LAPORAN TUGAS BESAR 1

IF2211 STRATEGI ALGORITMA

Kelompok Tembak2

Disusun Oleh:

13523124 - Muhammad Raihaan Perdana

13523140 - Mahesa Fadhillah Andre

13523155 - M Abizzar Gamadrian

PROGRAM STUDI TEKNIK INFORMATIKA

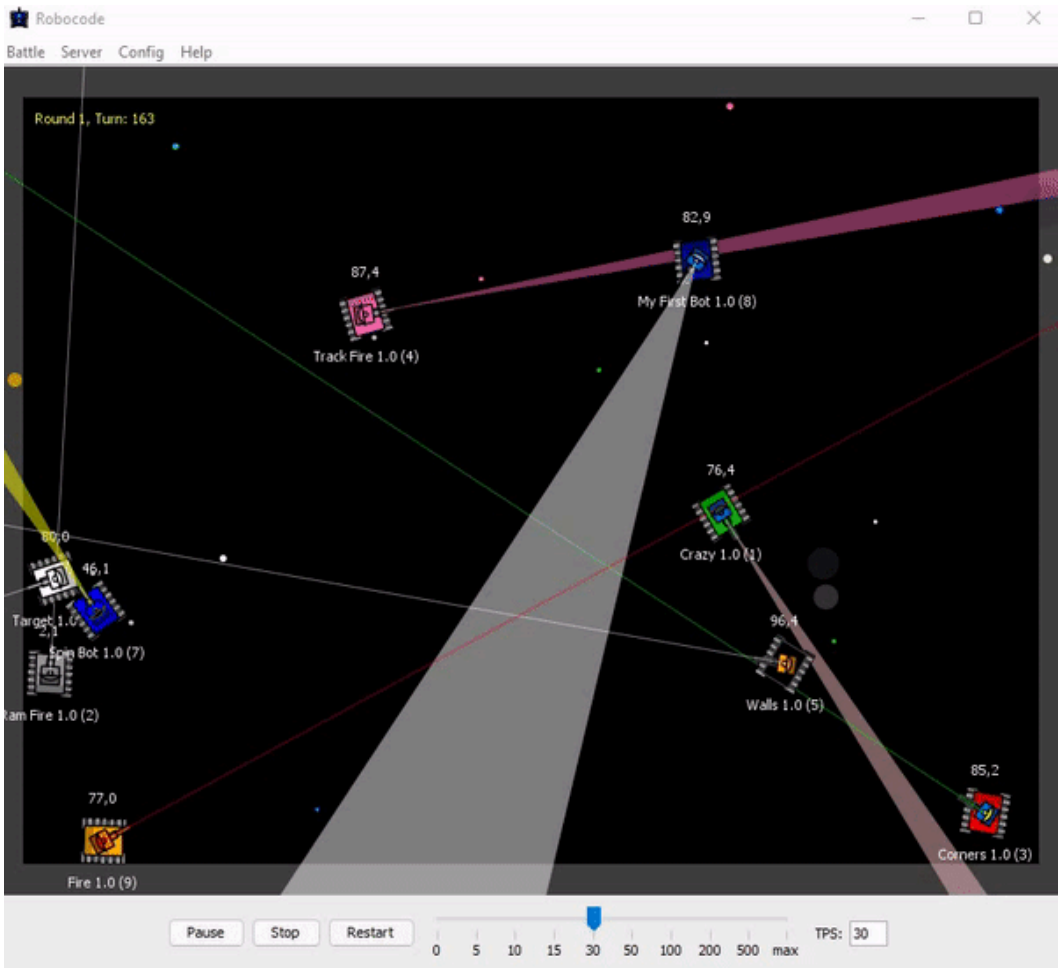
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

