Laporan Tugas Besar I IF2211 Strategi Algoritma

Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan “Worms”

Kelompok Konvermex_135

(Nama - NIM Anggota)

Girvin Junod - 13519096

David Owen Adiwiguna - 13519169

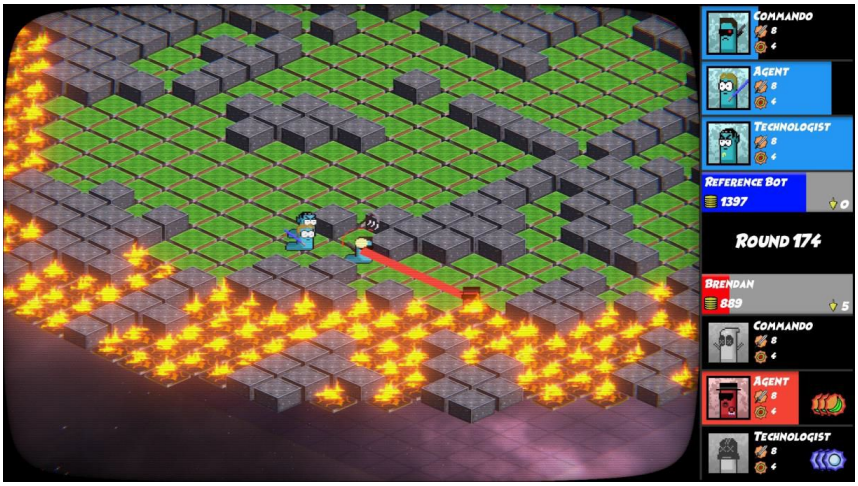
Leonard Mattheus - 13519215

Institut Teknologi Bandung

2021

Bab 1. Deskripsi Tugas

Worms adalah sebuah *turned-based game* dari Entelect Challenge 2019 yang didesain agar pemain dapat beradu bot ciptaan mereka. Setiap pemain akan memiliki 3 *worms* dengan perannya masing-masing. Pemain dinyatakan menang jika ia berhasil mengeliminasi pasukan *worms* lawan menggunakan bot *worms*-nya yang diprogram dengan strategi dan algoritma buatannya.



Gambar 1.1. Contoh Tampilan Game

Tugasnya adalah mengimplementasikan seorang "pemain" *Worms* berupa *bot*, dengan menggunakan strategi *greedy* untuk memenangkan permainan. Untuk mengimplementasikan seorang "pemain" tersebut, disarankan untuk melanjutkan program yang terdapat pada *starter bot* di dalam *starter pack* pada laman berikut ini:
(https://github.com/EntelectChallenge/2019-Worms/releases/tag/2019.3.2)

Hasil akhir yang diharapkan adalah terbentuknya sebuah *bot* untuk permainan *Worms* ini yang menggunakan strategi algoritma *greedy* dalam pemrogramannya untuk memenangkan permainan.

Bab 2. Landasan Teori

Algoritma *greedy* merupakan metode yang populer dan sederhana untuk memecahkan persoalan optimasi. Algoritma ini memecahkan persoalan secara langkah per langkah (*step by step*) sedemikian sehingga, pada setiap langkah dilakukan pengambilan pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensinya ke depan (prinsip "take what you can get now!") dan "berharap" bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global. Optimum lokal ini dapat ditentukan dari banyak faktor tergantung persoalan, seperti contohnya selalu mengambil pilihan dengan nilai variabel tertentu yang terbesar.

Elemen-elemen algoritma *greedy*:

1. Himpunan kandidat, C : berisi kandidat yang akan dipilih pada setiap Langkah (misal: simpul/sisi di dalam graf, job, task, koin, benda, karakter, dsb)
2. Himpunan solusi, S : berisi kandidat yang sudah dipilih
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
5. Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi objektif : memaksimumkan atau meminimumkan

Skema Umum Algoritma Greedy adalah sebagai berikut:

```
function greedy(C : himpunan_kandidat) → himpunan_solusi
 { Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
 x : kandidat
 S : himpunan_solusi
Algoritma:
 S ← {} { inisialisasi S dengan kosong }
 while (not SOLUSI(S)) and (C != {} ) do
   x ← SELEKSI(C) { pilih sebuah kandidat dari C}
   C ← C - {x} { buang x dari C karena sudah dipilih }
```

```
  if LAYAK(S ∪ {x}) then { x memenuhi kelayakan untuk dimasukkan ke dalam  himpunan solusi }
    S ← S ∪ {x} { masukkan x ke dalam himpunan solusi }
  endif
endwhile
{SOLUSI(S) or C = {} }

if SOLUSI(S) then { solusi sudah lengkap }
  return S
else
 write('tidak ada solusi')
endif
```

