Network	

BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Kedudukan penulis dalam pelaksanaan kerja magang ini berada di bawah pengawasan dari *Tokyo Denki University*. Penulis kemudian ditetapkan sebagai bagian dari mahasiswa *Tokyo Denki University* dan berperan sebagai mahasiswa internasional jangka pendek di universitas tersebut.

Koordinasi pelaksanaan kerja magang dibawah pengawasan Prof. Osamu Shigo selaku kepala dari laboratorium *software engineering Tokyo Denki University*. Selain itu, koordinasi kerja magang juga dibimbing oleh Bapak Shizuga selaku *executive member of CNA (Career Net Auction, Inc.)* .

3.2 Tugas yang Dilakukan

Dalam program kerja magang, penulis ditetapkan sebagai anggota dari tim pada laboratorium rekayasa piranti lunak. Projek pengembangan *game* ini bertujuan untuk mengembangkan permainan berbasis *Android* bernama "*Avenger of the Inn"*

untuk perangkat *Android* . Dalam hal ini, penulis bertanggung jawab atas bagian *Artificial Intelligence*. Dalam pembuatannya penulis bekerjasama dengan Michael Setiawan Suhardjono selaku *battle system developer*.

Pertama, tim melakukan pertemuan tentang permainan *(game)* apa yang akan dikembangkan. Kesimpulan yang didapat dalam *game requirements* adalah akan dikembangkan permainan *simulation RPG* yang diberi nama "*Avenger of the Inn*", dengan minimal versi yang dibutuhkan adalah *Android* 2.2. Target konsumen yang diinginkan adalah berumur 10-30 tahun. Setelah memutuskan *game requirements*, *game design*, dan peran untuk setiap anggota tim, pengembangan *game* dimulai. *Development Environment* yang dipakai adalah *Eclipse Software Development Kit* (SDK).

3.3 Uraian Pelaksanaan Kerja Magang

3.3.1 Proses Pelaksanaan

Untuk mengembangkan *Artificial Intelligence* bagi permainan ini, peraturan permainan harus dimengerti oleh penulis. Berikut adalah penjelasan mengenai peraturan permainan :

1. Permainan "*Avenger of the Inn*" merupakan permainan *Strategy Role-playing Game* yang terdiri atas banyak cell (kotak) seperti papan catur. Sampel permainan dapat dilihat pada Gambar 3.1.

Gambar 3.1 Sampel Permainan

2. Tujuan utama dari permainan ini adalah mengalahkan semua musuh atau menghancurkan kotak harta karun.

3. Masing - masing tipe dari *monster* (musuh) mempunyai pola serangan dan jarak gerak yang berbeda - beda. Terdapat 4 jenis *monster*, yaitu .

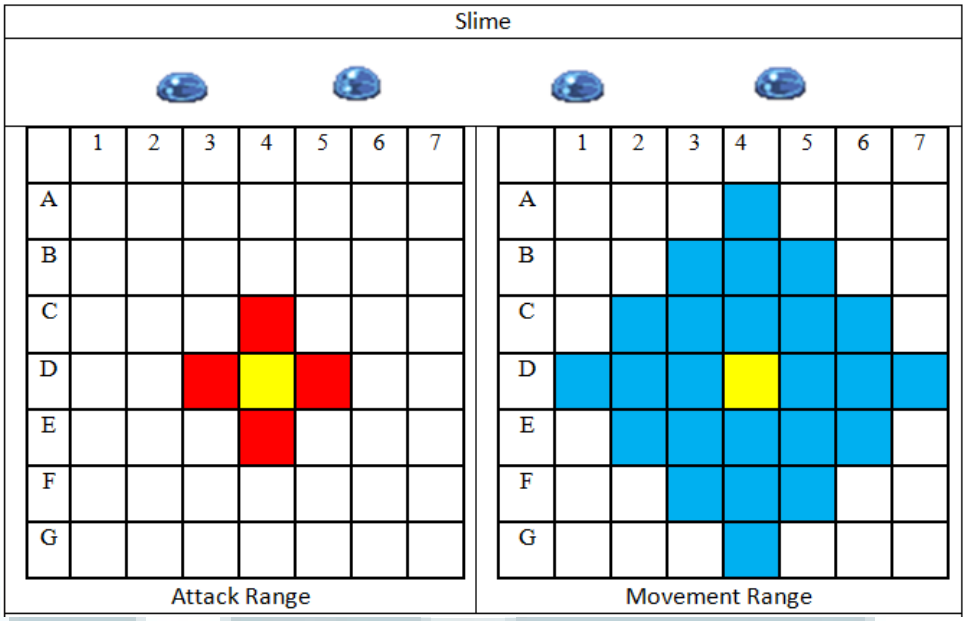
a. *Slime* (Gambar 3.2)

b. *Bee* (Gambar 3.3)

c. *Wolfman* (Gambar 3.4)

d. *Goblin* (Gambar 3.5)

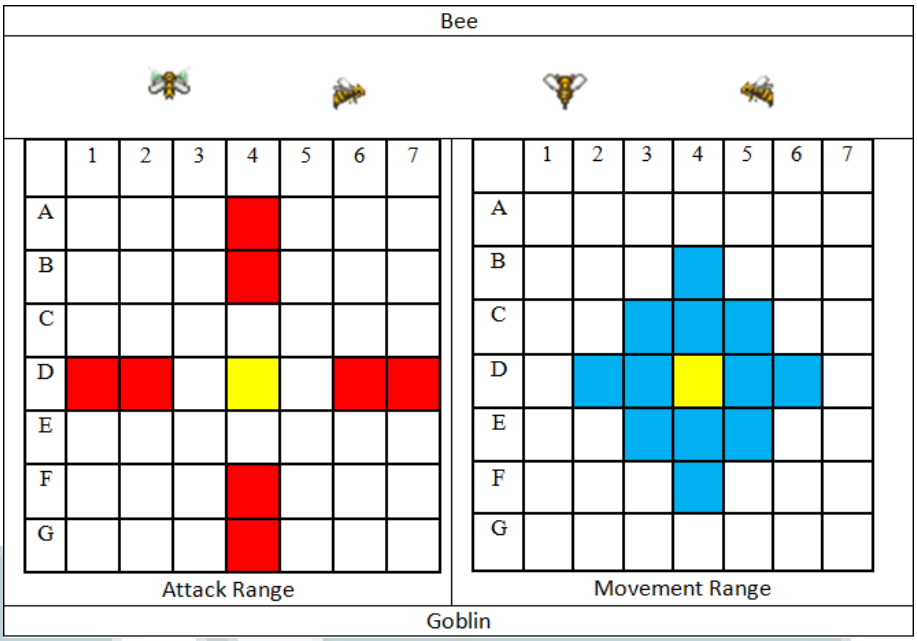
Jarak gerak dan pola serangan dari setiap *monster* dapat dilihat dari perwakilan gambar tiap *monster* pada halaman selanjutnya.



Gambar 3.2 *Slime Pattern*

Pada Gambar 3.2 dapat dilihat bahwa *monster slime* hanya memiliki 4 jarak serang yang pendek, yaitu C4, D3, D5, dan E4. D4 adalah posisi *monster slime* berada. Untuk jarak gerak dapat dilihat bahwa *monster slime* memiliki jarak gerak 3 *cells* di sekeliling posisi awal (D4). Dimulai dari A4 sampai G4, dan D1 sampai D7, kotak yang diwarnai merupakan jarak gerak dari monster *slime*.