Daftar Isi

Bab 1: Deskripsi Tugas



Gambar 1 Robocode Tank Royale

Robocode adalah permainan pemrograman yang bertujuan untuk membuat kode bot dalam bentuk tank virtual untuk berkompetisi melawan bot lain di arena. Pertempuran Robocode berlangsung hingga bot-bot bertarung hanya tersisa satu seperti permainan Battle Royale, karena itulah permainan ini dinamakan Tank Royale. Nama Robocode adalah singkatan dari "Robot code," yang berasal dari versi asli/pertama permainan ini. Robocode Tank Royale adalah evolusi/versi berikutnya dari permainan ini, di mana bot dapat berpartisipasi melalui Internet/jaringan. Dalam permainan ini, pemain berperan sebagai programmer bot dan tidak memiliki kendali langsung atas permainan. Pemain hanya bertugas untuk membuat program yang menentukan logika atau "otak" bot. Program yang dibuat akan berisi instruksi tentang cara bot bergerak, mendeteksi bot lawan, menembakkan senjatanya, serta bagaimana bot bereaksi terhadap berbagai kejadian selama pertempuran.

Pada Tugas Besar pertama Strategi Algoritma ini, mahasiswa diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Tentunya mahasiswa harus menggunakan **strategi *greedy*** dalam membuat bot ini.

Komponen-komponen dari permainan ini antara lain:

1. Rounds dan Turns

   Pertempuran dapat terdiri dari beberapa rounds. Secara default, satu pertempuran berisi 10 rounds, di mana setiap rounds akan memiliki pemenang dan yang kalah.

   Setiap round dibagi menjadi beberapa turns, yang merupakan unit waktu terkecil. Satu turn adalah satu ketukan waktu dan satu putaran permainan. Jumlah turn dalam satu round tergantung pada berapa lama waktu yang dibutuhkan hingga hanya tersisa bot terakhir yang bertahan.

   Pada setiap turn, sebuah bot dapat:

   - Menggerakkan bot, memindai musuh, dan menembakkan senjata.
   - Bereaksi terhadap peristiwa seperti saat bot terkena peluru atau bertabrakan dengan bot lain atau dinding.
   - Perintah untuk bergerak, berputar, memindai, menembak, dan sebagainya dikirim ke server untuk setiap turn.

   Perlu diperhatikan bahwa API (Application Programming Interface) bot resmi secara otomatis mengirimkan niat bot ke server di balik layar, sehingga Anda tidak perlu mengkhawatirkannya, kecuali jika Anda membuat API Bot sendiri.

   Pada setiap turn, bot akan secara otomatis menerima informasi terbaru tentang posisinya dan orientasinya di medan perang. Bot juga akan mendapatkan informasi tentang bot musuh ketika mereka terdeteksi oleh pemindai.

   Perlu diketahui bahwa game engine yang akan digunakan pada tugas besar ini tidak mengikuti aturan default mengenai komponen Round & Turns.

2. Batas Waktu Giliran

   Penting untuk dicatat bahwa setiap bot memiliki batas waktu untuk setiap turn yang disebut turn timeout, biasanya antara 30-50 ms (dapat diatur sebagai aturan pertempuran). Ini berarti bahwa bot tidak bisa mengambil waktu sebanyak yang mereka inginkan untuk bergerak dan menyelesaikan turn saat ini.

   Setiap kali turn baru dimulai, penghitung waktu ulang diatur ulang dan mulai berjalan. Jika batas waktu tercapai dan bot tidak mengirimkan pergerakannya untuk turn tersebut, maka tidak ada perintah yang dikirim ke server. Akibatnya, bot akan melewatkan turn tersebut. Jika bot melewatkan turn, ia tidak akan bisa menyesuaikan gerakannya atau menembakkan senjatanya karena server tidak menerima perintah tepat waktu sebelum turn berikutnya dimulai.

3. Energi

   Semua bot memulai permainan dengan jumlah energi awal sebanyak 100 poin energi.

   - Bot akan kehilangan energi jika ditembak atau ditabrak oleh bot musuh.
   - Bot juga akan kehilangan energi jika menembakkan meriamnya.
   - Bot akan mendapatkan energi jika peluru dari meriamnya mengenai musuh. Energi yang didapat akan lebih banyak 3 kali lipat dari energi yang digunakan untuk menembakkan peluru.
   - Bot dengan energi nol akan dinonaktifkan dan tidak bisa bergerak. Jika bot terkena serangan dalam keadaan ini, bot akan hancur.

