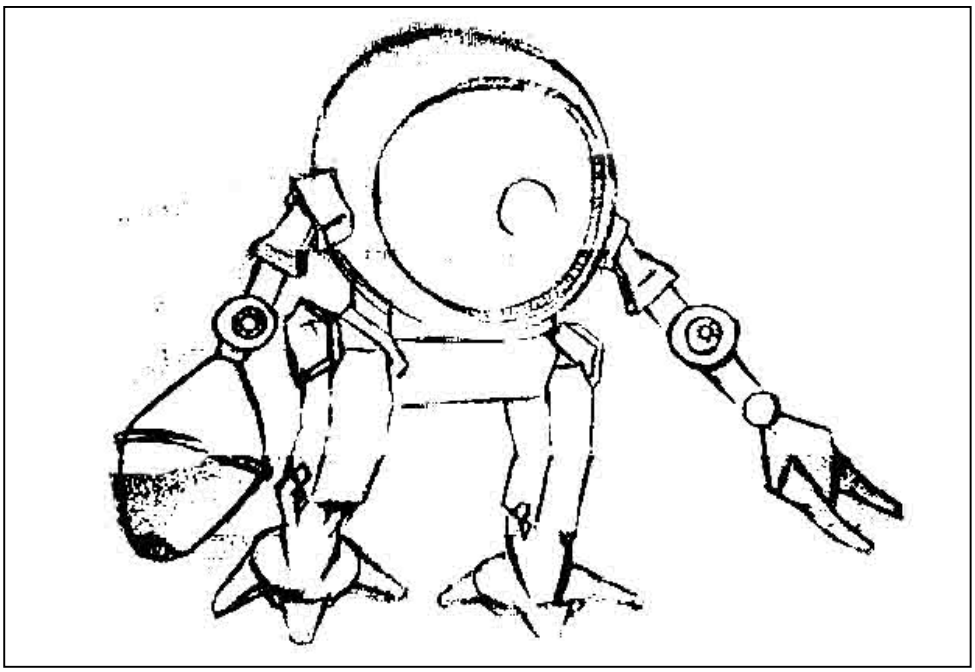
Nama	Robot Tipe B
HP	200
SP	100
Strength	1
Vitality	2
Intelligence	2
Wisdom	3

Tabel 3.8 *Skill* Robot Tipe B

***Skill* Robot Tipe B**	**Kemungkinan Digunakan**	**Akurasi Serangan**	**Konsumsi SP**	***Damage***	**Keterangan**
Normal Attack	60%	70%	0	1x	*Physical attack*, serangan standar.
Poison	10%	30%	16	0.5x	*Magical attack*, membuat target terkena *status ailment poison.*
Stun	30%	40%	12	0.3x	*Magical attack*, membuat target terkena *status ailment stun.*

Tabel 3.9 *Item Drop* dari Robot Tipe B

***Item Drop* dari Robot Tipe B**	**Keterangan**
Potion	*Drop rate* 30%.
Ether	*Drop rate* 10%.
Antidote	*Drop rate* 20%.
Elixir	*Drop rate* 3%.
Money	*Amount dropped*: 500-900



Gambar 3.7 Desain Robot Tipe C oleh Saudari Renata

Robot Tipe C adalah robot yang terkuat yang terdapat dalam *game* ini, yang kemungkinan munculnya dalam *random battle* lebih rendah jika dibandingkan dengan Robot Tipe A dan Robot Tipe B. Robot ini berbentuk mirip dengan Robot Tipe A, tetapi dengan menggunakan *color scheme* yang berbeda. Robot Tipe C memiliki kekuatan serangan dan pertahanan yang cukup kuat, baik dalam *physical attack* maupun *magical attack*.

Robot ini dapat menyebabkan semua jenis *status ailments* kepada *player character*: *stun, confuse,* dan *poison*. Robot ini kuat dan mungkin sangat menyulitkan bagi pemain, tetapi dengan mengalahkan robot ini pemain bisa mendapatkan *item* yang lebih baik dan uang yang lebih banyak jika dibandingkan dengan robot-robot tipe lainnya.

Berbeda dengan Robot Tipe A dan Robot Tipe B yang dapat menyerang *player character* secara berpasangan, Robot Tipe C hanya muncul sendirian tanpa berpasangan dengan robot-robot lainnya.

Tabel 3.10 *Stats* Robot Tipe C

Nama	Robot Tipe C
HP	500
SP	200
Strength	7
Vitality	7
Intelligence	7
Wisdom	7

Tabel 3.11 *Skill* Robot Tipe C

***Skill* Robot Tipe C**	**Kemungkinan Digunakan**	**Akurasi Serangan**	**Konsumsi SP**	***Damage***	**Keterangan**
Normal Attack	50%	80%	0	1x	*Physical attack*, serangan standar.
Special Attack	29%	80%	29	1.5x	*Physical attack*, lebih kuat dari serangan standar.
Stun	7%	30%	12	0.3x	*Magical attack*, membuat target terkena *status ailment stun*.
Confuse	7%	30%	12	0.4x	*Magical attack*, membuat target terkena *status ailment confuse*.
Poison	7%	50%	16	0.6x	*Magical attack*, membuat target terkena *status ailment poison*.

Tabel 3.12 *Item Drop* dari Robot Tipe C

***Item Drop* dari Robot Tipe C**	**Keterangan**
Potion	*Drop rate* 5%.
Ether	*Drop rate* 10%.
Antidote	*Drop rate* 5%.
Elixir	*Drop rate* 40%.
Bracelet	*Drop rate* 40%.
Money	*Amount dropped*: 10000-14000

Vincent adalah tokoh antagonis dalam *game* ini, dan juga berlaku sebagai *final boss* yang harus dikalahkan untuk menamatkan *game*. Sebagai *final boss*, Vincent memiliki *stats* yang jauh lebih tinggi dibandingkan robot-robot yang dapat ditemui pemain dalam *random battle*. Vincent dapat menyebabkan *status ailments stun* dan *confuse*, dan memiliki tiga buah serangan khusus yang dapat menyebabkan *damage* cukup besar kepada *player character*.

Gambar 3.8 Desain Karakter Vincent oleh Saudari Renata

Tabel 3.13 *Stats* Vincent

Nama	Vincent
HP	1000
SP	400
Strength	15
Vitality	11
Intelligence	14
Wisdom	12

Tabel 3.14 *Skill* Vincent

***Skill* Vincent**	**Kemungkinan Digunakan**	**Akurasi Serangan**	**Konsumsi SP**	***Damage***	**Keterangan**
Normal Attack	45%	90%	0	1x	*Physical attack*, serangan standar.
Special Attack 1	15%	80%	12	1.2x	*Magical attack*, serangan langsung kepada pemain tanpa menggunakan *status ailment*.
Special Attack 2	15%	80%	19	1.4x	*Magical attack*, serangan langsung kepada pemain tanpa menggunakan *status ailment*.
Special Attack 3	5%	100%	31	2x	*Magical attack*, serangan langsung kepada pemain tanpa menggunakan *status ailment*. Serangan terkuat *final boss* dalam *game* ini.
Stun	10%	50%	12	0.5x	*Magical attack*, membuat target terkena *status ailment stun*.

Tabel 3.15 *Skill* Vincent (lanjutan)

***Skill* Vincent**	**Kemungkinan Digunakan**	**Akurasi Serangan**	**Konsumsi SP**	***Damage***	**Keterangan**
Confuse	10%	50%	12	0.5x	*Magical attack*, membuat target terkena *status ailment confuse*.

Vincent tidak memiliki kemungkinan untuk men-*drop item* karena Vincent adalah *final boss* dalam *game* ini, yang mana *game* akan tamat setelah Vincent dikalahkan.