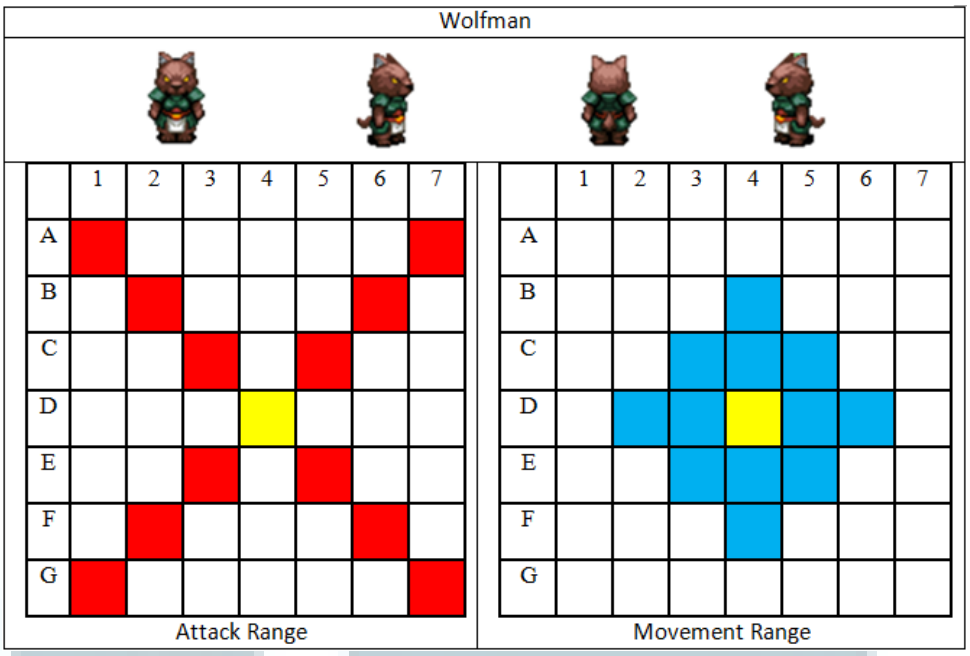
Dengan pola jarak serang seperti ini, *monster slime* harus berjalan ke sebelah pemain agar dapat melakukan aksi serang. Oleh karena itu *monster slime* memiliki jarak gerak paling luas diantara semua *monster* lainnya.



Gambar 3.3 *Bee Pattern*

Monster bee memiliki jarak serang yang unik. Bertolak belakang dengan *monster slime*, *monster bee* tidak dapat menyerang pemain yang berada di sebelah *monster bee*. Dapat dilihat jarak serang *monster bee* adalah A4, B4, D1, D2, D6, D7, F4, dan G4. D4 adalah posisi dimana *monster bee* berada. Untuk jarak gerak, *monster bee* hanya memiliki jarak gerak 2 *cells* di sekeliling posisi awal (D4). Dimulai dari B4 sampai F4 dan D2 sampai D6, kotak yang diwarnai merupakan jarak gerak *monster bee*.

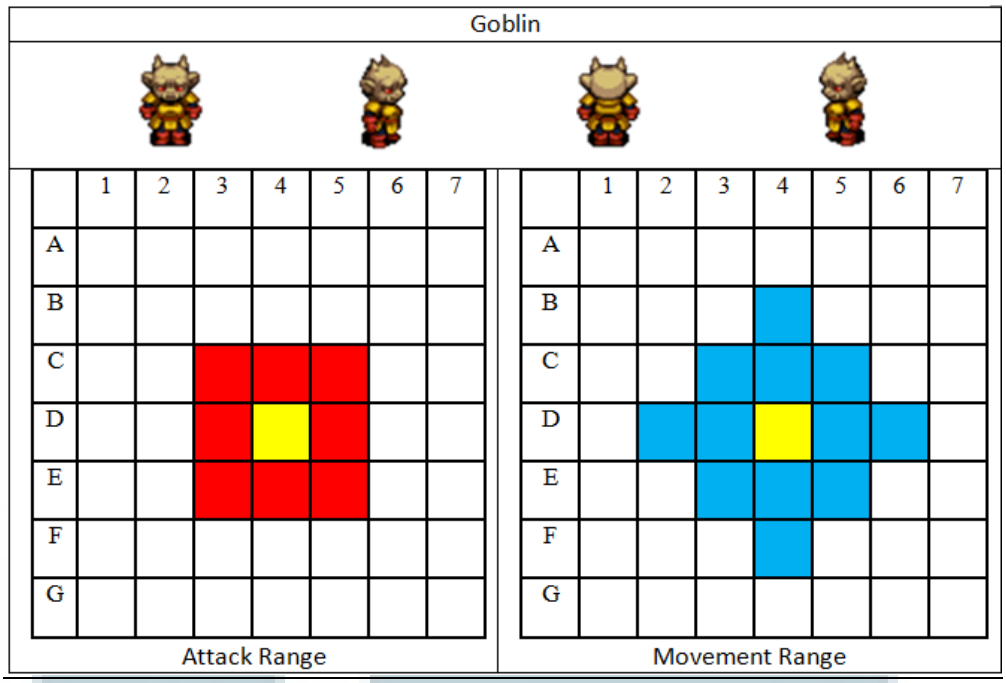
Walaupun hanya memiliki jarak gerak 2 *cells* dari posisi awal, jarak serang yang unik memungkinkan *monster bee* untuk melakukan aksi serang dari jarak jauh, bahkan dari belakang *monster* lainnya. Kombinasi ini membuat *monster bee* dapat menjangkau semua pemain yang berada di dalam jarak respon *monster(3 cells* dari posisi awal).



Gambar 3.4 *Wolfman Pattern*

Monster Wolfman memiliki jarak serang berpola huruf kapital X. Oleh karena itu, *monster wolfman* tidak dapat melakukan aksi serang dalam posisi lurus dari posisi awal. Jarak serang *monster wolfman* adalah A1, A7, B2, B6, C3, C5, E3, E5, F2, F6, G1, dan G7. D4 adalah posisi awal *monster wolfman*. Untuk jarak gerak, *monster wolfman* memiliki jarak gerak yang sama dengan *monster bee*, yaitu 2 *cells* dari posisi awal (D4). Dimulai dari B4 sampai F4 dan D2 sampai D6, kotak yang diwarnai merupakan jarak gerak *monster wolfman*.

Jarak serang *monster wolfman* mungkin merupakan jarak serang yang dapat menjangkau pemain paling jauh.Walaupun *monster wolfman* sedikit sulit dalam menjangkau pemain dengan posisi lurus dari *monster*, kombinasi jarak gerak dan jarak serang *monster wolfman* tanpa disangka dapat menjangkau pemain yang berada di posisi yang salah, sehingga *monster* ini merupakan monster yang kuat.



Gambar 3.5 *Goblin Pattern*

Berdasarkan *game design* yang dibuat bersama dengan tim pengembangan game ini, *monster goblin* direncanakan mempunyai serangan yang paling kuat diantara semua monster. Jarak serang *monster goblin* adalah tepat di sekeliling posisi awal *monster* berada (D4). Jarak serang tersebut adalah C3-C5, D3, D5, dan E3-E5. D4 adalah posisi awal *monster*. *Goblin* memiliki jarak gerak yang sama dengan *monster bee* dan *wolfman*, yaitu 2 *cells* dari posisi awal *monster* (D4). Dimulai dari B4 sampai F4 dan D2 sampai D6, kotak yang diwarnai merupakan jarak gerak *monster goblin*.

Monster goblin dirancang sebagai *monster* dengan serangan terkuat, sehingga apabila diberi kombinasi jarak gerak dan jarak serang yang berlebihan, akan merusak keseimbangan dalam permainan. Walaupun begitu, jarak serang disekeliling *monster* memastikan pemain yang terlalu dekat tidak akan lepas dari serangan *goblin*.