4. Peluru

   Semakin banyak energi (daya tembak) yang digunakan untuk menembakkan peluru, semakin berat peluru tersebut dan semakin lambat gerakannya. Namun, peluru yang lebih berat juga menghasilkan lebih banyak kerusakan dan memungkinkan bot mendapatkan lebih banyak energi saat mengenai bot musuh.

   Seperti disebutkan sebelumnya, peluru yang lebih berat akan bergerak lebih lambat. Ini berarti akan membutuhkan waktu lebih lama untuk mencapai target, meningkatkan risiko peluru tidak mengenai sasaran. Sebaliknya, peluru yang lebih ringan bergerak lebih cepat, sehingga lebih mudah mengenai target, tetapi peluru ringan tidak memberikan banyak poin energi saat mengenai bot musuh.

5. Panas Meriam (Gun Heat)

   Saat menembakkan peluru, meriam akan menjadi panas. Peluru yang lebih berat menghasilkan lebih banyak panas dibandingkan peluru yang lebih ringan. Ketika meriam terlalu panas, bot tidak dapat menembak hingga suhu meriam turun ke nol. Selain itu, meriam juga sudah dalam keadaan panas di awal round dan perlu waktu untuk mendingin sebelum bisa digunakan untuk pertama kalinya.

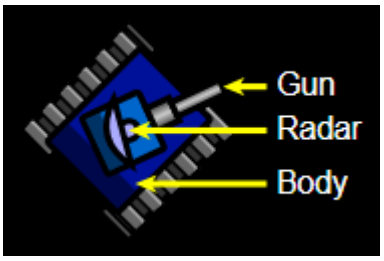
6. Tabrakan

   Perlu diperhatikan bahwa bot akan menerima kerusakan jika menabrak dinding (batas arena), yang disebut wall damage. Hal yang sama juga terjadi jika bot bertabrakan dengan bot lain.

   Jika bot menabrak bot musuh dengan bergerak maju, ini disebut ramming (menabrak dengan sengaja), yang akan memberikan sedikit skor tambahan bagi bot yang menyerang.

7. Bagian Tubuh Tank

Tubuh tank terdiri dari 3 bagian:



Body adalah bagian utama dari tank yang digunakan untuk menggerakkan tank.

Gun digunakan untuk menembakkan peluru dan dapat berputar bersama *body* atau independen dari *body*.

Radar digunakan untuk memindai posisi musuh dan dapat berputar bersama *body* atau independen dari *body*.

8. Pergerakan

   Bot dapat bergerak maju dan mundur hingga kecepatan maksimum. Dibutuhkan beberapa giliran untuk mencapai kecepatan maksimum. Bot dapat mengalami percepatan maksimum sebesar 1 unit per giliran dan pengereman dengan perlambatan maksimum 2 unit per giliran. Percepatan dan perlambatan maksimum tidak bergantung pada kecepatan bot saat itu.

9. Berbelok

   Seperti yang disebutkan sebelumnya, bagian tubuh, turret (meriam), dan radar dapat berputar secara independen satu sama lain. Jika turret atau radar tidak diputar, maka keduanya akan mengarah ke arah yang sama dengan tubuh bot.

   Setiap bagian tubuh memiliki kecepatan putar yang berbeda. Radar adalah bagian tercepat dan dapat berputar hingga 45 derajat per giliran, yang berarti dapat berputar 360 derajat dalam 8 giliran. Turret dan meriam dapat berputar hingga 20 derajat per giliran.

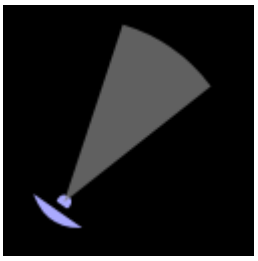
   Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok.

   Perlu diperhatikan bahwa tidak ada energi yang dikonsumsi saat bot bergerak atau berbelok.

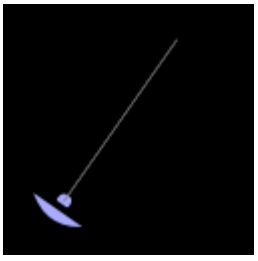
10. Pemindaian

    Aspek penting dalam Robocode adalah memindai bot musuh menggunakan radar. Radar dapat mendeteksi bot dalam jangkauan hingga 1200 piksel. Musuh yang berada lebih dari 1200 piksel dari bot tidak dapat terdeteksi atau dipindai oleh radar.