Pengaturan Game Engine

Pada *game engine Worms*, terdapat dua player yang akan bertarung masing-masing memiliki 3 worm. Penambahan bot dilakukan dengan menyesuaikan bot yang sudah di build melalui maven project yang akan dibangun dependenciesnya melalui pow.xml dan bot.json. Jadi, setiap perintah yang kita rancang di kelas java akan dieksekusi sesuai dengan prinsip pemrograman berorientasi objek (OOP). Cara membangun game engine adalah sebagai berikut:

1. Masuk ke dalam folder game-runner
2. Masukkan game-engine yang ada pada starter pack
3. Sesuaikan game-config-runner.json dan masukkan folder yang ditunjuk sebagai player A maupun player B
4. Klik build yang ada pada folder untuk menginstall game engine

Menambahkan Strategi Greedy pada Bot

Menyematkan strategi greedy ke dalam game engine bisa dilakukan dengan cara membuat folder bot dengan struktur sebagai berikut:

Bot

→ java

→ src/main

→ Bot.java

→ Main.java

→ target

→ Bot.class

→ Main.class

→ bot.json

→ game dependencies.jar

Algoritma *greedy* akan dimasukkan ke dalam bot.java pada bagian run(). Terdapat beberapa *command* yang dapat dimasukkan ke bot untuk menentukan langkah yang akan dilakukan oleh cacing yang bersangkutan. *Command* yang ada yaitu:

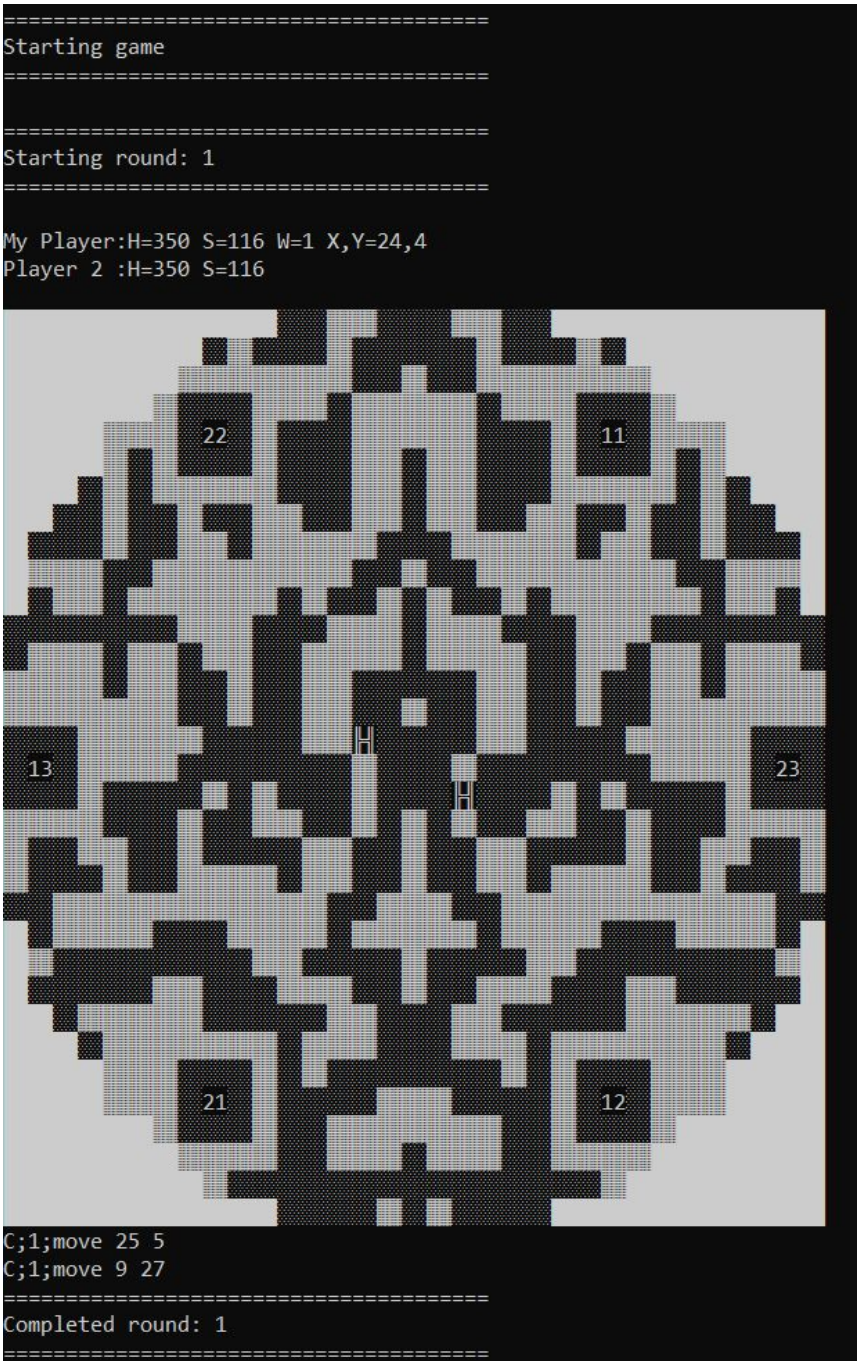
1. *Select*: berfungsi untuk menunjuk worm supaya bisa melakukan aksinya secara terus menerus. Hanya memiliki 5 *turn* saja
2. *Shoot*: berguna untuk menembak laser yang berbentuk garis diikuti dengan direction dalam bentuk huruf N,S,E,W,dsb
3. *Banana*: berguna untuk melempar *banana bomb* yang memiliki damage area kepada sekumpulan cacing di area tersebut dengan damage tertinggi di cell yang dilempari *banana bomb*.
4. *Snowball*: berguna untuk melempar *snowball* ke lawan yang memiliki area of effect dimana cacing di area tersebut akan beku dan tidak bisa melakukan apapun untuk 5 ronde.
5. *Move*: berfungsi untuk membuat *worm* bergerak ke suatu koordinat *cell*.
6. *Dig*: berguna untuk mengubah *cell dirt* menjadi *cell air*.