Nilai yang dituliskan dalam *field Damage* pada tabel-tabel *skill* untuk setiap karakter di atas menunjukkan berapa besar *damage* yang dapat disebabkan oleh *skill* tersebut jika dibandingkan dengan perhitungan *damage* dengan rumus-rumus perhitungan *damage* yang ditetapkan sebelumnya. Misalnya untuk Robot Tipe B yang memiliki poin *Strength* 1, *Vitality* 2, *Intelligence* 2, dan *Wisdom* 3, maka serangan dengan tipe *physical* dengan 1x *damage* akan menyebabkan *damage* pada target sebesar:

1**Damage* = *7***Strength* + *Weapon Attack Bonus*

1**Damage* = 7*1 + 0

1**Damage* = 7

Damage = 7

Jika Robot Tipe B menggunakan serangan *Poison* yang menyebabkan *magical damage* dengan 0.5x *damage*, maka serangan tersebut akan menyebabkan *damage* pada target sebesar:

0.3**Damage* = *7***Intelligence* + *Weapon Attack Bonus*

0.3**Damage* = 7*2 + 0

0.3**Damage* = 14

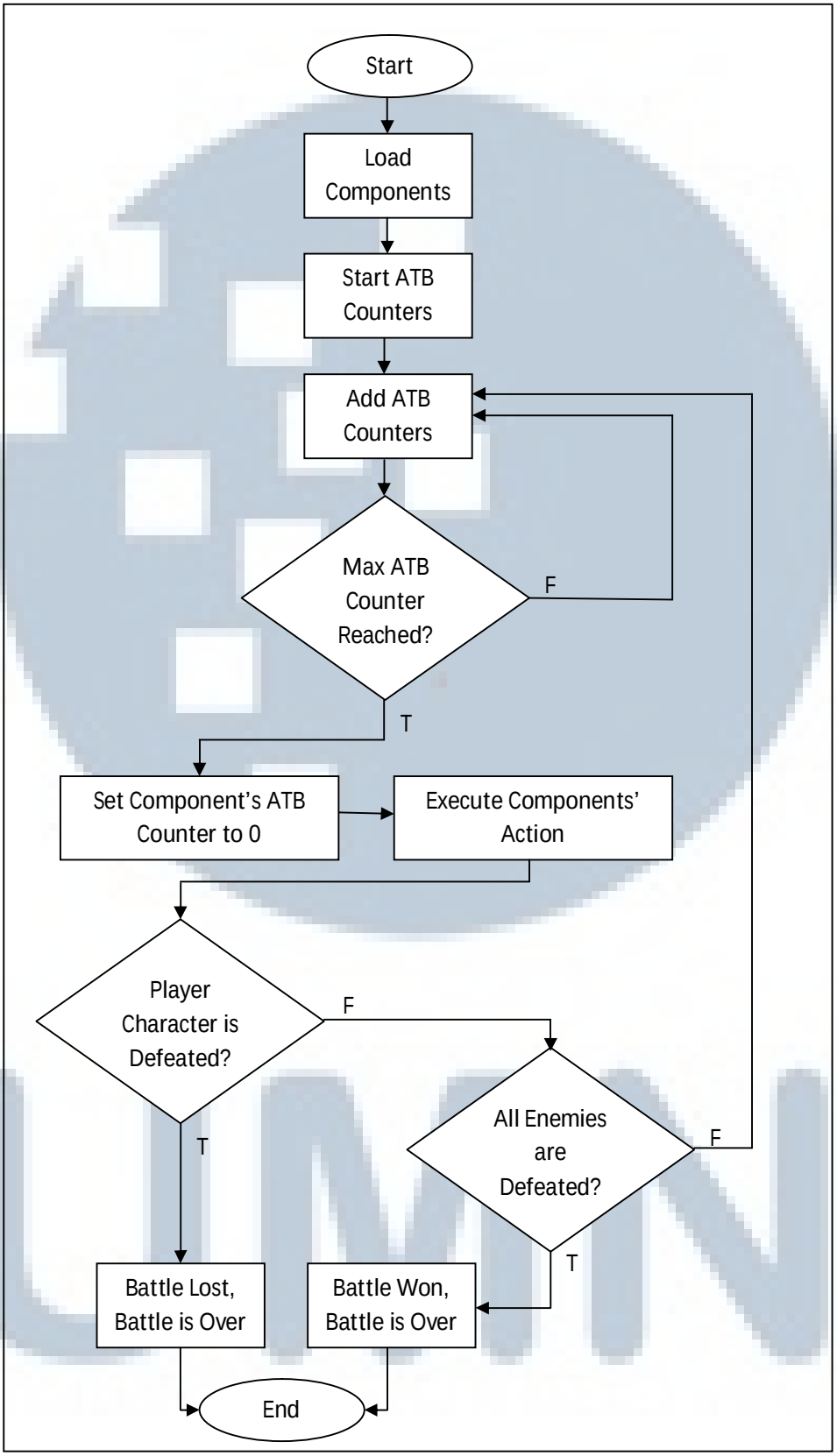
Damage = 70/3

Damage = 23.33 → *Damage* = 23

3.4.3 *Flowchart*

Battle system yang dikembangkan akan menjalankan *ATB counter* begitu masuk ke *battle scene*. Masing-masing karakter yang muncul di dalam *battle scene* memiliki *ATB counter* masing-masing. Pada saat *ATB counter* mereka mencapai nilai maksimum, mereka dapat menyerang atau menggunakan kemampuan yang mereka miliki dalam *battle.*

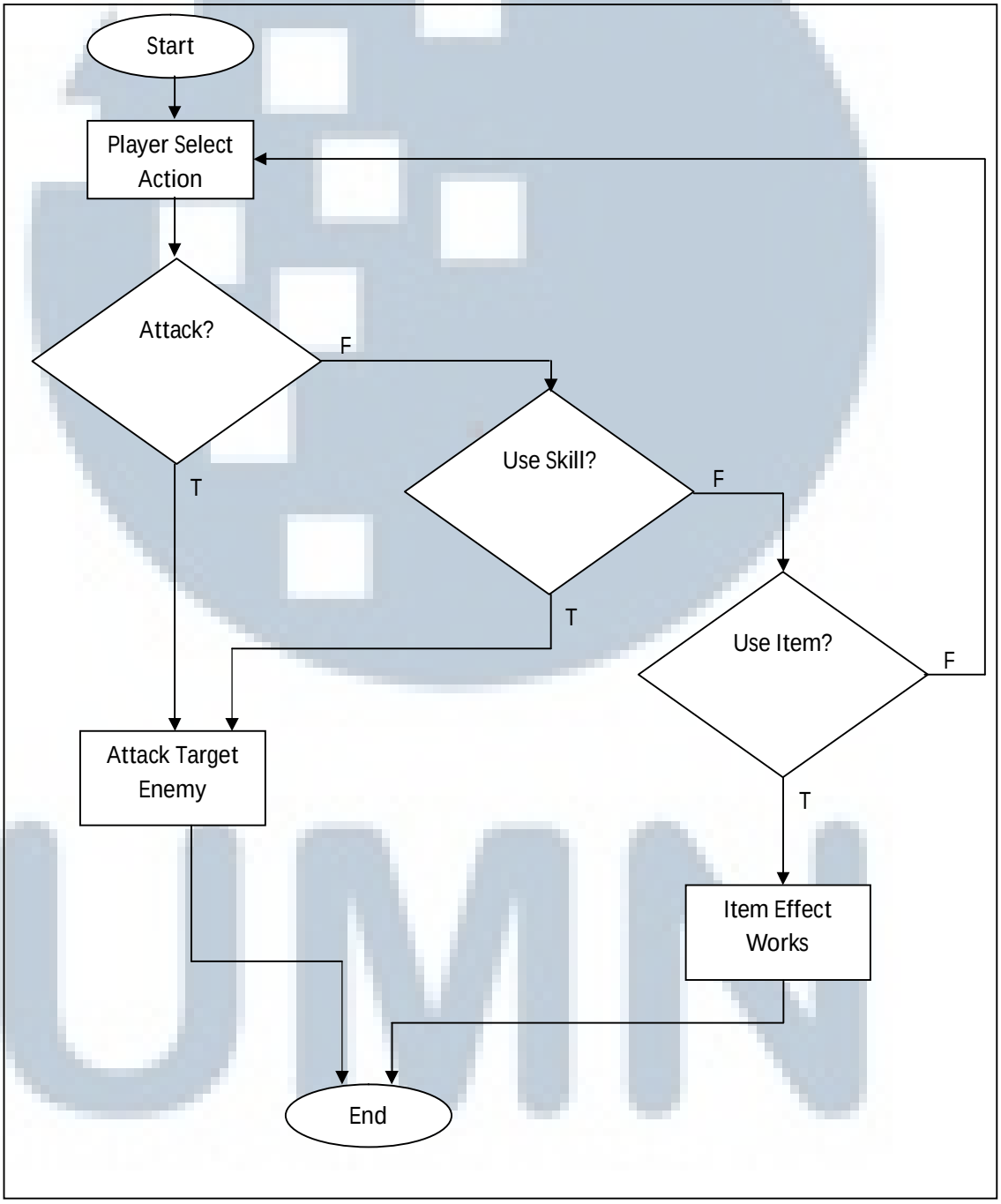
Alur *battle system* ini, jika dibuat dalam bentuk *flowchart*, dapat digambarkan dengan menggunakan *flowchart* berikut ini.