Berdasarkan aturan permainan yang telah dijelaskan, dapat ditentukan beberapa sifat cerdas bagi *monster* dan dengan algoritma *Greedy* disusunlah sifat cerdas *monster* sebagai berikut :

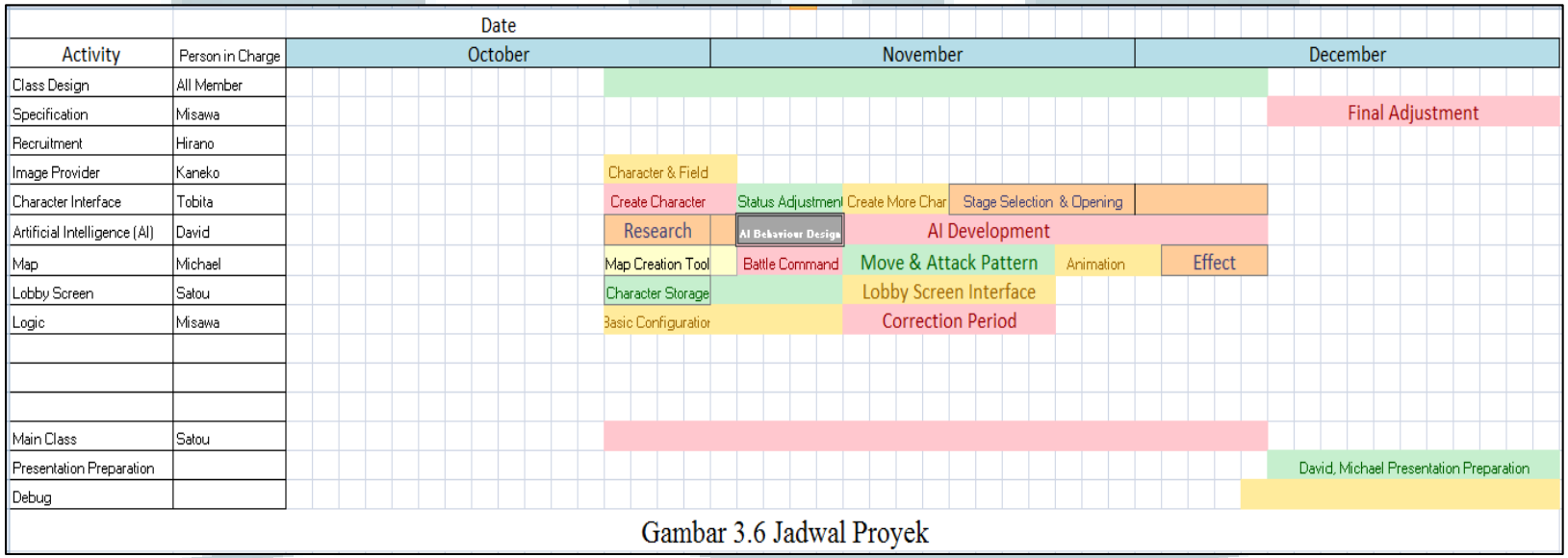
1. *Monster* tidak mengejar pemain, tetapi pemain yang harus mendekat kepada *monster* tersebut.
2. *Monster* memiliki tugas untuk melindungi kotak harta karun, maka dari itu setiap *monster* tidak boleh bergerak terlalu jauh dari kotak harta karun.
3. Karena jarak gerak maksimum dalam permainan adalah 3 *cells*, *monster* hanya akan merespon apabila pemain ada di dalam jarak 3 *cells* dari *monster* tersebut.
4. *Monster* hanya dapat berjalan sekali dan menyerang satu pemain setiap giliran.
5. Apabila terdapat pemain di dalam jarak serang *monster*, pemain tersebut yang akan diserang. Apabila tidak, *monster* akan menyerang pemain apabila pemain yang berada dalam kombinasi jarak serang dan jarak gerak *monster* tersebut.
6. *Monster* akan menyerang pemain pertama yang ditemukan dalam pemeriksaan jarak serang.

Tergantung dari pola serang dan jarak gerak, penempatan untuk *Artificial Intelligence (AI)* harus berbeda untuk tiap *monster*. Misalnya, " *Slime*" harus bergerak ke samping pemain untuk menyerang pemain tersebut. "*Bee*" memiliki jarak serang yang panjang, karena itu *monster* ini dapat menyerang di belakang posisi monster lain.

Pada proyek ini, proses pengembangan *AI* merupakan tahap terakhir dalam pengembangan aplikasi. Pengembangan *AI* harus tertunda sampai semua *class*

diagram yang dikembangkan tim sudah dapat diimplementasikan dan mengetahui cara untuk mengimplementasikan semua fungsi - fungsi yang ada. Oleh karena itu, disusunlah teori-teori mengenai AI oleh penulis sebelum memasuki tahap *coding*.

Jadwal pengerjaan dari proyek game ini dapat dilihat pada Gambar 3.6 pada halaman selanjutnya.



Gambar 3.6 Jadwal Proyek

Sebelum mengerjakan bagian *coding*, penulis mempelajari cara - cara untuk mengembangkan aplikasi *Android* menggunakan *Android Software Development Kit*. Penulis menggunakan *Eclipse* dan *Android SDK* sebagai *Development Kit* utama. Setelah *class diagram* sudah benar - benar *solid*, maka dimulailah pengembangan sifat-sifat AI pada *monster* berdasarkan aturan permainan.

3.3.2 Rumusan Masalah

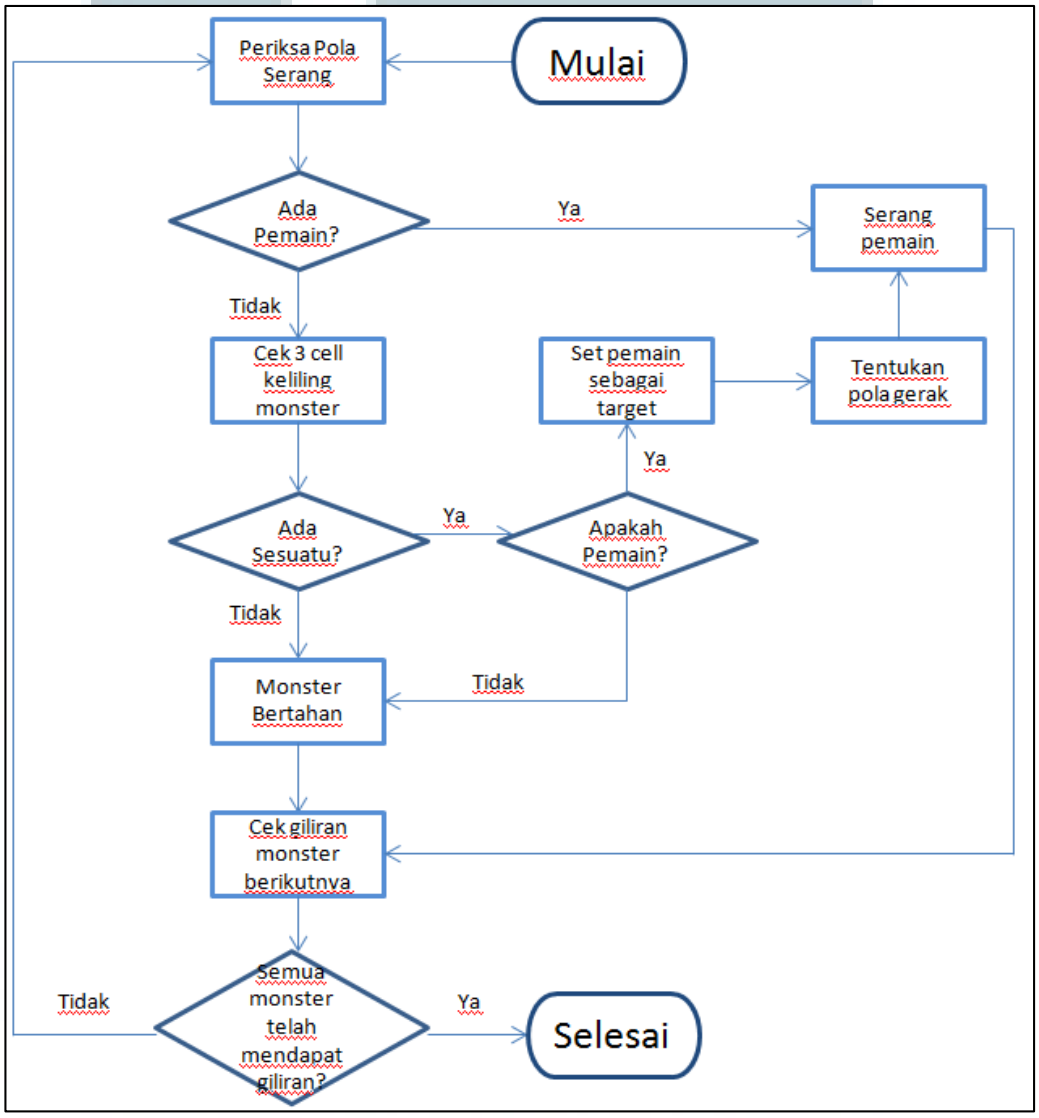
Permainan ini diharapkan dapat dimainkan secara individu. Maka dari itu diperlukan musuh untuk menjadi lawan permainan. Untuk menciptakan musuh yang memiliki sifat cerdas, maka perlu dikembangkan algoritma yang sesuai dengan logika permainan.