Penting untuk diperhatikan bahwa sebuah bot hanya dapat memindai bot musuh yang berada dalam jangkauan sudut pemindaian (scan arc)-nya. Sudut pemindaian ini merupakan "sapuan radar" dari arah radar sebelumnya ke arah radar saat ini dalam satu giliran.



Jika radar tidak bergerak dalam suatu giliran, artinya radar tetap mengarah ke arah yang sama seperti pada giliran sebelumnya, maka sudut pemindaian akan menjadi nol derajat, dan bot tidak akan dapat mendeteksi musuh.



Oleh karena itu, sangat disarankan untuk selalu mengubah arah radar agar tetap dapat memindai musuh.

11. Skor

Pada akhir pertempuran, setiap bot akan diranking berdasarkan total skor yang diperoleh masing-masing bot selama keseluruhan pertempuran. Tentunya, tujuan utama pada tugas besar ini adalah membuat bot yang memberikan skor setinggi mungkin. Berikut adalah rincian komponen skor pada pertempuran:

- **Bullet Damage:** Bot mendapatkan **poin sebesar *damage*** yang dibuat kepada bot musuh menggunakan peluru.
- **Bullet Damage Bonus:** Apabila peluru berhasil membunuh bot musuh, bot mendapatkan **poin sebesar 20% dari *damage*** yang dibuat kepada musuh yang terbunuh.
- **Survival Score:** Setiap ada bot yang mati, bot lainya yang masih bertahan pada ronde tersebut mendapatkan **50 poin.**
- **Last Survival Bonus:** Bot terakhir yang bertahan pada suatu ronde akan mendapatkan **10 poin** dikali dengan banyaknya musuh.
- **Ram Damage:** Bot mendapatkan **poin sebesar 2 kalinya *damage*** yang dibuat kepada bot musuh dengan cara menabrak.

- **Ram Damage Bonus:** Apabila musuh terbunuh dengan cara ditabrak, bot mendapatkan **poin sebesar 30% dari *damage*** yang dibuat kepada musuh yang terbunuh.

Skor akhir bot adalah akumulasi dari 6 komponen diatas. Perlu diperhatikan bahwa game akan menampilkan berapa kali suatu bot meraih peringkat 1, 2, atau 3 pada setiap ronde. Namun, hal ini tidak dihitung sebagai komponen skor maupun untuk perangkingan akhir. Bot yang dianggap menang pertempuran adalah bot dengan akumulasi skor tertinggi.

Bab 2: Landasan Teori

2.1 Algoritma *Greedy*

Algoritma Greedy merupakan metode yang populer dan relatif sederhana untuk memecahkan persoalan optimasi. Persoalan optimasi terbagi menjadi dua jenis, yaitu maksimasi (mencari nilai maksimum) dan minimasi (mencari nilai minimum).

Prinsip utama algoritma Greedy adalah "take what you can get now!", yang artinya mengambil pilihan yang tampak terbaik saat itu tanpa mempertimbangkan konsekuensi di masa depan. Algoritma ini membangun solusi langkah per langkah (step by step), di mana pada setiap langkah dibuat keputusan yang dianggap terbaik menurut kriteria tertentu.

2.1.1 Elemen-elemen Algoritma Greedy

Algoritma Greedy memiliki beberapa elemen penting:

1. Himpunan kandidat (C): Berisi elemen-elemen yang dapat dipilih untuk menyusun solusi, misalnya simpul/sisi dalam graf, job, task, atau objek lainnya.
2. Himpunan solusi (S): Berisi kandidat-kandidat yang sudah dipilih sebagai bagian dari solusi.
3. Fungsi solusi: Menentukan apakah himpunan solusi sudah membentuk solusi lengkap.
4. Fungsi seleksi: Strategi untuk memilih kandidat berdasarkan kriteria tertentu. Fungsi ini bersifat heuristik.
5. Fungsi kelayakan: Memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi.
6. Fungsi objektif: Fungsi yang akan dimaksimumkan atau diminimumkan.