Gambar 3.9 *Flowchart* dari *Battle System* Secara Umum

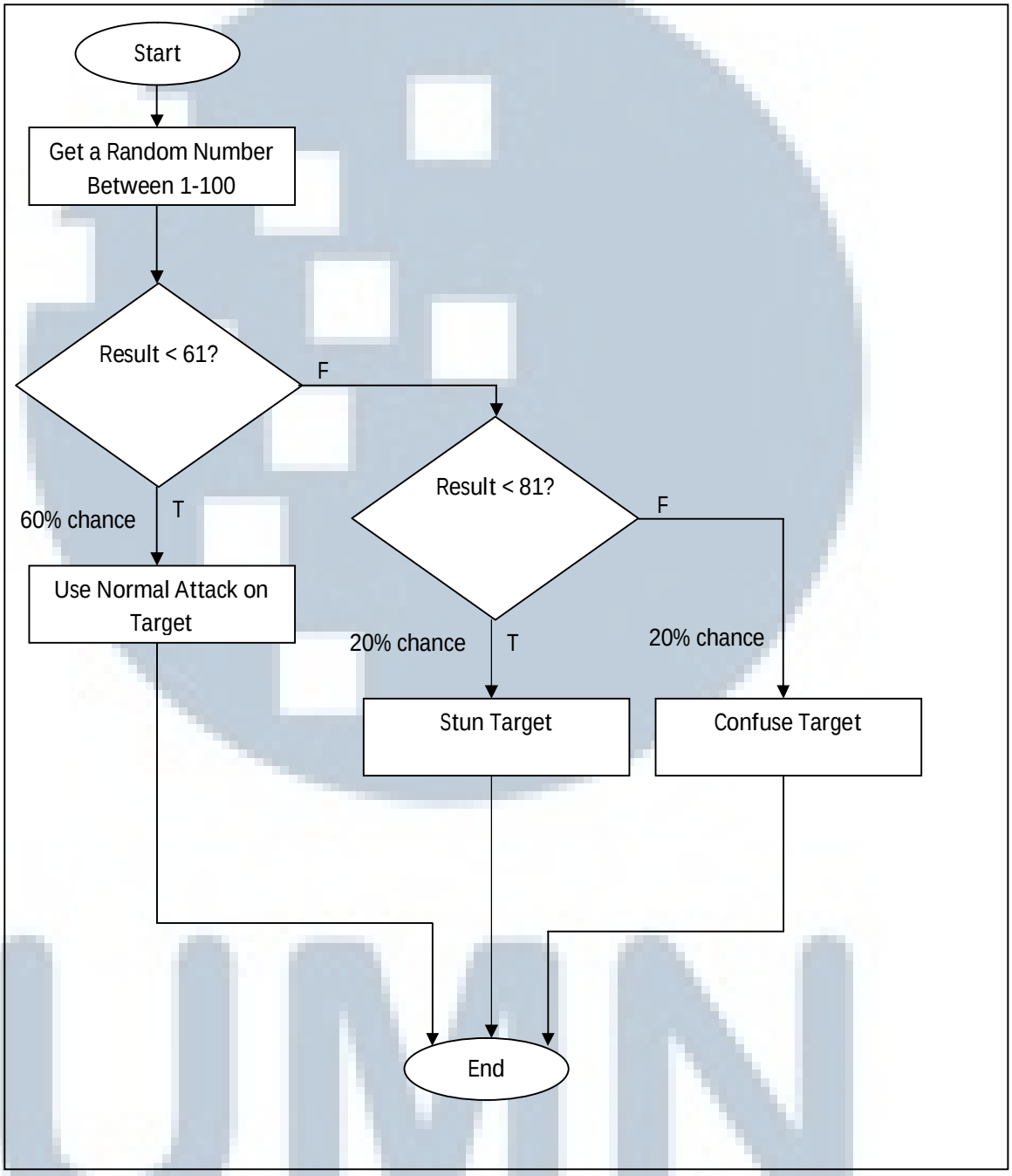
Dalam *flowchart* sebelumnya, terdapat komponen proses yang bernama "*Execute Components' Action*". Komponen proses tersebut dapat dibuat ke dalam *flowchart* baru untuk masing-masing karakter yang terdapat di dalam *game* ini.

Berikut ini gambaran *flowchart* dari tindakan-tindakan yang dapat dilakukan oleh karakter Vla dalam *battle*.



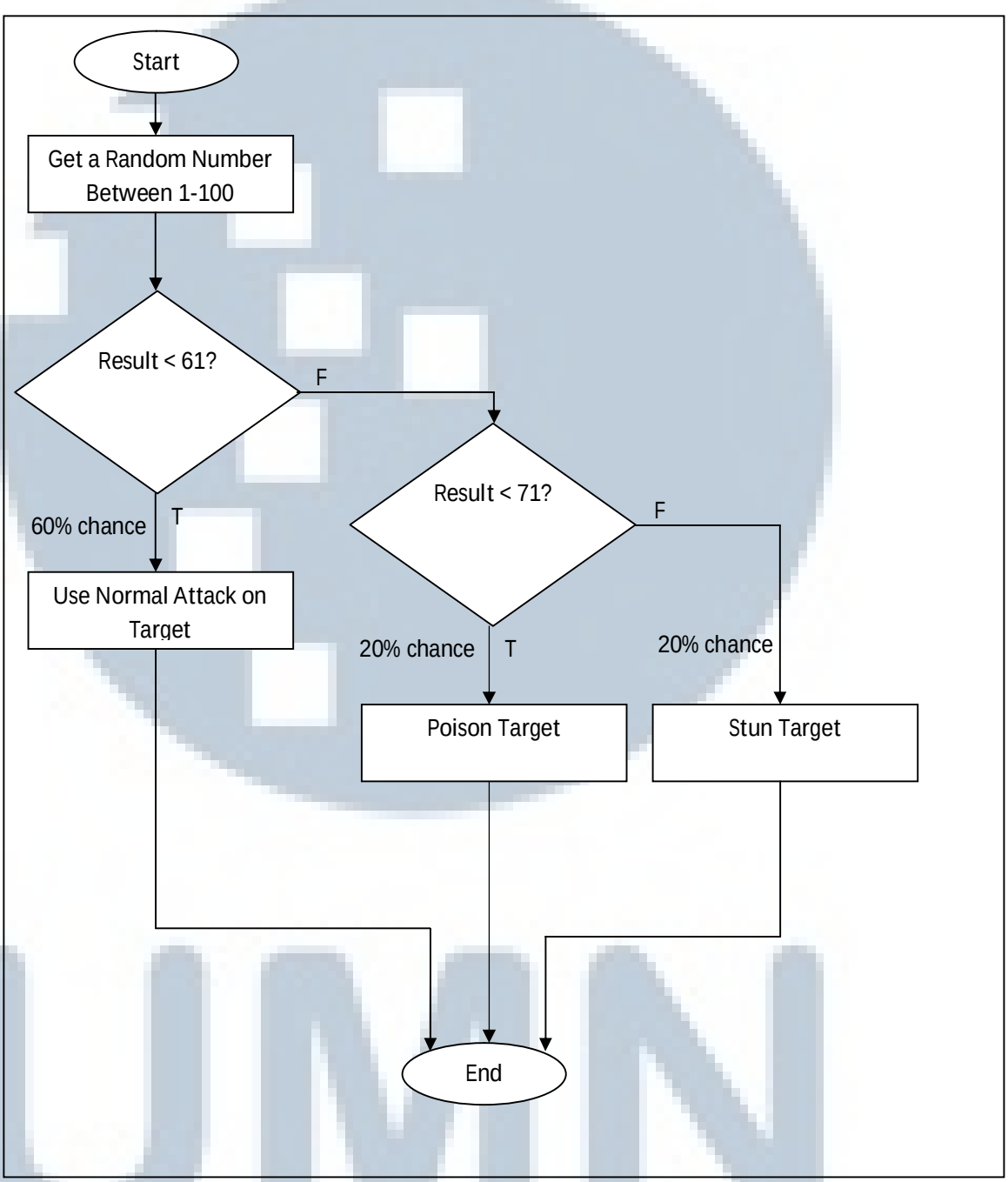
Gambar 3.10 *Flowchart* tindakan Vla

Flowchart untuk tindakan Robot Tipe A bisa digambarkan dalam *flowchart* seperti berikut ini.