Tergantung dari peraturan-peraturan dari sebuah permainan, sifat kecerdasan buatan harus disesuaikan. Masalah yang muncul adalah cara menciptakan sifat cerdas yang membuat permainan lebih menarik. Sifat cerdas ini diharapkan ringan untuk dioperasikan oleh *Android*, tetapi harus tetap tangguh untuk dikalahkan.

Untuk mengembangkan sebuah sifat cerdas, pengembangan permainan ini harus sudah mencapai tahap tertentu dimana fungsi-fungsi yang ada dapat diimplementasikan oleh penulis supaya sifat cerdas menjadi lebih optimal. Sebelum itu, sifat cerdas hanya dapat dideskripsikan menggunakan *pseudocode.*

3.3.3 Pemecahan Masalah

Seperti yang sudah dijelaskan oleh penulis, setiap sifat cerdas berbeda di dalam setiap situasi, dan harus disesuaikan dengan permainan. Dengan mengimplementasi sifat cerdas berdasarkan pada peraturan yang ada pada permainan ini, berikut adalah pola aksi sifat cerdas yang digambarkan berupa *Flow Chart* pada Gambar 3.7.



Gambar 3.7 *Flow Chart* Sifat Cerdas *Avenger of the Inn*

Penjelasan *Flow Chart* pada Gambar 3.7 adalah sebagai berikut :

1. Pertama sekali *monster* yang memasuki gilirannya akan memeriksa apakah terdapat pemain yang berada di dalam pola serang *monster*. Apabila terdapat pemain dalam pola serang, maka *monster* akan menyerang pemain pertama yang terdeteksi, dimulai dari *cell* paling kiri atas. Giliran *monster* ini selesai.

2. Apabila tidak terdapat pemain dalam pola serang, maka *monster* akan melakukan pemeriksaan apakah terdapat pemain di dalam jarak 3*cells* di sekitar *monster*.

3. Jika terdapat pemain di dalam jarak 3 *cells* di sekitar *monster*, dilakukan pemeriksaan untuk setiap pola langkah *monster*. Apabila pola serang yang terdapat di dalam pola gerak *monster* tersebut mengandung pemain, maka *monster* akan berjalan ke pola gerak tersebut dan menyerang pemain.

4. Jika tidak ditemukan pemain di dalam jarak tersebut, monster akan melakukan aksi bertahan. Aksi bertahan akan mengurangi besar serangan pemain ke *monster* tersebut.

Aturan ini dapat diimplementasi dengan *pseudo-code* seperti berikut :

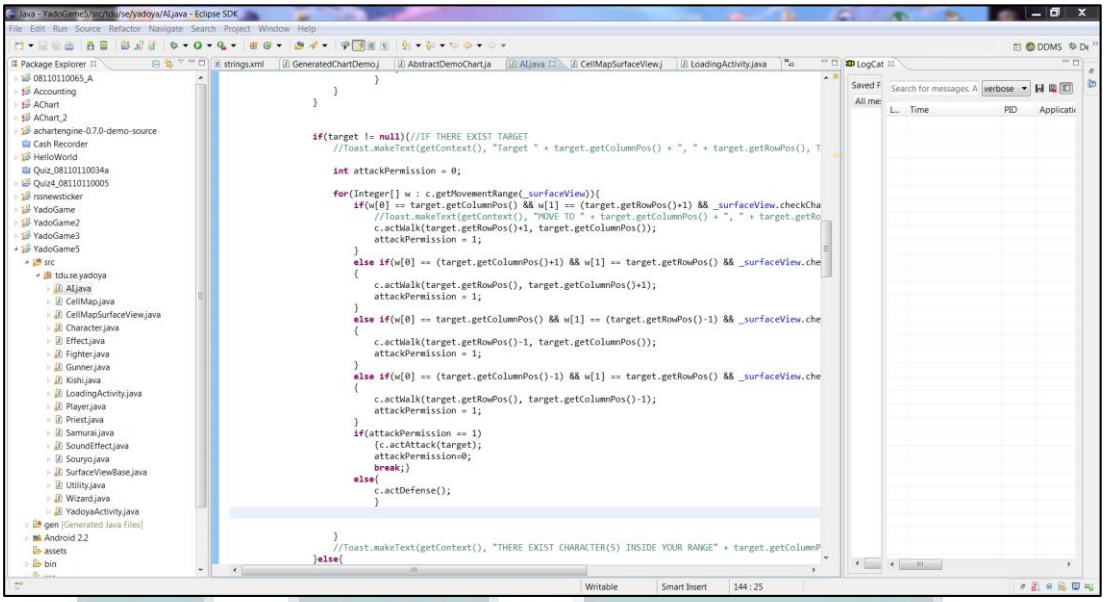
```
void aifunction()
      {
      for each (monster)
           {
           for each(monster attack range)
                {
                if(player exist)
                     {
                     attack player
                     end of this monster turn
                     } END IF
           } END FOR EACH MONSTER ATTACK RANGE
```

```
        check 3 cells around monster
         if(exists something inside 3 cells around monster)
                    {
                    if(something is player)
                          {
                                monster walk to player
                                monster attack payer
                                }END IF
                          }END IF
                    else monster do defence;
                    end of this monster turn
          } END FOR EACH MONSTER
    } END OF FUNCTION
```

Hasil *coding* (*script* seluruhnya) dapat dilihat pada lampiran yang disertakan dengan laporan.

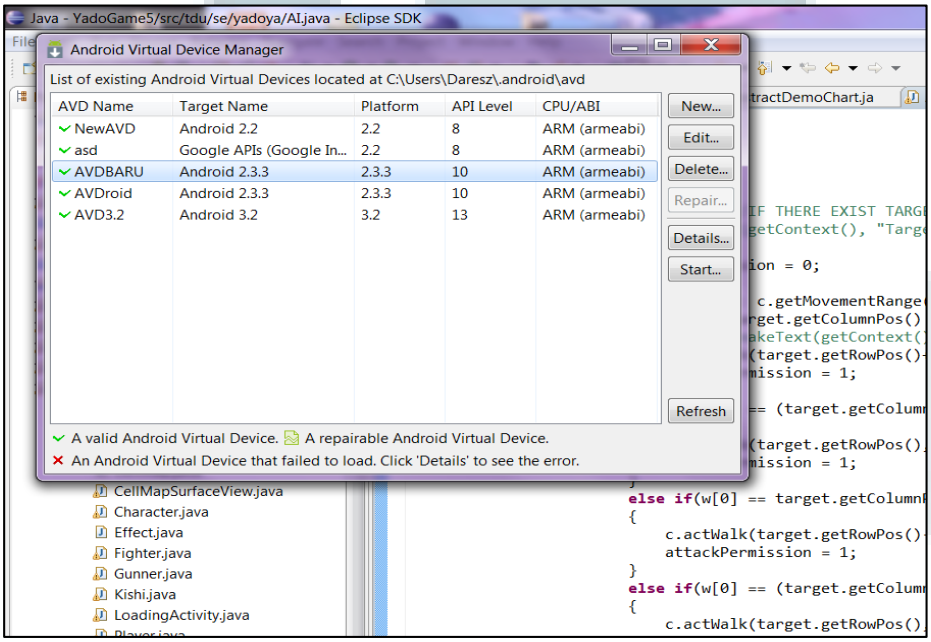
3.3.4 Skenario Pengujian

Berdasarkan sifat cerdas yang telah dikembangkan, dilakukan percobaan untuk memastikan bahwa sifat cerdas telah berjalan dengan semestinya dalam permainan. Percobaan dilakukan menggunakan *Android Virtual Device* (AVD) yang disediakan dalam *development environment* yang digunakan (*Eclipse*). Tampilan aplikasi dapat dilihat pada Gambar 3.8.