2.1.2 Skema Umum Algoritma Greedy

```
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

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

2.1.3 Kelebihan dan Keterbatasan

1. Kelebihan:
    - Sederhana dan mudah diimplementasikan
    - Efisien dalam hal waktu komputasi
    - Cocok untuk persoalan dengan karakteristik optimal substructure
2. Keterbatasan:
    - Tidak selalu menghasilkan solusi optimal global
    - Tidak dapat mundur atau mengubah keputusan yang telah dibuat

Algoritma Greedy tidak selalu menghasilkan solusi optimal karena tidak mempertimbangkan semua kemungkinan dan hanya fokus pada pilihan terbaik saat itu.

2.2 Cara Kerja Program Robocode Tank Royale

2.2.1 Gambaran Umum Game

Robocode Tank Royale adalah permainan pemrograman di mana pemain membuat bot dalam bentuk tank virtual untuk berkompetisi melawan bot lain. Game ini terdiri dari beberapa rounds, dan setiap round dibagi menjadi turns. Pada setiap turn, bot dapat melakukan berbagai aksi seperti bergerak, memindai musuh, dan menembak.

2.2.2 Komponen Tank dan Aksinya

Tank dalam Robocode Tank Royale terdiri dari tiga bagian utama:

1. Body: Bagian utama tank untuk bergerak
2. Gun: Digunakan untuk menembakkan peluru
3. Radar: Digunakan untuk memindai posisi musuh

Bot dapat melakukan beberapa aksi dasar:

1. Bergerak: Maju dan mundur dengan kecepatan yang dapat diatur
2. Berbelok: Memutar body, gun, dan radar dengan kecepatan yang berbeda-beda
3. Menembak: Mengeluarkan peluru dengan kekuatan yang dapat diatur
4. Memindai: Menggunakan radar untuk mendeteksi posisi musuh

2.2.3 Implementasi Algoritma Greedy dalam Bot

Dalam konteks Robocode Tank Royale, algoritma Greedy dapat diimplementasikan dengan langkah-langkah berikut:

1. Identifikasi Elemen Greedy:
    - Himpunan kandidat: Aksi-aksi yang dapat dilakukan bot (bergerak, berputar, menembak)
    - Himpunan solusi: Urutan aksi yang dipilih untuk dilakukan
    - Fungsi seleksi: Strategi untuk memilih aksi terbaik berdasarkan kondisi saat ini
    - Fungsi kelayakan: Memeriksa apakah aksi dapat dilakukan
    - Fungsi objektif: Memaksimalkan skor akhir
2. Penerapan dalam Siklus Game:

    Pada setiap turn, bot:

    - Mengumpulkan informasi dari lingkungan (posisi musuh, kondisi bot)
    - Mengevaluasi pilihan aksi berdasarkan strategi Greedy
    - Memilih aksi terbaik dan melaksanakannya
3. Contoh Strategi Greedy yang Dapat Diterapkan:
    - Memilih target berdasarkan jarak terdekat
    - Menembak dengan kekuatan optimal berdasarkan jarak target
    - Memilih posisi yang meminimalkan kemungkinan terkena tembakan
    - Mengejar musuh dengan energi terendah

2.2.4 Menjalankan Bot

Untuk menjalankan bot dalam Robocode Tank Royale, diperlukan beberapa langkah:

1. Pengembangan Bot:
    - Membuat kode bot dalam bahasa C# menggunakan API Robocode Tank Royale
    - Mengimplementasikan strategi Greedy dalam kode
2. Kompilasi:
    - Mengkompilasi kode menjadi file executable
3. Menjalankan Game:
    - Menjalankan game engine Robocode Tank Royale
    - Menghubungkan bot ke game engine
    - Memulai pertempuran antara bot
4. Evaluasi Performa:
    - Menganalisis hasil pertempuran berdasarkan skor yang diperoleh
    - Menyesuaikan strategi Greedy jika diperlukan

Bot dalam Robocode Tank Royale melakukan aksinya berdasarkan logika yang telah diprogram. Pada setiap turn, bot menerima informasi dari lingkungan, mengolah informasi tersebut, dan memutuskan aksi yang akan dilakukan. Keputusan ini dibuat berdasarkan strategi Greedy yang telah diimplementasikan, dengan tujuan utama memaksimalkan skor akhir.