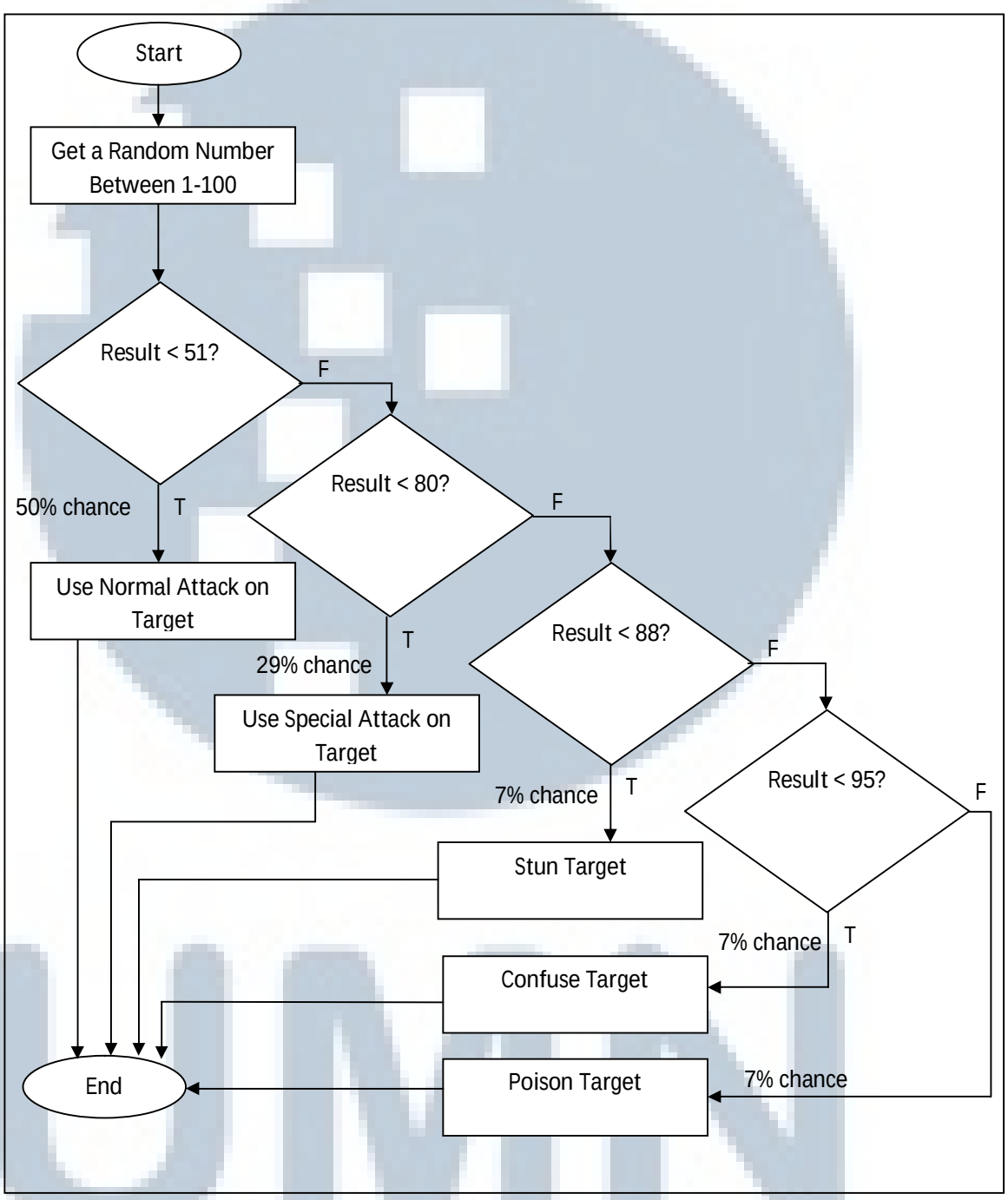
Gambar 3.11 *Flowchart* tindakan Robot Tipe A

Sementara untuk Robot Tipe B, *flowchart* untuk tindakannya dapat kita gambarkan sebagai berikut.



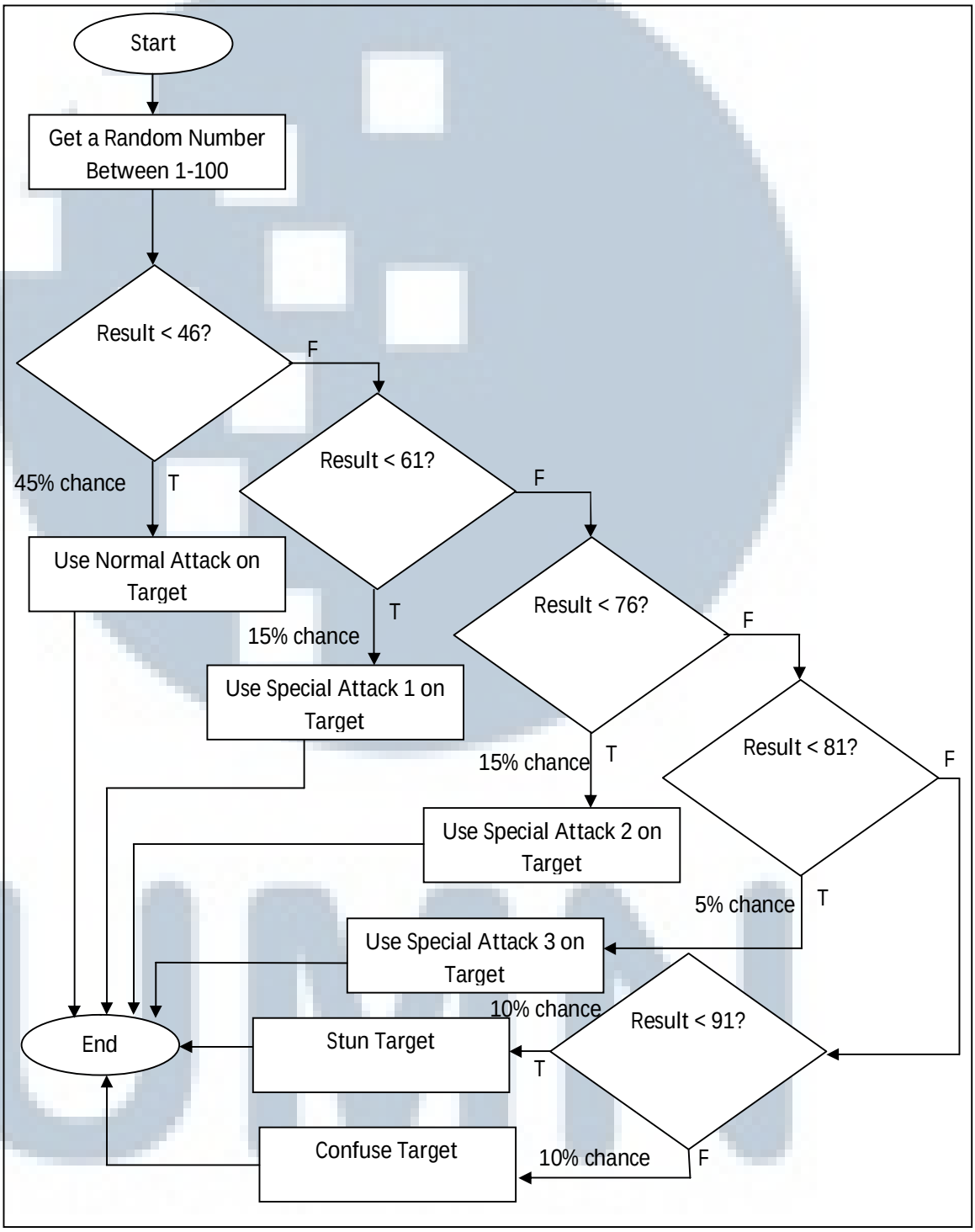
Gambar 3.12 *Flowchart* tindakan Robot Tipe B

Flowchart untuk pengambilan tindakan oleh Robot Tipe C dapat digambarkan seperti *flowchart* di bawah ini.