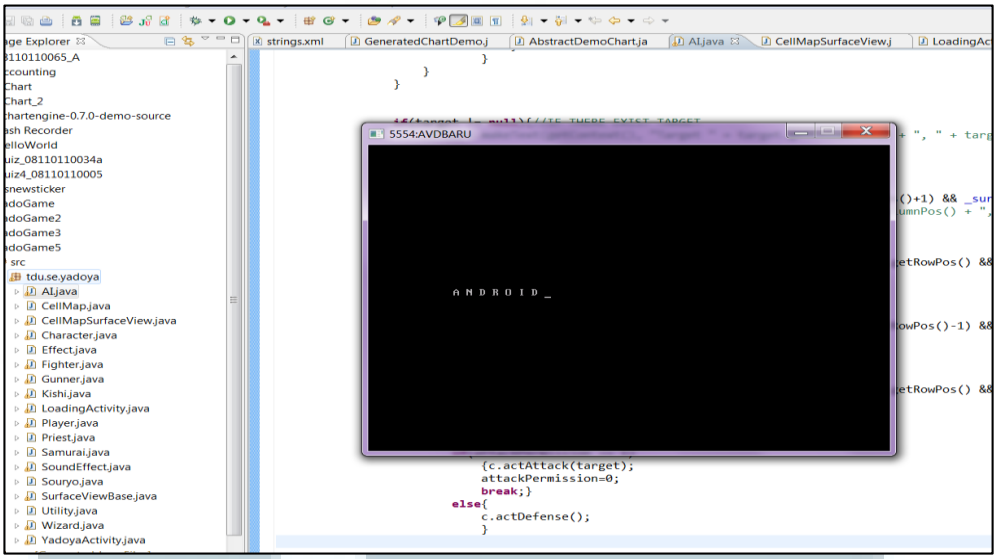
Gambar 3.8 Tampilan *Eclipse Software Development Kit*

Android Virtual Devices (AVD) harus mengikuti *game requirements*, dimana versi *Android* minimal adalah *Android* 2.2. Oleh karena itu dibuatlah AVD dengan versi 2.3.3 untuk mencoba sifat cerdas.

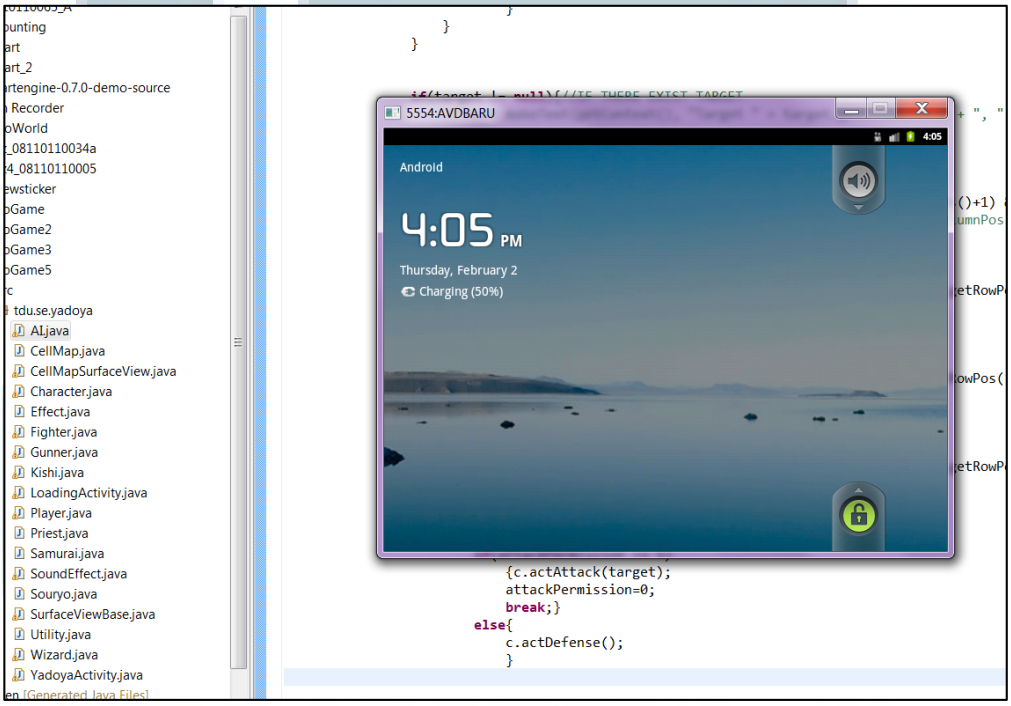


Gambar 3.9 Tampilan AVD Manager

Setelah dibuat, maka dijalankan AVD yang akan dipakai, kemudian akan dilakukan instalasi terhadap permainan yang telah dikembangkan.

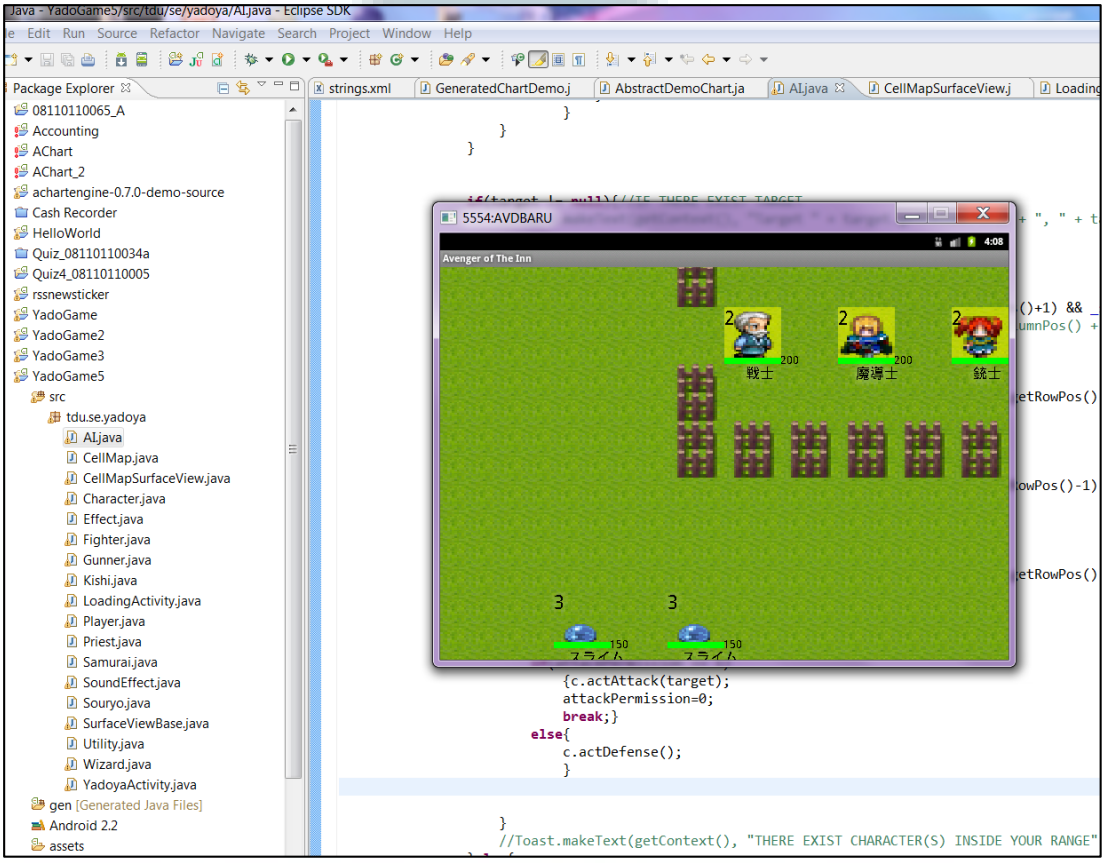


Gambar 3.10 Tampilan AVD Yang Dijalankan



Gambar 3.11 AVD Siap Dipakai

Setelah instalasi terhadap permainan selesai dilakukan, maka AVD akan menjalankan permainan secara otomatis. Tampilan AVD menjalan permainan dapat dilihat pada Gambar 3.12.