Untuk menyusun identitas bot, kita harus mengaturnya sesuai dengan pengaturan yang tertera pada bot.json. Pengaturan pada bot.json adalah sebagai berikut:

```
{
    "author": <nama pembuat bot>,
    "email":<email pembuat bot>,
    "nickName": <Tampilan nama pada visualizer>,
    "botLocation": <lokasi bot di build pada kelas tertentu>,
    "botFileName": <Format jar untuk menjalankan bot>,
    "botLanguage": <bahasa pemograman yang dipilih>
}
```

Menjalankan Game Engine

Kita dapat menjalankan run, sehingga muncul cmd yang menampilkan state setiap round yang ada. Hasil dari menjalankan run adalah file matches-logs yang berisi history setiap round beserta health yang tersisa bagi *worms* Pemain A dan B.



Gambar 2.1 Contoh Tampilan Command Line Interface

Bab 3. Pemanfaatan Strategi *Greedy*

Asumsi yang mendasari penyusunan strategi permainan ini adalah

1. Setiap *Worms* akan bergiliran melakukan aksi dimulai dari ID pertama, kedua, ketiga, kembali ke pertama, dan seterusnya.
2. *Worms* berjumlah 3 buah di setiap tim nya
3. Poin didapatkan dari menembak tepat sasaran, ataupun rajin menggali dirt.
4. Tim yang berhasil membunuh semua *worms* lawan akan menang
5. Apabila kedua tim masih tidak ada yang mati, akan dilanjutkan hingga ronde maksimum 400. Tim yang menang akan ditentukan berdasarkan perolehan poin tertinggi.

Proses *mapping* persoalan *Worms* menjadi elemen-elemen algoritma Greedy (himpunan kandidat, himpunan solusi, fungsi seleksi, fungsi kelayakan, fungsi objektif)
Adapun uraian dari mapping persoalan *Worms* adalah:

Himpunan kandidat :
{*banana bomb*, *snowball*, *attack*, *move*, *dig*, *do nothing, select*}

Himpunan solusi :
Command yang valid untuk setiap rondenya dan *worms* mendapatkan perintah yang sesuai dengan aturan main dan untuk memenangkan permainan.

Fungsi seleksi :
Pilihlah aksi yang memberikan *damage* tertinggi ke musuh atau damage terendah ke diri sendiri sehingga dapat mengeliminasi *worms* musuh dan menjaga *worms* pemain untuk tetap bertahan.

Fungsi kelayakan :
Command valid seperti persediaan a*mmo banana bomb* dan *snowball* masih ada, arah gerak *worms* valid, *select* masih tersedia, ada *wall* yang bisa di *dig*.

Fungsi objektif :
Memenangkan permainan dengan menjadi pemain terakhir dengan *worms* yang masih hidup.

Eksplorasi alternatif solusi *greedy*

Pertama, kelompok kami merancang strategi *greedy by points*. Dalam teknis permainan, poin akan digunakan untuk menentukan pemenang ketika permainan mencapai ronde 400. Untuk itu, strategi greedy ini memiliki fokus agar *worms* dapat mendapatkan poin terbanyak dan selamat sampai ronde 400. Dari permainan, tindakan dengan poin terbesar adalah menyerang *worms* musuh dan mengalahkan *worms* musuh. Namun, jika *worms* terkena serangan, maka akan kehilangan poin. Oleh karena itu, lebih baik jika *worms* tidak menyerang dan tidak diserang musuh namun hanya bergerak-gerak menghindari lawan. *Worms* juga melakukan tindakan lain yang memberikan poin seperti *dig* dan freeze. Pada akhirnya kami memutuskan untuk tidak menggunakan strategi ini karena kami menganggap strategi ini tidak memungkinkan karena 400 ronde itu lama dan peta permainan akan semakin mengecil akibat *cell lava* sehingga tidak mungkin untuk *worms* untuk terus menghindar dari pertempuran sampai ronde 400, oleh karena itu kami menilai bahwa strategi ini tidak efektif. Efisiensi dari algoritma ini kami perkirakan biasa saja dengan kompleksitas O(n).