Dalam pengembangan bot, tantangan utama adalah merancang fungsi seleksi yang efektif untuk memilih aksi terbaik pada setiap turn. Strategi Greedy yang baik akan memperhitungkan berbagai faktor seperti posisi musuh, jarak, energi, dan kondisi pertempuran saat itu.

Bab 3: Aplikasi Strategi *Greedy*

3.1 Mapping Persoalan Robocode Tank Royale ke Elemen Algoritma Greedy

Dalam implementasi bot pada Robocode Tank Royale, strategi greedy dapat dimodelkan dengan elemen-elemen berikut:

- Himpunan Kandidat: Semua aksi yang dapat diambil bot pada setiap langkah, termasuk bergerak, menembak, dan menghindari musuh.
- Himpunan Solusi: Serangkaian keputusan yang diambil oleh bot selama pertandingan untuk bertahan hidup dan memberikan damage maksimal.
- Fungsi Solusi: Evaluasi apakah keputusan yang diambil bot memberikan keuntungan terbaik dalam pertempuran.
- Fungsi Seleksi: Memilih aksi terbaik berdasarkan informasi saat ini, seperti menargetkan musuh dengan HP rendah atau menghindari tembakan musuh.
- Fungsi Kelayakan: Memastikan bahwa aksi yang diambil memungkinkan bot tetap hidup dan tidak menempatkan dirinya dalam situasi berbahaya (misalnya, menabrak dinding atau tetap diam dalam jangkauan tembakan musuh).
- Fungsi Objektif: Memaksimalkan survival rate dan damage output, dengan mempertimbangkan efisiensi pergerakan dan tembakan.

3.2 Eksplorasi 4 Alternatif Solusi *Greedy*

1. *Greedy by Survival*: Solusi ini mengutamakan pergerakan random secara greedy mempersulit musuh memprediksi gerakan/menembakkan peluru secara akurat serta pengaturan *damage* tembakan untuk efisiensi energi agar dapat bertahan hidup lebih lama.
    - Himpunan Kandidat
        - Menembak dengan kekuatan berbeda berdasarkan jarak musuh (3 jika <200, 2 jika <500, 1 jika lebih jauh).
        - Bergerak maju atau mundur dalam jarak acak antara 800-1600 unit untuk menghindari pola yang mudah diprediksi.
        - Mengubah arah secara acak antara 30-90 derajat setelah bergerak maju atau mundur.
        - Memutar turret secara konstan (180 derajat ke kiri, lalu 180 derajat ke kanan) untuk memaksimalkan deteksi musuh dan mempersempit deteksi agar lebih efisien.
        - Menghindari tembok dengan mundur dan berbelok 80-120 derajat ketika menabrak.
        - Menghindari musuh dalam jarak dekat dengan mundur 300 unit jika energi terlalu rendah.
    - Himpunan Solusi
        - Mengoptimalkan damage dengan menembak lebih kuat ketika musuh berada dalam jarak dekat dan memiliki energi cukup.
        - Mengacak pola pergerakan untuk menghindari tembakan musuh dan mencegah prediksi pergerakan oleh lawan.
        - Memaksimalkan deteksi musuh dengan memutar turret secara terus-menerus.

    - Menjaga kelangsungan hidup dengan menghindari tembok dan bergerak menjauh jika energi rendah atau terkena serangan.
- Fungsi Solusi
    - Ketika musuh terdeteksi, bot akan menembak dengan kekuatan tertentu berdasarkan jarak dan energi yang tersisa.
    - Ketika terkena tembakan atau energi rendah, bot akan bergerak mundur dan menghindar secara acak untuk mengurangi risiko terkena tembakan berikutnya.
    - Ketika menabrak tembok, bot akan mundur dan berbelok secara acak antara 80-120 derajat untuk menghindari terjebak.
    - Ketika tidak menemukan musuh, bot akan tetap bergerak maju/mundur dan memutar turret untuk mencari lawan.
- Fungsi Seleksi
    - Saat melihat musuh:
        - Jika energi > 30 atau jarak < 200 → pilih tembakan kuat (1-3).