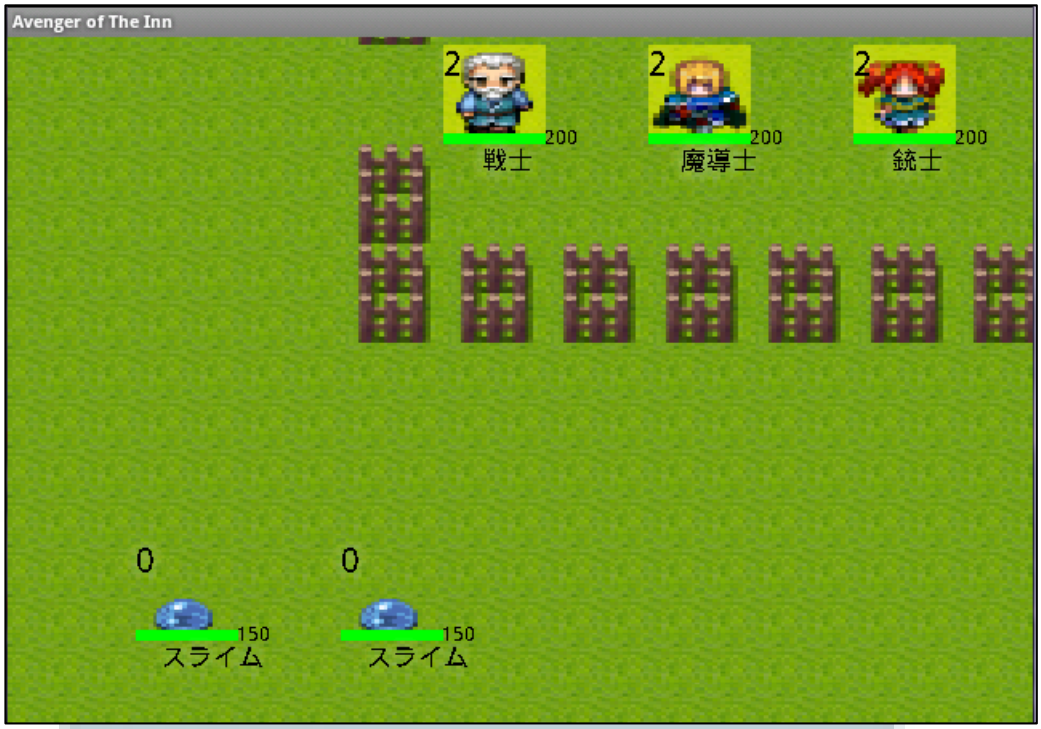


Gambar 3.12 AVD Menjalankan Permainan

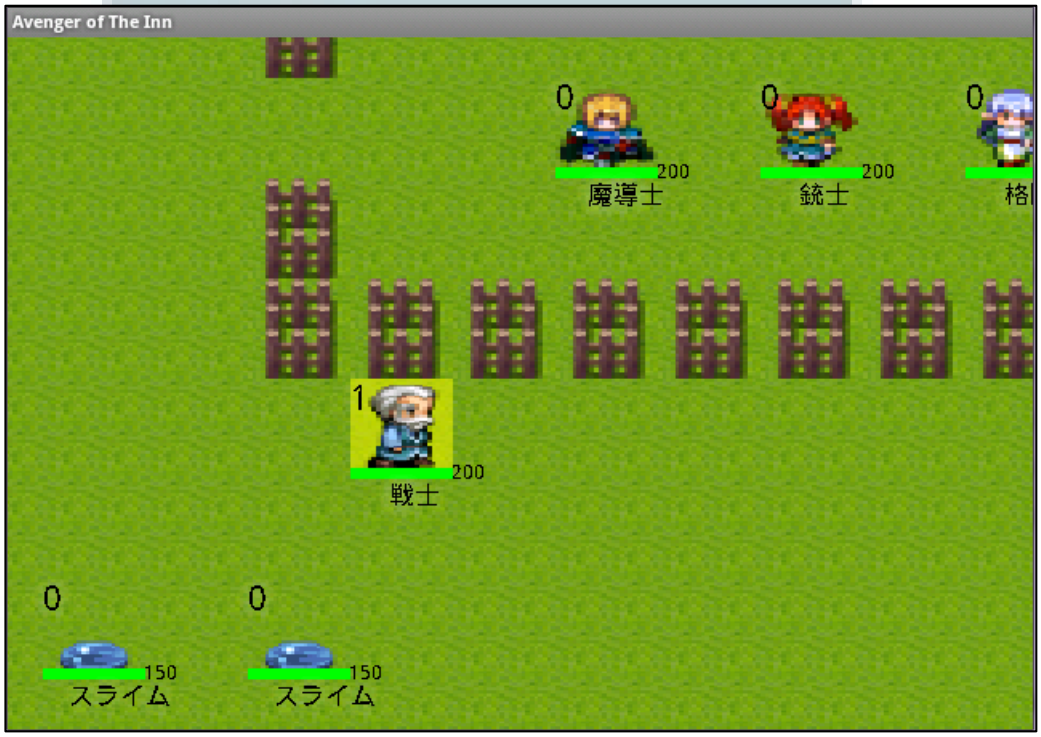
Percobaan dimulai dengan cara mendekatkan pemain kepada *monster* dan memperhatikan respon dari *monster* tersebut.

- *Monster Slime Testing*

Potongan program *monster slime* diimplementasikan ke dalam permainan, sehingga pola gerak dan pola serang *monster* akan sesuai dengan pola *monster slime*. Kondisi awal dan posisi pemain dapat dilihat pada Gambar 3.13 dan Gambar 3.14.



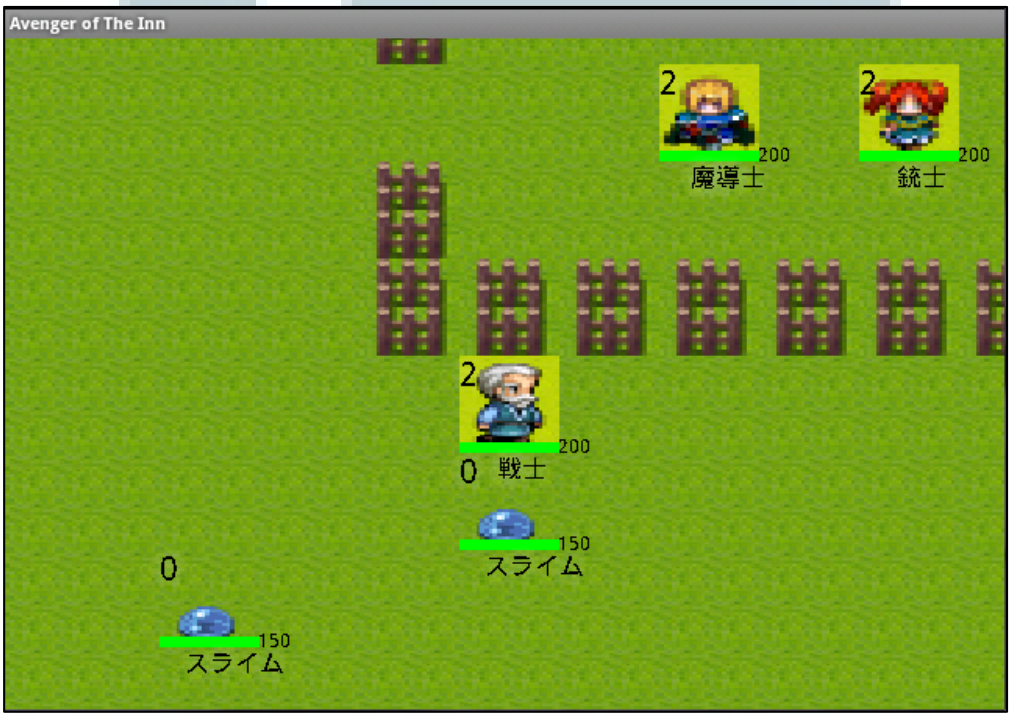
Gambar 3.13 Kondisi Awal Percobaan *Monster Slime*



Gambar 3.14 Posisi Pemain Dalam Percobaan *Monster Slime*

Pada Gambar 3.13 dapat dilihat posisi awal pemain dan *monster*. Dikarenakan dalam uji coba ini hanya satu pemain saja yang akan didekatkan kepada *monster slime*, maka pemain lain akan diberikan aksi bertahan. Pada Gambar 3.14 dapat dilihat satu pemain telah berjalan mendekati *monster slime*. Setelah memastikan pemain telah memasuki jarak respon dari *slime*, pemain diberikan aksi bertahan.