Kedua, kelompok kami merancang strategi *greedy by position*. Karena adanya zona api di permainan, maka bagian tengah dari peta permainan adalah daerah paling aman dari zona api ini. Bagian tengah ini juga memberikan kelebihan lain yaitu *worms* tim kami jadi lebih mungkin untuk melawan *worms* musuh dalam kondisi banyak lawan satu sehingga lebih mungkin untuk mengalahkan *worms* musuh, namun strategi ini memiliki kelemahan yaitu rentan akan terkena *splash damage* dari *banana bomb* atau *snowball* sekaligus dan juga rentan terjadi *friendly fire* sehingga kurang efektif untuk memenangkan permainan. Strategi ini efisiensinya dapat dibilang biasa saja, dengan perkiraan kompleksitas O(n).

Ketiga, kami merancang strategi *greedy by distance*. Pada awal permainan, setiap *worms* akan mulai pada tempat yang berbeda dari satu sama lain. Strategi *greedy* ini adalah tiap *worms* akan bergerak acak sampai dia bertemu *worms* musuh lalu dia akan bergerak menuju dan menyerang *worms* musuh tersebut. Prioritasnya adalah *worms* musuh yang terdekat dari *worms* itu. Strategi ini cukup efektif dalam membuat *worms* menemukan dan melawan musuh. Namun, karena algoritma *greedy*-nya selalu mengincar *worms* terdekat, kami berpikir ini akan membuat *worms*-nya jadi melawan *worms* musuh dalam pertempuran 1 lawan 1 yang kadang dapat jadi

tidak menguntungkan karena perbedaan profession dari *worms* yang bertempur sehingga hasilnya tidak konsisten. Tidak hanya itu, kami berpikir bisa jadi worms kami malah masuk ke dalam pertempuran satu melawan banyak yang tidak menguntungkan. Oleh karena itu, strategi ini kami nilai tidak begitu efektif karena ketidakkonsistenan ini. Efisiensi strategi ini kami perkirakan biasa saja dengan kompleksitas O(n).

Keempat, kami merancang strategi *greedy by damage*. Strategi ini berfokus pada memberikan damage kepada *worms* musuh sebesar mungkin dan meminimalisir damage yang diterima. Dalam strategi ini, *worms* menggunakan banana bomb dan *snowball* selagi bisa, karena kedua senjata tersebut memiliki potensi damage paling besar dengan catatan *snowball* hanya membekukan musuh dan memberi *worms* kesempatan menyerang. Karena strategi ini kamu juga memerhatikan *damage* yang diterima walaupun tidak sepenting *damage* yang diberikan, kami merancang juga ada pengecualian di mana aksi yang memberi *damage* ini justru akan lebih menyakiti tim sendiri karena hal seperti *friendly fire*. Contohnya, *snowball* ataupun *banana bomb* tidak akan dipakai jika *worms* itu berada di dalam area *damage* dari *snowball* atau *banana bomb*. Pengecualian ini bisa dihiraukan jika *HP* dari *worms* tersisa sedikit sehingga opsi paling efektif untuk melakukan *damage* ke musuh adalah untuk melempar *banana bom* atau *snowball* ini ke musuh sehingga mengenai *worms* itu sendiri. Contohnya, kalau *HP worms* tinggal 50, jika ia menggunakan tiga *banana bomb*, dia akan memberi *damage* sebanyak 60 ke musuh dan 50 ke diri sendiri yang jika kalau *banana bomb* tidak dikeluarkan, kemungkinan besar *damage* yang diberi dan diterima akan sama atau berbeda 8 akibat perbedaan 1 *turn* sehingga cocok dengan strategi greedy kami. Ketika tidak bertemu dengan musuh, *worms* kami dibuat untuk bergerak di area sekitarnya. Ini agar *worms* musuh yang datang mendekati *worms* kami, bukan sebaliknya. Selain itu, kami membuat *worms* kami akan mulai mendekati tengah dari peta permainan ketika permainan sudah berlangsung sampai sekitar 70 ronde untuk menghindari *cell lava* yang muncul dari pinggir peta. *Worms* kami juga akan bergerak ketika ada *worms* kami yang telah diserang sehingga *worms* lain akan datang untuk membantu dia agar *damage* yang diberi ke musuh lebih banyak dan mengurangi *damage* yang diterima oleh *worms* yang diserang. *Worms* kami bergeraknya dari sekitar ronde 70 karena kami ingin membuat *worms* musuh untuk datang mencari *worms* kami sehingga kemungkinan *banana bomb* dan *snowball* musuh akan terpakai untuk satu *worms* saja sehingga mengurangi potensi *damage* total musuh. Dengan membuat