        - Jika energi di antara 10-30 → pilih tembakan lemah (0.5).
        - Jika energi sangat rendah → hindari pertempuran dengan mundur 300 unit.
    - Saat terkena tembakan: pilih sudut acak 60-120 derajat dan mundur 300 unit untuk menghindari tembakan lanjutan.
    - Saat menabrak dinding: pilih sudut acak 80-120 derajat untuk memutar dan menjauh dari tembok.
- Fungsi Kelayakan
    - Bot harus selalu bergerak agar tidak menjadi target statis yang mudah ditembak.
    - Bot harus menjaga energi dengan tidak menembak berlebihan saat energi rendah.
    - Bot harus segera mundur jika dalam bahaya (terlalu dekat dengan musuh saat energi rendah).
    - Bot harus tetap mencari musuh dengan memutar turret jika tidak ada target terdeteksi.
- Fungsi Objektif
    - Damage Maksimal: Dengan memanfaatkan serangan optimal berdasarkan jarak dan energi, bot bisa memberikan serangan efektif tanpa kehabisan tenaga terlalu cepat.
    - Survival: Dengan pola pergerakan acak dan strategi mundur, bot dapat bertahan lebih lama dalam pertempuran.
    - Efisiensi Pergerakan: Dengan bergerak maju/mundur dalam jarak acak 800-1600 unit, bot menghindari pola yang mudah diprediksi dan tetap fleksibel di arena.

2. *Greedy by Maximum Firepower*: Solusi ini mengutamakan damage output maksimal dengan selalu menggunakan firepower tertinggi.
    - Himpunan kandidat
        - Menembak selalu dengan kekuatan penuh setiap kali musuh tertedeksi
        - Bergerak dalam pola pentagon teratur
        - Mengelak ketika terkena tembakan dengan bergerak tegak lurus dari arah peluru

    - berputar arah dan menjauh ketika menabrak dinding
- Himpunan Solusi
    - Mengoptimalkan damage dengan menembak dengan kekuatan maksimal setiap kali menemukan musuh
    - Pergerakan dalam pola pentagon teratur untuk mencari musuh dan menyulitkan musuh untuk menembak
    - Melakukan gerakan menghindar yang baik saat terkena tembakan
    - Pemulihan dengan cepat saat menabrak dinding
- Fungsi solusi
    - Ketika musuh terdeteksi, selalu menembak dengan kekuatan maksimal untuk menghasilkan damage terbesar
    - ketika terkena tembakan, selalu bergerak tegak lurus dari arah peluru untuk meminimalisir peluang terkena peluru tembakan berikutnya
    - ketika menabrak dinding, selalu berputar dengan sudut 120 derajat dan bergerak menjauh untuk menghindari terjebak
    - Mempertahankan pergerakan konstan dalam pola yang sudah ditentukan untuk menghindari serangan sekaligus mencari musuh
- Fungsi Seleksi
    - Saat melihat musuh: selalu pilih kekuatan tembakan maksimal tanpa pertimbangkan jarak atau energi
    - saat terkena tembakan: pilih sudut yang optimal untuk melakukan gerakan menghindar dari tembakan musuh
    - saat menabrak dinding: pilih sudut yang cukup besar untuk berputar menjauhi dinding
- Fungsi kelayakan
    - Bot harus selalu bergerak untuk menghindar menjadi target yang mudah
    - Bot harus segera menyelamatkan diri saat terkena tembakan atau menabrak dinding
    - Bot harus selalu menembak setiap kali melihat musuh untuk memaksimalkan peluang mengenai target
- Fungsi Objektif
    - Damage maksimal: dengan selalu menembakkan peluru dengan kekuatan penuh untuk memaksimalkan damage potensial
    - Survival: dengan melakukan kombinasi gerakan pola dan gerakan menghindar yang reaktif
    - Efisiensi Pergerakan: dengan pola pentagon yang teratur dan jarak menghindar yang terukur saat terkena tembakan