Gambar 3.13 *Flowchart* tindakan Robot Tipe C

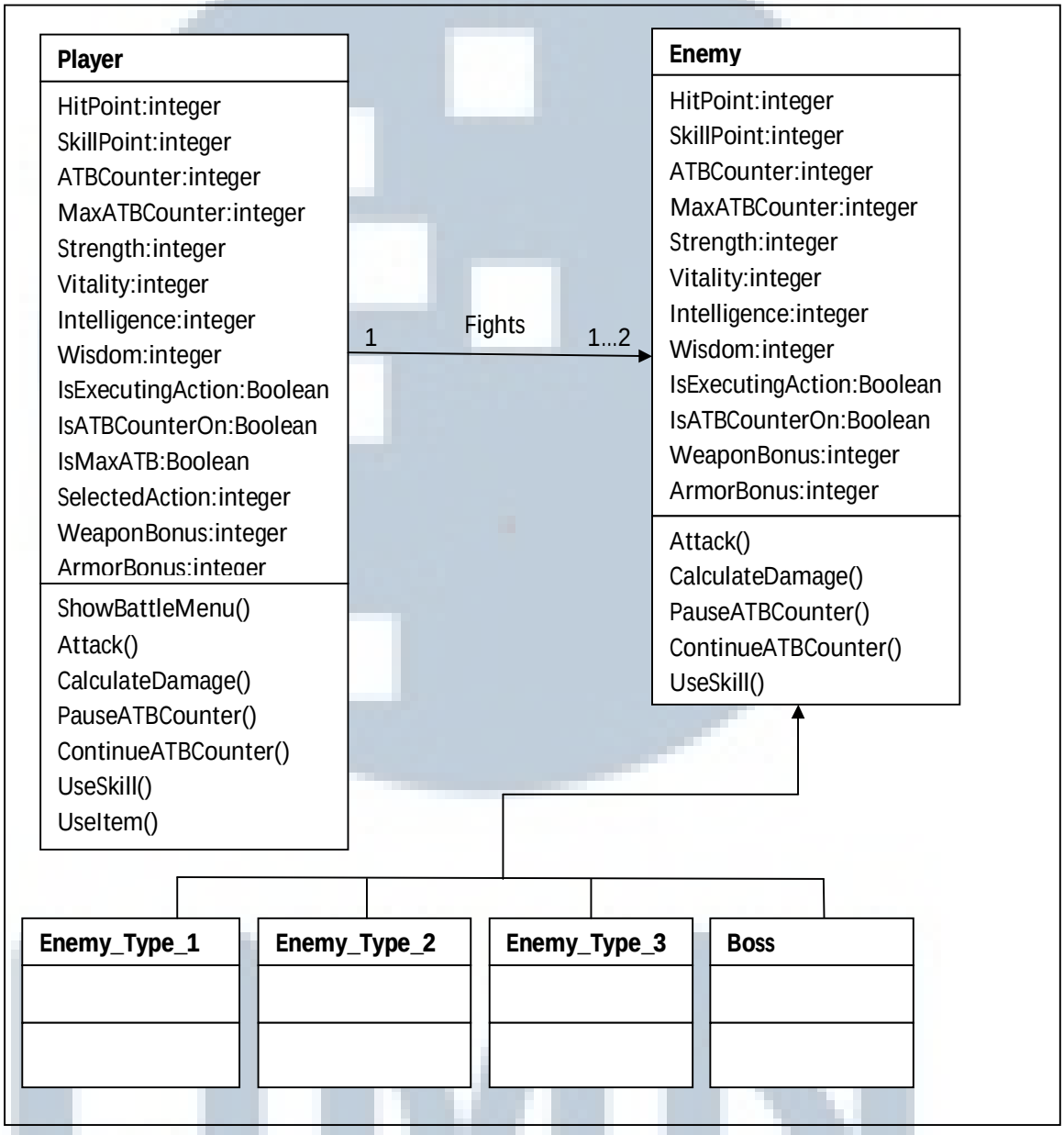
Berikut ini *flowchart* untuk tindakan-tindakan yang diambil oleh *final boss* di dalam *game* ini, Vincent.



Gambar 3.14 *Flowchart* tindakan Vincent, *final boss* dalam *game* ini

3.4.4 *Class Diagram*

Class-class yang digunakan dalam pembuatan *battle system* ini secara umum dapat digambarkan ke dalam *class diagram* sebagai berikut.



Gambar 3.15 *Class Diagram* dari *Battle System*

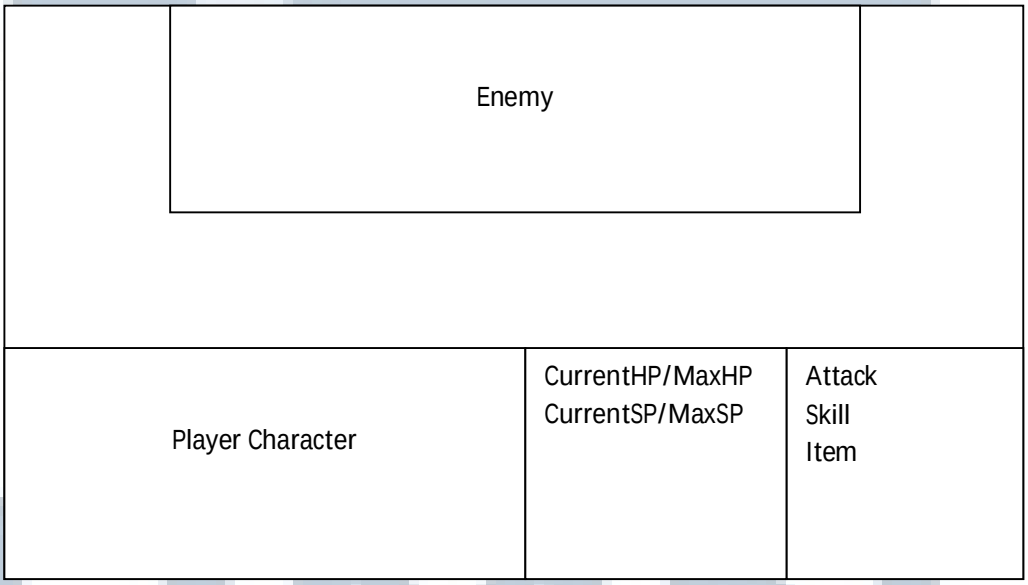
Class diagram tersebut menunjukkan gambaran desain *class* beserta *property* dan *method* yang terdapat di dalamnya dan bagaimana setiap objek

dalam *battle system* yang bersangkutan akan berinteraksi. Dalam proses implementasi, *property* dan *method* yang berada dalam *class* menjadi sedikit berbeda karena peserta magang harus mengikuti aturan *scripting* standar pada Unity, di mana terdapat *method-method* standar dari *class GameObject*, yang merupakan *class* induk dari objek-objek yang dibuat dalam game, yang harus diimplementasikan dengan benar agar *game* dapat berjalan.

3.4.5 Laporan Implementasi

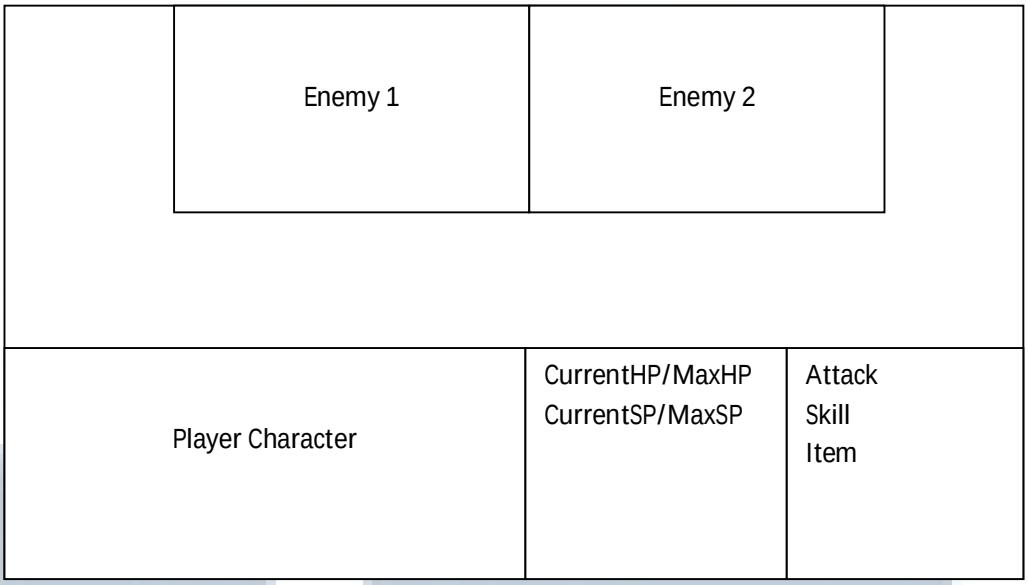
3.4.5.1 Desain Tampilan

Berikut ini rancangan *user interface battle system* yang dibuat oleh peserta magang yang dijadikan acuan dalam proses implementasi *battle system game* ini pada *game engine* Unity.