worms musuh yang datang mencari kami, kami juga membuat *worms* kami tidak perlu membuang *turn* untuk *move* ke musuh namun bisa untuk langsung menyerang. Dengan *greedy by damage*, juga bisa dibuat *worms* kami untuk menghindari *friendly fire* terutama *friendly fire* dari *attack*. Strategi ini cukup efektif, namun damage tidak maksimal karena dari desainnya *worms* tidak bisa menunggu dan menargetkan *banana bomb* atau *snowball* agar terkena beberapa *worms* sekaligus namun hanya langsung melemparkan banana bomb atau *snowball* ke *worms* musuh pertama yang ada dalam jangkauannya. Namun ini normal karena dipakai strategi algoritma *greedy* sehingga wajar jika tidak ada pertimbangan untuk masa depan di algoritmanya. Strategi ini juga efektif dalam memenangkan permainan karena dirancang untuk memberi *damage* sebanyak mungkin ke musuh dan meminimalisir *damage* yang diterima oleh *worms*. Efisiensi strategi ini kami perkirakan biasa saja, dengan perkiraan kompleksitas O(n).

Strategi yang akhirnya kami pilih adalah strategi *greedy by damage* karena menurut kami strategi ini dapat memenangkan permainan dengan cukup konsisten karena algoritmanya yang berfokus pada melakukan *damage* sebanyak mungkin dan meminimalisir *damage* yang diterima, selain itu dalam strategi ini kami juga menggunakan *banana bomb* dan *snowball* sebisa mungkin karena senjata tersebut melakukan *damage* yang paling besar sehingga strategi ini kami nilai cukup konsisten dalam memenangkan permainan dan memiliki kelemahan yang minim, tidak seperti alternatif strategi lainnya. Kelemahan strategi ini yang kami temukan adalah kemungkinan *damage* yang diberikan ke musuh tidak benar-benar maksimal karena memakai strategi *greedy*, namun kami anggap ini normal dari pemakaian strategi *greedy* sehingga tidak terlalu dipermasalahkan. Kami tidak memilih strategi *greedy by points* karena kami pikir tidak memungkinkan untuk *worms* bertahan hidup selama 400 ronde karena map permainan yang terus mengecil akibat *lava cell* yang muncul sejak ronde 100 dan *worms* musuh yang terus mencoba membunuh *worms* kami. Kami juga tidak memilih strategi *greedy by position* karena strategi tersebut rawan terkena *splash damage* dari *banana bomb* dan *snowball* yang dilempar oleh *worms* lawan, rawan terjadinya *friendly fire,* dan kemungkinan strateginya gagal dieksekusi karena sudah dicegat oleh worms musuh dari awal *worms* kami bergerak. Sementara untuk strategi *greedy by distance*, kami menilai strategi ini tidak konsisten sehingga tidak bisa diandalkan dan sangat bergantung akan profesi dari *worms* lawan dan pergerakan *worms* lawan.

Bab 4. Implementasi dan Pengujian

Implementasi program dalam *game engine*

Berikut adalah pseudocode dari bagian-bagian implementasi program dalam game engine. Line yang dimulai dengan "//" adalah komentar yang bukan bagian dari pseudocode, hanya untuk memperjelas.

```
//Move dari panggilan bantuan
if  (ada panggilan bantuan) then
            if (HP bot enemy <=20) then
                Batal panggil bantuan
          //Bantuan adalah memanggil worms lain untuk datang ke cell worms yang bertempur
          //Cek Apakah bisa select (diprioritaskan snowball trus select)
             Else then
                  Worms pergi ke arah panggilan bantuan
//Ambil health pack
 if (ada health pack di sekitar) then
     Worms bergerak ke lokasi health pack

//Penyerangan dengan banana Bomb melewati wall
        mencari worms untuk target banana bomb
        if (ada target yang bisa dilempar banana bomb)
            if (masih ada ammo banana bomb dan jarak musuh tidak terlalu dekat)
                Panggil bantuan
           Serang musuh dengan banana bomb melewati wall

//Penyerangan
        if (ada enemy worms dalam jarak serang)
            //batalkan bantuannya kalau musuh udah sekarat
            if (HP bot musuh <= 20)
                Batalkan panggilan bantuan
            if (ada musuh yang bisa dilempar banana bomb dan masih ada bananabomb) then
                Panggil bantuan
           if (jarak musuh tidak dekat) then
                     Lempar musuh dengan banana bomb
                else then
                     If (HP worms kurang dari 60) then
                         Lempar musuh dengan banana bomb
            else if (ada lawan yang bisa dilempar snowball dan masih ada snowball) then
                Panggil bantuan
                If (jarak musuh tidak dekat) then
                 Lempar musuh dengan snowball lalu select worms yang baru melempar
           else then
                 If (HP worms kurang dari 60) then
```

```
                    Lempar musuh dengan snowball
            else then
                Shoot ke arah direction musuh

//Move
//kalau tidak ada musuh dan tidak ada panggilan bantuan, dia bergerak (Perpindahan)
        if (jika sudah round 70 atau lebih dan worms tidak di tengah peta permainan) then
                Bergerak ke bagian tengah peta permainan
                if (cell tujuan adalah dirt) then
                    dig cell tersebut
                else if (cell air)
                    gerak ke cell tersebut
                else then
                    do nothing
        else then
                Bergerak random di sekitar area
                if (cell tujuan adalah dirt) then
                    dig cell tersebut
                else if (cell air)
                    gerak ke cell tersebut
                else then
                    do nothing
```