Seperti yang diinginkan, *monster slime* yang dapat melakukan aksi serang, berjalan ke posisi dimana kombinasi jarak serang dan jarak gerak ditemukan. *Slime* berjalan ke samping pemain dan melakukan aksi serang. *Slime* lainnya tidak mendapatkan kombinasi jarak serang dan jarak gerak, sehingga hanya melakukan aksi bertahan. Hasil akhir percobaan dapat dilihat pada Gambar 3.15.



Gambar 3.15 Hasil Respon Percobaan *Monster Slime*

Setelah semua *monster* telah menyelesaikan langkahnya, giliran berpindah ke pemain. Sifat cerdas berjalan sesuai dengan yang diharapkan.

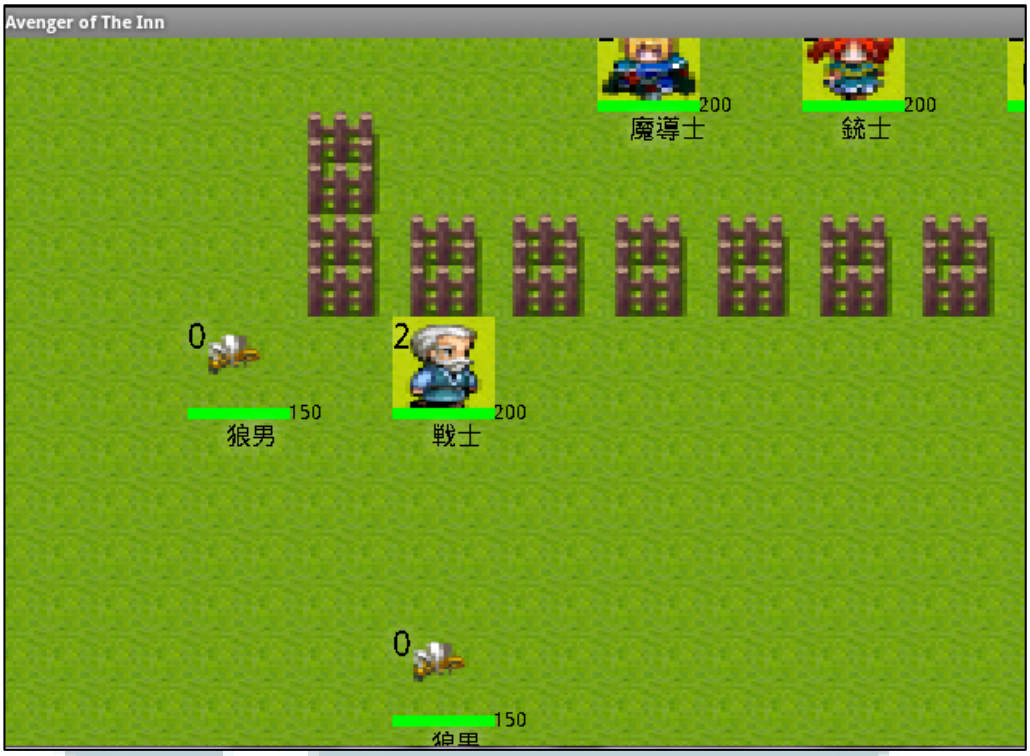
- *Monster Bee Testing*

Potongan program *monster bee* diimplementasikan ke dalam permainan, sehingga pola gerak dan pola serang *monster* akan sesuai dengan pola *monster bee*. Kondisi awal dan posisi pemain dapat dilihat pada Gambar 3.16 dan Gambar 3.17.

Gambar 3.16 Kondisi Awal Percobaan *Monster Bee*

Gambar 3.17 Posisi Pemain Dalam Percobaan *Monster Bee*

Tahap pengujian *monster slime* diterapkan juga dalam pengujian *monster bee*. Kedua *monster bee* tersebut seharusnya dapat menemukan kombinasi jarak gerak dan jarak serang yang mempunyai pemain di dalamnya, sehingga kedua *monster* tersebut akan melakukan aksi serang terhadap pemain. Hasil akhir percobaan dapat dilihat pada Gambar 3.18.



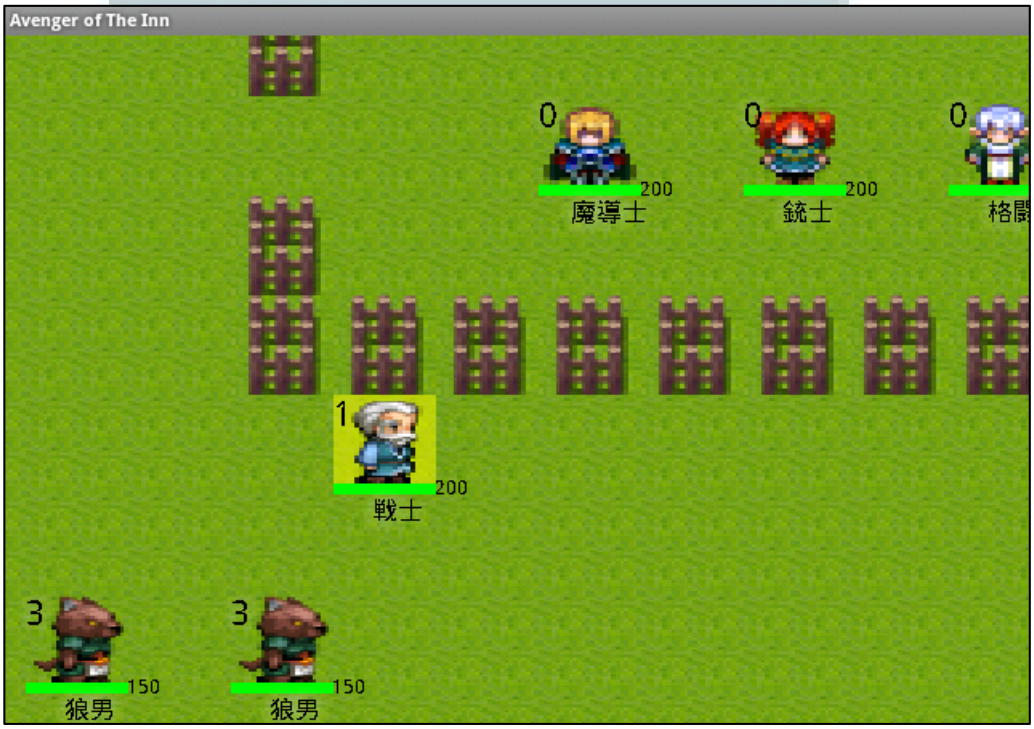
Gambar 3.18 Hasil Respon Percobaan *Monster Bee*

Dapat dilihat bahwa *monster bee* memposisikan diri dan menyerang sesuai dengan rancangan awal, sehingga akan didapat hasil akhir seperti pada Gambar 3.18. Setelah semua *monster* telah menyelesaikan langkahnya, giliran berpindah ke pemain. Dari hasil tersebut ditarik kesimpulan bahwa sifat cerdas *monster bee* berjalan sesuai dengan yang diharapkan.