Gambar 3.16 Desain *Battle System* untuk 1v1

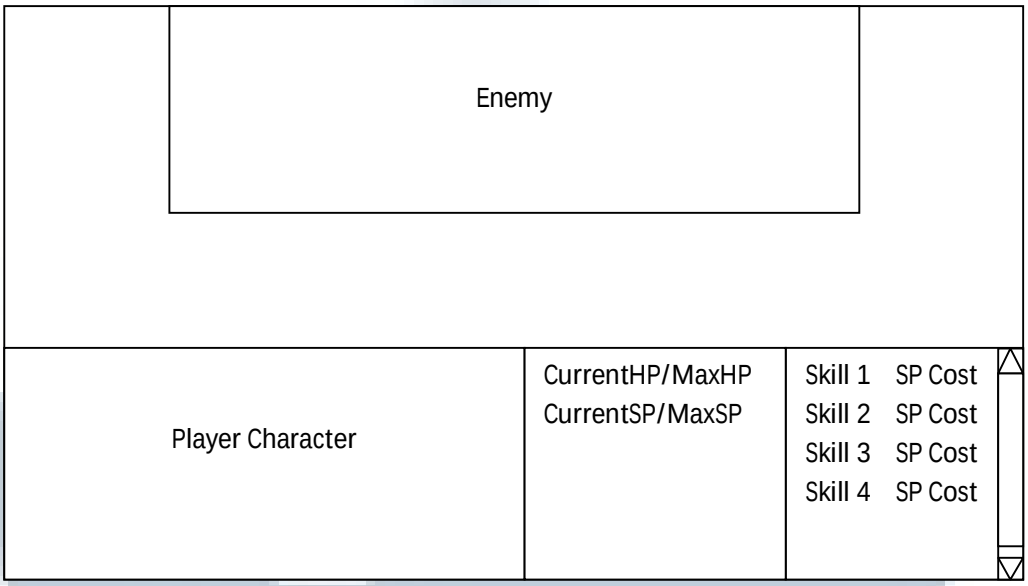
Dalam *battle system* game ini akan ditampilkan *player character*, musuh yang dihadapi, nilai *hit point* dan *skill point* milik *player character*, dan juga *battle menu* yang berisi daftar tindakan yang dapat dilakukan oleh *player character* dalam *battle*.



Gambar 3.17 Desain *Battle System* untuk 1v2

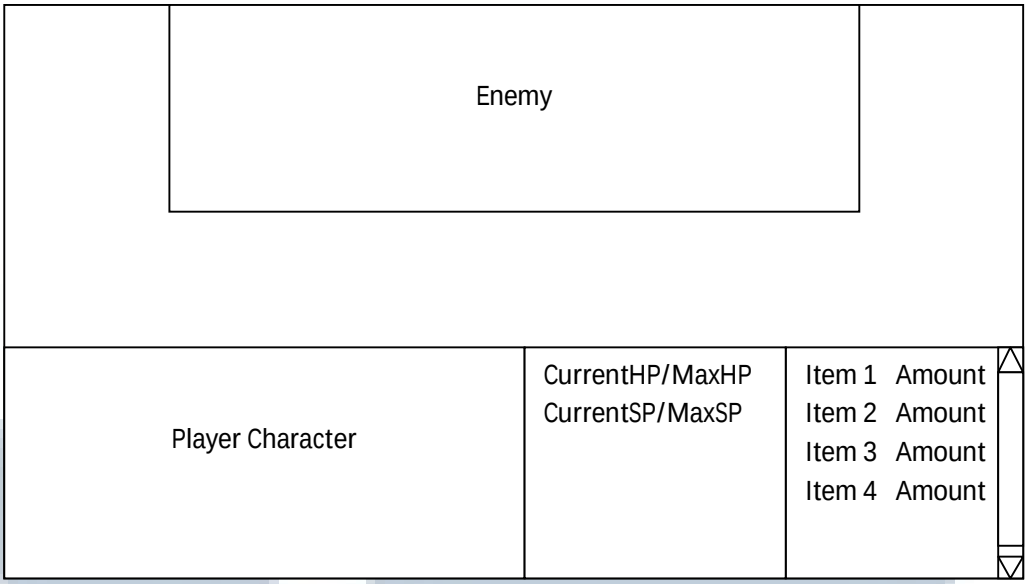
Pada saat menghadapi dua musuh, pemain akan mendapatkan tampilan *user interface* yang kurang lebih sama dengan tampilan yang diberikan untuk *battle* dengan satu musuh. Kedua musuh akan diposisikan bersebelahan dalam *battle*, berseberangan dengan *player character*.

Dalam *battle menu* terdapat tiga pilihan, yaitu *attack*, *skill*, dan *item*. Jika pemain memilih menu *attack*, maka *player character* akan langsung menyerang musuh yang dijadikan sebagai target dari serangan yang dilakukan.



Gambar 3.18 Jika Menu *Skill* Dipilih

Jika pemain memilih menu *skill*, maka akan ditampilkan daftar *skill* yang dimiliki oleh *player character* dan konsumsi SP untuk *skill* tersebut. Jika *player character* tidak memiliki cukup SP untuk menggunakan *skill* suatu *skill*, maka *skill* tersebut tidak dapat digunakan oleh *player character*. Jika *player character* memiliki cukup SP, maka *skill* tersebut akan digunakan dan jumlah SP milik *player character* akan dikurangi sejumlah nilai konsumsi SP dari *skill* tersebut.



Gambar 3.19 Jika Menu *Item* Dipilih

Jika menu *item* dipilih, maka *game* akan menampilkan daftar *item* yang dimiliki oleh *player character* dan jumlah *item* tersebut yang berada di dalam *inventory*. *Item* yang berjumlah nol dalam *inventory* tidak akan ditampilkan dalam daftar *item* karena *player character* tidak memiliki *item* tersebut.

3.4.5.2 Implementasi Sistem ATB

Setelah desain *user interface dummy battle system* selesai dibuat oleh peserta magang, peserta magang mulai mengimplementasikan sistem ATB yang digunakan untuk mengatur waktu dan giliran menyerang untuk setiap karakter yang terdapat dalam *battle scene*.

Berikut ini tampilan *screenshot* dari *dummy battle scene* yang telah memiliki sistem ATB.

Gambar 3.20 Tampilan *Dummy Battle System*

Gambar di atas menunjukkan kondisi saat *player character* (kiri) melawan satu musuh dalam *battle scene*. Pada sebelah kanan bawah layar terlihat *battle menu*. Di posisi di mana seharusnya nilai HP dan SP dari *player charater* ditampilkan, terlihat status *ATB meter* milik *player character* (dicetak tebal) dan *ATB meter* milik musuh (ditampilkan di bawah *ATB meter* milik *player character*). Peserta magang menempatkan *ATB meter* di sana karena pada tahap ini peserta magang belum mengimplementasikan perhitungan *damage* dan penggunaan *skill* oleh *player character*, sehingga posisi itu dapat digunakan untuk menampilkan *ATB meter* agar hasil implementasi dan kesalahan-kesalahan dalam perhitungan ATB dan giliran menyerang dapat terlihat oleh peserta magang.

Karena model-model untuk *player character* dan musuh belum selesai dibuat oleh tim desainer, pada tahap ini peserta magang menggunakan *dummy model* yang telah diberikan sebelumnya untuk digunakan sebagai model untuk *player character* dan musuh.