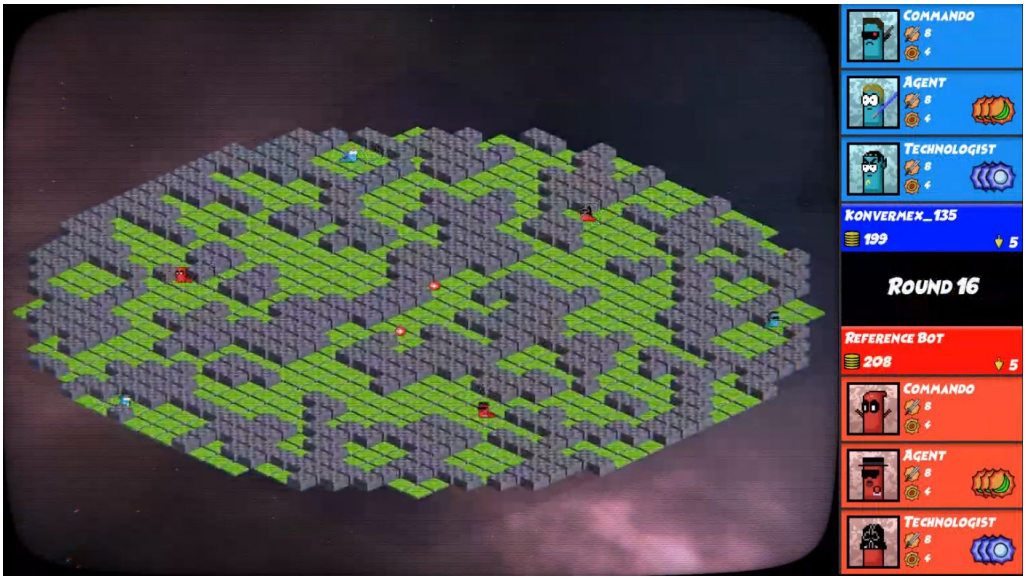
Penjelasan Struktur Data

Ada pun struktur data yang kami gunakan terbagi menjadi 3 bagian besar yaitu command, entities, enum yang disertakan dengan bot.java dan main.java.

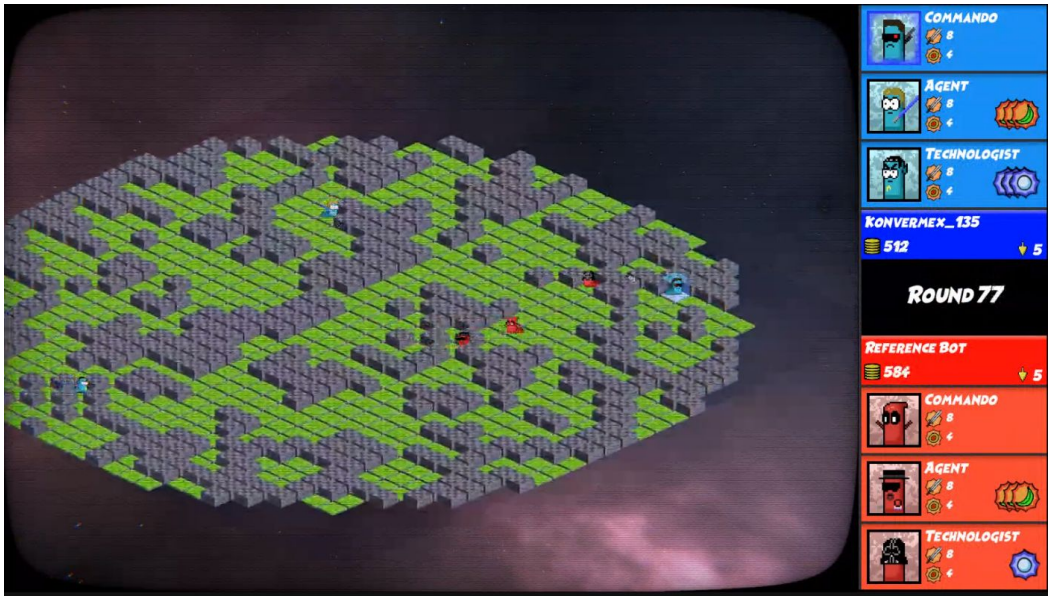
1. Command berfungsi untuk menyimpan kelas-kelas yang berhubungan dengan prosedur game-engine yang dijelaskan pada Bab 2
    a. BananaCommand : command untuk melempar bom banana ke koordinat tertentu
    b. Command: interface yang berguna untuk merender string yang ada
    c. DigCommand: command untuk menggali tembok yang bertipe *DIRT*
    d. DoNothingCommand: bengong aja
    e. MoveCommand: berguna untuk memindahkan worm ke posisi tertentu
    f. SelectCommand: berguna untuk memilih worm yang akan mengeksekusi perintah tertentu
    g. ShootCommand: command untuk menembak musuh dengan laser yang segaris dengan arah mata angin
    h. SnowballCommand: command untuk membekukan musuh