- *Monster Wolfman Testing*

Potongan program *monster wolfman* diimplementasikan ke dalam permainan, sehingga pola gerak dan pola serang *monster* akan sesuai dengan pola *monster wolfman*. Kondisi awal dan posisi pemain dapat dilihat pada Gambar 3.19 dan Gambar 3.20.

Gambar 3.19 Kondisi Awal Percobaan *Monster Wolfman*



Gambar 3.20 Posisi Pemain Dalam Percobaan *Monster Wolfman*

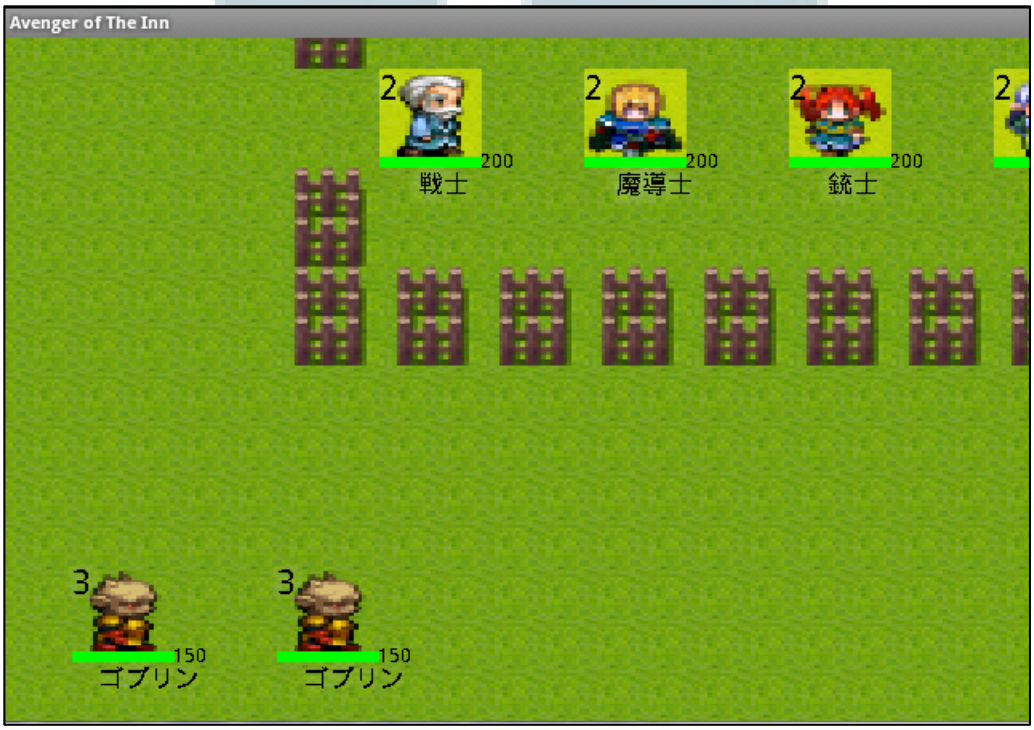
Tahap pengujian *monster slime* diterapkan juga dalam pengujian *monster wolfman*. Kedua *monster wolfman* tersebut seharusnya dapat menemukan kombinasi jarak gerak dan jarak serang yang mempunyai pemain di dalamnya, sehingga kedua *monster* tersebut akan melakukan aksi serang terhadap pemain. Hasil akhir percobaan dapat dilihat pada Gambar 3.21.

Gambar 3.21 Hasil Respon Percobaan *Monster Wolfman*

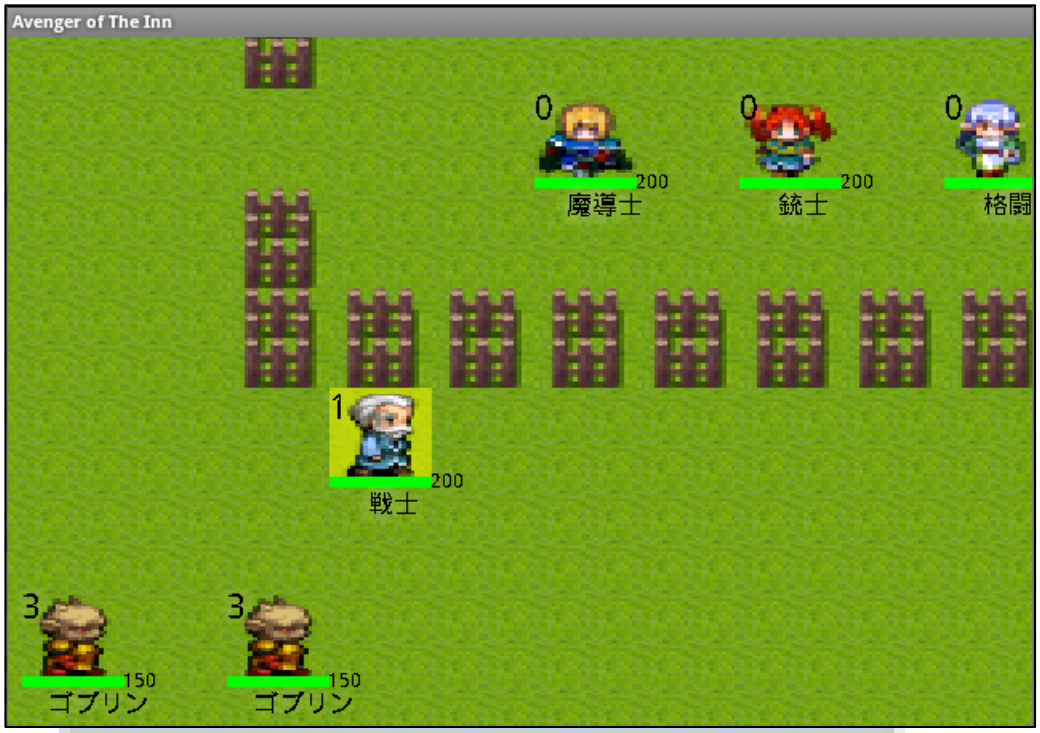
Dapat dilihat bahwa *monster wolfman* memposisikan diri dan menyerang sesuai dengan rancangan awal, sehingga akan didapat hasil akhir seperti pada Gambar 3.21. Setelah semua *monster* telah menyelesaikan langkahnya, giliran berpindah ke pemain. Dari hasil tersebut ditarik kesimpulan bahwa sifat cerdas *monster bee* berjalan sesuai dengan yang diharapkan.

- *Monster Goblin Testing*

Potongan program *monster goblin* diimplementasikan ke dalam permainan, sehingga pola gerak dan pola serang *monster* akan sesuai dengan pola *monster goblin*. Kondisi awal dan posisi pemain dapat dilihat pada Gambar 3.22 dan Gambar 3.23.



Gambar 3.22 Kondisi Awal Percobaan *Monster Goblin*



Gambar 3.23 Posisi Pemain Dalam Percobaan *Monster Goblin*

Tahap pengujian *monster slime* diterapkan juga dalam pengujian *monster goblin*. Kedua *monster goblin* tersebut seharusnya dapat menemukan kombinasi jarak gerak dan jarak serang yang mempunyai pemain di dalamnya, sehingga kedua *monster* tersebut akan melakukan aksi serang terhadap pemain. Hasil akhir percobaan dapat dilihat pada Gambar 3.24.

Gambar 3.24 Hasil Respon Percobaan *Monster Goblin*

Dapat dilihat bahwa *monster goblin* memposisikan diri dan menyerang sesuai dengan rancangan awal, sehingga akan didapat hasil akhir seperti pada Gambar 3.24. Dari hasil tersebut ditarik kesimpulan bahwa sifat cerdas *monster bee* berjalan sesuai dengan yang diharapkan. Percobaan memberikan hasil bahwa setiap jenis *monster* memberikan respon sesuai dengan jarak gerak dan jarak serang masing-masing sesuai dengan yang ditetapkan dalam aturan permainan.