Gambar 3.21 *ATB Meter Player Character* Penuh

Pada saat *ATB meter* dari *player character* penuh, akan muncul sebuah tanda panah pada *battle menu* Tanda panah tersebut dapat digerakkan untuk memilih tindakan yang akan dilakukan oleh *player character* pada saat itu. Pada saat penulisan laporan ini, menu *skill* dan *item* belum dapat berfungsi sehingga *player character* hanya bisa memilih *attack* pada *battle menu.*

Gambar 3.22 *Player Character* Menyerang

Jika pemain memilih *attack*, *player character* akan berlari ke arah musuh untuk menyerang, kemudian kembali lagi ke posisinya semula.

Gambar 3.23 Musuh Menyerang

Saat *ATB* meter musuh mencapai maksimum, musuh akan mendekati *player character* untuk menyerang dan kemudian kembali ke tempatnya semula. Pada tahap ini, musuh hanya bisa menyerang dengan serangan standar karena *skill* untuk musuh belum diimplementasikan ke dalam *script*.

3.4.5.3 Implementasi Kalkulasi *Damage* dan *Status Ailments*

Setelah mendapatkan dokumen desain dari Saudari Cut Astrid Zuhra mengenai pembagian *stats* untuk *player character* dan musuh-musuh yang dihadapi dalam *battle*, peserta magang mulai mengimplementasikan perhitungan kalkulasi *damage* dan *status ailments*.

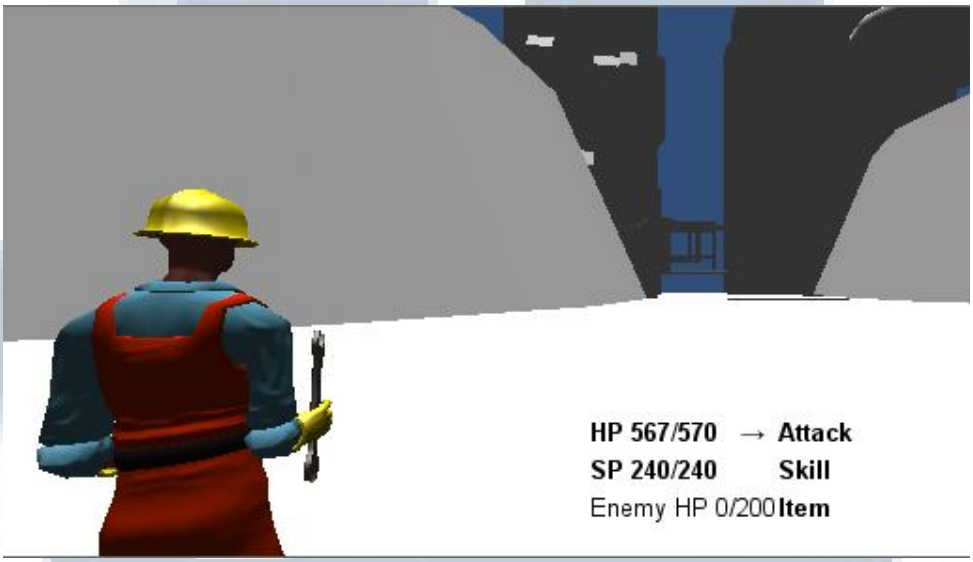
Berikut ini *screenshot* tampilan hasil implementasi untuk kalkulasi *damage*.

Gambar 3.24 Tampilan *Battle System* dengan Tampilan Nilai HP dan SP

Pada tahap ini, *battle* sudah dapat diselesaikan dengan cara menghabiskan HP milik musuh atau *player character*. Pada saat HP milik suatu karakter mencapai nol, karakter tersebut akan menghilang dari *scene*. Jika salah satu pihak sudah kalah, maka *battle* selesai.

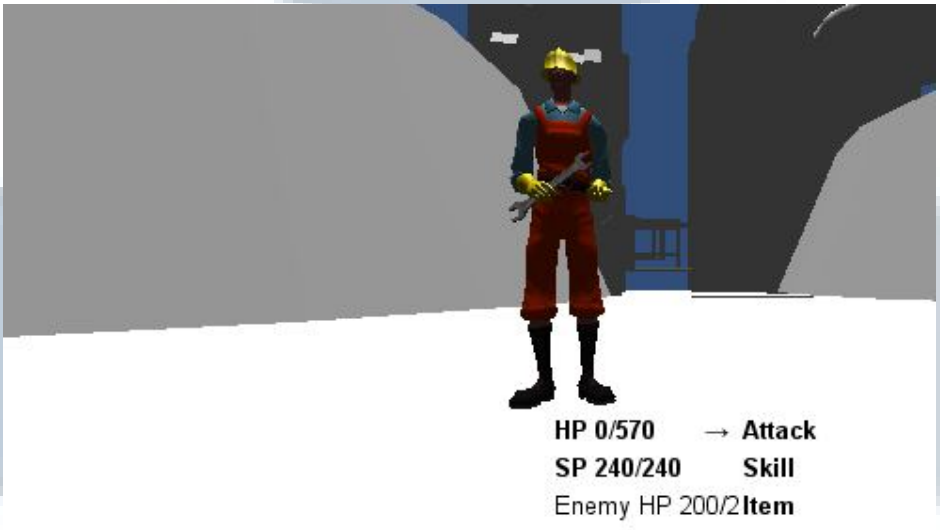
Gambar 3.25 Nilai HP *Player Character* dan Musuh Setelah Saling Menyerang

Player character dan musuh akan kalah dan menghilang dari *battle scene* jika nilai HPnya mencapai nol. Berikut ini tampilan *dummy battle scene* pada saat *player character* berhasil mengalahkan musuh.



Gambar 3.26 Musuh Menghilang Setelah Dikalahkan

Jika musuh berhasil mengalahkan *player character*, *dummy battle scene* akan terlihat seperti berikut.



Gambar 3.27 *Player Character* Dikalahkan

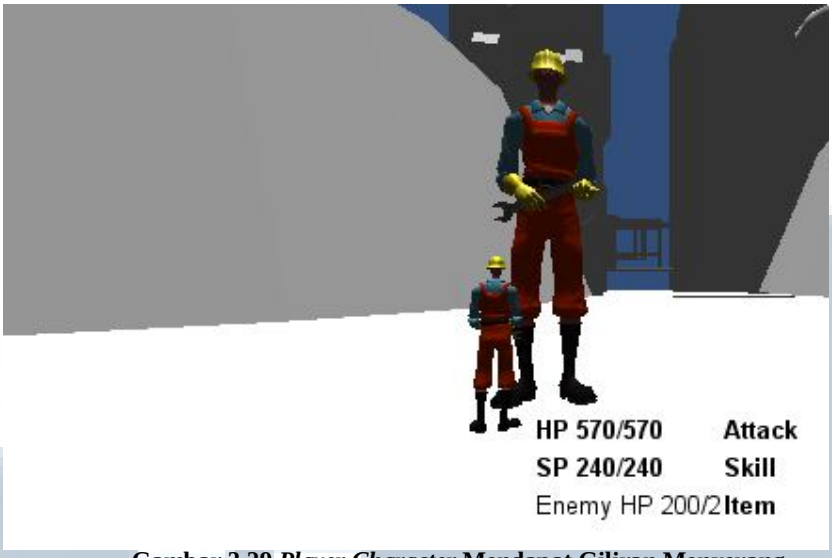
Pada tahap ini, peserta magang telah membuat *script* untuk *skill* milik Robot Tipe A, yang dipasangkan ke model *dummy* musuh yang ditampilkan dalam *screenshot* di atas, dan juga *script* untuk efek dua dari tiga *status ailment* yang terdapat di dalam *game* ini, yaitu *stun* dan *poison*. Kedua *status ailment* tersebut sudah dapat berjalan di tahap ini, tetapi masih terdapat *bug* untuk *status ailment poison*, di mana efek *poison* akan terhenti jika *status ailment stun* sedang aktif pada karakter yang bersangkutan.

3.4.5.4 Simulasi

Berikut ini simulasi sebuah *battle* antara *player character* melawan Robot Tipe A dengan menggunakan *dummy*, dari awal *battle* hingga Robot Tipe A berhasil dikalahkan.