2. Entities berguna sebagai objek yang ada pada gamestate
    a. BananaBombs: objek *banana bomb*
    b. Cell: cell yang ditempati *worms* (baik worm musuh maupun worm sendiri)
    c. GameState: objek yang menyimpan state tertentu mulai dari health, id, damage, dan informasi lainnya
    d. MyPlayer: objek yang berisi score total , id pemain, dan ketiga worm yang dimiliki player A
    e. MyWorm: *inventory* dari *worms* permain yang berisi *weapon*, *banana bombs*, dan *snowballs*
    f. Opponent: objek yang berisi score total , id pemain, dan ketiga worm yang dimiliki player B
    g. Occupier: objek yang berisi informasi cell telah diduduki atau tidak
    h. Position: objek posisi berupa koordinat
    i. PowerUp: objek *health pack*
    j. Snowballs : objek *snowball*
    k. Weapon : senjata *shoot* yang dimiliki *worm*
    l. Worm: objek *worms* yang ada di permainan
3. Enums berguna sebagai alat iterasi yang dibuat untuk mencocokan setiap kemungkinan yang ada
    a. CellType: tipe dari *cell* yaitu *dirt, air, space,,* dan *lava*
    b. Direction: Arahan *worms* untuk bergerak
    c. PowerUpType: ada atau tidaknya health pack di *cell*.
4. Bot.java yang berisi algoritma bot yang dibuat oleh pemain
5. Main.java yang berisi mekanisme pembacaan state dan perintah eksekusi state

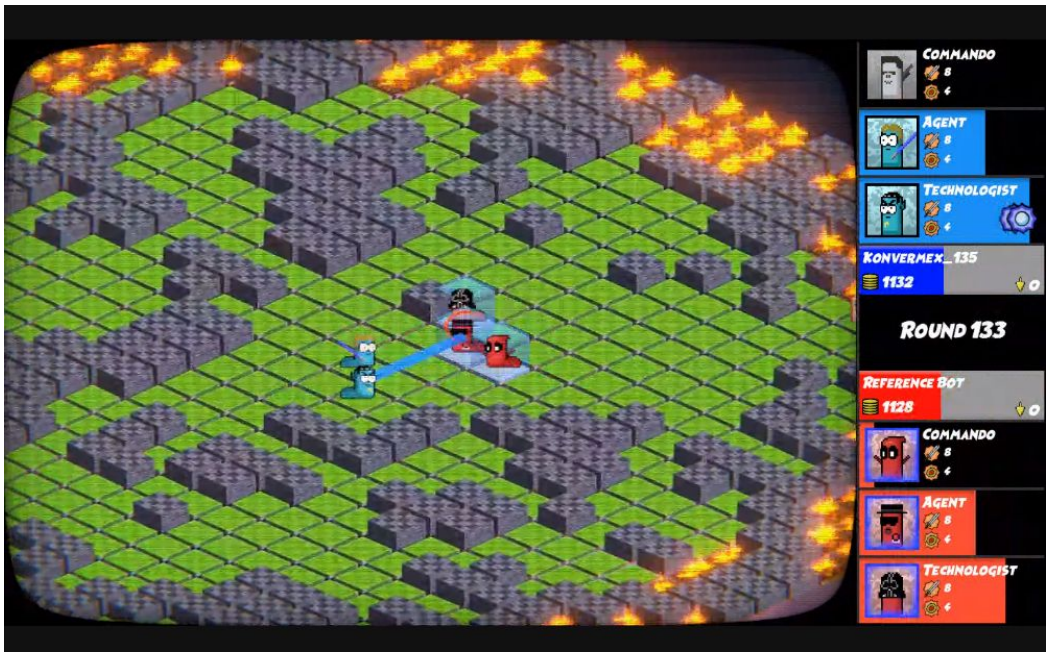
Analisis Desain Solusi yang Diimplementasikan dari Pengujian