Gambar 3.28 *Battle* antara *Player Character* dan Robot Tipe A Dimulai

Pemain menggerakkan *player character* pada saat menjelajahi kota. Setelah beberapa langkah, pemain masuk ke dalam *random battle* menghadapi sebuah Robot Tipe A.



Gambar 3.29 *Player Character* Mendapat Giliran Menyerang

Player character lebih dulu mendapatkan giliran menyerang. Pemain menyerang dengan menggunakan menu *attack*, *player character* melancarkan serangan fisik standar kepada Robot Tipe A.

Gambar 3.30 Robot Tipe A Menyerang Setelah Menerima *Damage*

Robot Tipe A menerima 81 *damage* dari serangan yang dilancarkan oleh *player character* sebelumnya. Kemudian Robot Tipe A mendapatkan giliran untuk menyerang.

Gambar 3.31 Serangan Robot Tipe A Meleset

Serangan yang dilancarkan oleh Robot Tipe A gagal mengenai *player character*. Nilai *hit point* milik *player character* tidak berkurang.

Gambar 3.32 *Player Character* Kembali menyerang

Player character kembali mendapatkan giliran untuk menyerang Robot Tipe A.

Gambar 3.33 Robot Tipe A Terkena *Damage*

Player character kembali ke posisinya. Robot Tipe A yang terkena serangan dari *player character* mendapatkan 81 *damage*. Nilai *hit point* milik Robot Tipe A pun berkurang.

Gambar 3.34 *Player Character* Terkena *Damage*

Robot Tipe A menggunakan *skill confuse*. *Player character* hanya menerima 1 *damage*, karena *skill confuse* memiliki kekuatan serangan yang sangat

rendah. *Player character* tidak terkena *status ailment confuse* karena efek dari *status ailment* ini belum diimplementasikan oleh peserta magang pada tahap ini.

Gambar 3.35 Robot Tipe A Dikalahkan

Player character kembali mendapatkan kesempatan menyerang. Pemain memilih menu *attack*, dan *player character* menyerang Robot Tipe A. Robot Tipe A yang hanya memiliki *hit point* senilai 18 terkena *damage* sebesar 81 poin. Pemain sukses mengalahkan Robot Tipe A dan menyelesaikan *battle*.

3.4.5.5 Hasil Akhir

Berikut ini gambar-gambar yang diambil dari tampilan hasil akhir *battle system* yang telah dibuat. Hasil akhir ini diperoleh setelah pekerjaan dilanjutkan pada pertengahan Januari 2013 sampai dengan akhir Februari 2013.

Gambar 3.36 Tampilan Akhir *Battle System*

Layout tampilan dari *battle system* ini sedikit berubah dari *dummy battle system* yang dibuat sebelumnya. Fungsi untuk menu *Skill* dan *Item* telah ditambahkan.

Gambar 3.37 Tampilan Menu *Skill*

Pada saat menu *Skill* dipilih, daftar *Skill* yang dapat digunakan dan konsumsi *Skill Point* dari *skill* yang bersangkutan akan ditampilkan. Jika *Skill Point* dari *player character* tidak mencukupi, maka tidak akan terjadi apa-apa pada saat *Skill* tersebut dipilih oleh pemain.

Gambar 3.38 Tampilan Menu *Item*

Pada saat menu *Item* dipilih, daftar *item* yang dimiliki oleh *player character* dan jumlah *item* tersebut akan ditampilkan di layar. Jika jumlah *item* tidak nol, berarti *item* tersebut dapat digunakan oleh *player character*.

3.4.6 Pengujian, Perbaikan, dan Koordinasi

Berikut ini tahapan pengujian yang dilakukan untuk memastikan implementasi *script* pada *game* dilakukan dengan benar.

1. Peserta magang akan menguji sendiri *script* yang telah dibuat untuk memastikan *script* tersebut berjalan dengan benar. Jika terdapat *bug* dalam program, maka peserta magang akan mencari penyebab dari *bug* tersebut dan memperbaikinya.
2. Pada saat *script* sudah dapat berjalan dengan benar, peserta magang akan menghubungi Saudari Dyahayu Tyaswening dan/atau Saudari Cut Astrid Zuhra untuk melaporkan kemajuan yang telah dicapai. Jika dibutuhkan, peserta magang akan mendemonstrasikan hasil pekerjaan yang telah dibuat. Jika terdapat kesalahan dalam implementasi *battle system* yang tidak sesuai dengan desain yang ditentukan, maka kesalahan tersebut akan diperbaiki. Jika implementasi dianggap sudah benar, maka peserta magang akan diberikan lanjutan dari dokumen desain yang dibuat oleh tim desainer dan membantu mengoreksi dan menambahkan jika terdapat kesalahan dalam dokumen desain.

3. Setiap minggunya peserta magang akan melaporkan hasil pekerjaannya kepada Bapak Yusup Martyastiadi sebagai pemimpin proyek dan melakukan konsultasi jika terdapat masalah dalam pengembangan *battle system*.

3.4.7 Sarana yang Digunakan

Sarana-sarana yang digunakan oleh peserta magang untuk pengembangan *battle system game* ini adalah sebagai berikut.

1. *Game engine* Unity versi 3.1.0f4, digunakan oleh peserta magang untuk penempatan model, meng-*generate* objek-objek dalam *game*, dan meng-*assign script* yang sudah dibuat ke objek yang sudah ditentukan.
2. JavaScript, digunakan sebagai bahasa *scripting* utama dalam pengembangan *game* ini.
3. UnityScript Editor, digunakan sebagai aplikasi *text editor* utama dalam proses *scripting* dan implementasi *game* yang bersangkutan.
4. "Unity 3.x Game Development Essentials: Game development with C# and JavaScript" yang ditulis oleh Will Goldstone, digunakan sebagai buku panduan utama oleh peserta magang untuk *self-tutorial* dalam mempelajari *game engine* Unity.
5. Model-model *dummy* yang disediakan oleh tim desainer untuk digunakan untuk *scripting battle system* sebelum tim desainer menyelesaikan model-model yang akan digunakan untuk *game* sebenarnya.
6. Contoh *game* buatan Bapak Yusup Martyastiadi dan tim desainer dalam proyek-proyek sebelumnya untuk dipelajari.
7. Dropbox digunakan oleh tim pengembang untuk berbagi dokumen dan *file* yang diperlukan untuk pengembangan.
8. Facebook, digunakan sebagai media komunikasi utama di antara tim pengembang.

3.5 Kendala yang Ditemukan

Kendala yang ditemukan oleh peserta magang selama proses pengembangan *battle system* untuk *game* ini antara lain sebagai berikut.

1. Proses pembelajaran untuk dapat menggunakan Unity. Hal ini cukup menyulitkan karena peserta magang belum pernah menggunakan *game engine* sebelumnya dan tidak cukup familiar dengan aplikasi-aplikasi untuk desain, yang memiliki cukup banyak kemiripan dengan Unity pada *user interface*-nya.
2. Penyeimbangan waktu yang dialokasikan untuk kerja magang dan mengerjakan proyek-proyek yang diberikan oleh dosen untuk tugas kuliah cukup merepotkan, karena tugas kuliah yang diberikan pada saat periode magang terhitung sangat banyak meskipun jumlah SKS perkuliahan yang diambil jauh lebih sedikit dibandingkan dengan semester-semester sebelumnya. Hal ini membuat rencana alokasi waktu yang telah ditetapkan sebelum kerja magang dimulai menjadi sedikit berantakan karena kondisi yang berbeda dengan estimasi peserta magang sebelum memulai proses kerja magang.

3.6 Solusi Atas Kendala yang Ditemukan

Solusi atas kendala – kendala yang ditemukan oleh peserta magang selama periode kerja magang adalah sebagai berikut.

1. Bertanya kepada orang-orang yang lebih berpengalaman menggunakan *game engine* Unity. Peserta magang mendapatkan banyak bantuan dari Bapak Yusup Martyastiadi, Saudara Jason Christian, dan tim desainer untuk dapat mengerti bagaimana sebuah *game engine* bekerja, terutama Unity.
2. Masalah kepadatan waktu ini sedikit sulit untuk diatasi karena peserta magang tidak dapat menolak tugas-tugas yang diberikan oleh dosen. Meskipun kendala ini tidak dapat benar-benar diatasi, peserta magang mencoba untuk menyeimbangkan pengerjaan tugas-tugas kuliah dan kerja magang agar keduanya dapat berjalan dengan baik.