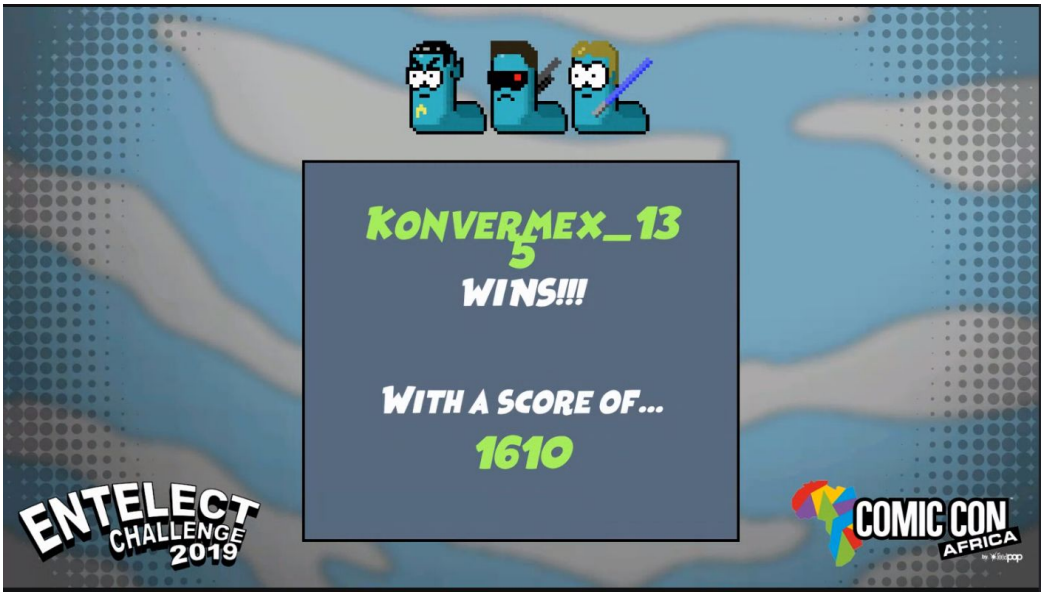
Gambar 4.1 Awal permainan



Gamber 4.2 Pertengahan permainan



Gambar 4.3 Pertempuran worms



Gambar 4.4 Akhir permainan

Di atas adalah beberapa screenshot dari pengujian-pengujian yang kami lakukan. Dari pengujian yang kami lakukan dari program yang telah kami buat, kami menemukan program kami berdasarkan strategi algoritma *greedy by damage* ini berhasil untuk mendapatkan solusi yang optimal terhadap permainan pada kebanyakan waktu. Bot kami mampu mendapatkan *win rate* yang cukup bagus ketika diadu dengan bot-bot lainnya. Bot kami juga mampu bekerja sesuai

dengan rancangan strategi algoritma *greedy by damage* kami. Bot mampu melakukan aksi seperti menyerang musuh dengan *banana bomb* dan *snowball* setiap kali dapat dilakukan, mampu menghindari attack yang *friendly fire*, dan aksi-aksi lainnya yang sesuai dengan rancangan strategi kami. Ada beberapa kondisi di mana strategi ini tidak begitu optimal, seperti saat *worms* kami akan langsung menghabiskan semua *banana bomb* dan *snowbal*l ke *worms* pertama yang ditemuinya padahal mungkin akan muncul kondisi di depannya di mana mereka akan bisa menggunakan *banana bomb* dan *snowball* itu ke banyak musuh sekaligus. Juga, algoritma kami buat masih terlimitasi dalam perhitungan *friendly fire* dari pelemparan *banana bomb* dan *snowball*. Jadi, jika ada *worms* teman yang berada dekat dengan *worms* musuh, maka *worms* akan melempar *banana bomb* atau *snowball* ke *worms* musuh itu tanpa pertimbangan *friendly fire* karena fokus untuk memberikan damage ke musuh. Ini dikarenakan keterbatasan algoritma kami dalam mendeteksi *friendly fire* itu dan perancangan algoritma kami yang mementingkan pemberian *damage* terlebih dahulu, baru meminimalisir *damage* yang diterima. Namun kami menilai ini tidak begitu berdampak pada hasil akhirnya karena dari pengetesan kami, situasi ini jarang terjadi yang sampai merugikan *worms* kami.

Bab 5. Kesimpulan dan Saran

Kelompok kami berhasil membuat sebuah algoritma yang berdasarkan strategi algoritma *greedy* untuk diimplementasikan pada *bot* dalam permainan ini sesuai dengan tujuan dari permainan *worms* ini, yaitu memenangkan permainan dengan cara mengeliminasi musuh. Strategi *greedy* yang kami pakai merupakan strategi *greedy by damage* yang berfokus pada melakukan damage sebanyak mungkin kepada *worms* musuh dan mendapatkan damage sekecil mungkin dari *worms* musuh.

Untuk pengembangan algoritma ini, sistem *targeting* banana bomb dan snowball bisa dikembangkan lagi, agar dapat mendeteksi apakah *banana bomb* atau *snowball* dapat mengenai 2 atau lebih *worms* lawan sekaligus. Selain itu, *select* bisa dimanfaatkan lebih baik lagi agar dapat memilih *worms* yang memiliki potensi untuk melakukan damage terbesar, *movement* dari *worms* juga bisa diatur jika sedang melawan *worms* lawan agar sebisa mungkin membuat *worms* lawan melakukan *friendly fire.* Algoritma untuk mendeteksi *friendly fire* dari *banana bomb* dan *snowball* juga bisa lebih dikembangkan agar lebih sesuai ke strategi *greedy by damage*.

Daftar Pustaka

1. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf
2. https://github.com/EntelectChallenge/2019-Worms