3. *Greedy by Brutal Movement:* Solusi ini mengutamakan pergerakan bot yang selalu berada dalam posisi menyerang serta melakukan langkah pergerakan sesuai dengan posisi musuh yang terdeteksi tanpa mempertimbangkan pergerakan musuh selanjutnya.
    - Himpunan Kandidat
        - Menembak dengan kekuatan penuh setiap kali melihat musuh.
        - Bergerak maju ke arah musuh tanpa mempertimbangkan posisi atau strategi lawan.
        - Tetap maju bahkan setelah menabrak musuh, lalu mundur sedikit dan menembak lagi.
        - Jika menabrak dinding, mundur sebentar lalu langsung kembali ke medan pertempuran tanpa perubahan strategi signifikan.

    - Terus memutar radar tanpa berhenti untuk mencari musuh secepat mungkin.
- Himpunan Solusi
    - Mengoptimalkan serangan dengan selalu menembak penuh (power 3) setiap kali melihat musuh.
    - Memastikan bot selalu bergerak menuju musuh untuk menjaga musuh tetap dalam jangkauan serangan.
    - Menggunakan kontak langsung (menabrak musuh) sebagai bagian dari strategi menyerang, bukan menghindar.
    - Tidak pernah menghindari pertempuran; jika bot menabrak dinding, hanya akan mundur sedikit lalu kembali menyerang.
- Fungsi Solusi
    - Ketika melihat musuh:
        - Tembak langsung dengan kekuatan penuh (Fire(3)).
        - Maju ke arah musuh (Forward(300) → Fire(3) → Forward(400) → Fire(3)).
    - Ketika menabrak musuh:
        - Dorong musuh dengan maju lebih jauh (Forward(500)).
        - Tetap menembak saat menabrak musuh (Fire(3)).
        - Mundur sedikit lalu lanjut menembak (Back(100) → Fire(3)).
    - Ketika menabrak dinding:
        - Mundur sebentar (Back(150)).
        - Berputar sedikit (TurnRight(90)).
        - Langsung kembali mencari musuh (Go()).
- Fungsi Seleksi
    - Saat melihat musuh: Selalu memilih untuk menyerang dengan fire power maksimal (3) tanpa mempertimbangkan sisa energi atau jarak.
    - Saat bertabrakan dengan musuh: Selalu memilih untuk menekan musuh dengan maju lebih jauh dan tetap menembak.
    - Saat menabrak dinding: Tidak memilih untuk menghindar lama, hanya mundur sebentar lalu kembali menyerang.
    - Saat tidak ada musuh dalam radar: Terus memutar radar untuk mencari target tanpa strategi bertahan.
- Fungsi Kelayakan
    - Bot harus terus bergerak menuju musuh, tidak boleh diam atau menunggu strategi musuh.
    - Bot harus selalu menembak saat melihat musuh, tanpa mempertimbangkan sisa energi.
    - Bot tidak boleh terlalu lama di pinggir arena; jika menabrak dinding, harus segera kembali ke tengah arena.
    - Bot harus menggunakan tabrakan sebagai bagian dari strategi menyerang, bukan bertahan.
- Fungsi Objektif
    - Maksimalkan Damage → Selalu menembak dengan kekuatan penuh dan memanfaatkan kontak fisik untuk menyerang.
    - Memastikan Kontak dengan Musuh → Selalu mendekat ke musuh untuk menjaga mereka dalam jangkauan tembakan.
    - Minimalkan Waktu Diam → Tidak ada strategi menunggu, bot harus selalu bergerak untuk mencari dan menyerang musuh.

        - Cepat Kembali ke Medan Tempur → Jika terkena dinding, langsung kembali menyerang tanpa strategi bertahan.
4. *Greedy by Aggression*: Solusi ini mengutamakan *damage output* dan *fast reaction movement.*
    - Himpunan Kandidat
        - Menembak dengan kekuatan berbeda berdasarkan jarak dan energi.
        - Bergerak secara acak untuk menghindari tembakan musuh.
        - Mundur atau memutar ketika terkena tembakan atau menabrak dinding/musuh.
        - Mencari musuh dengan menggerakkan radar 360°.
    - Himpunan Solusi
        - Mengoptimalkan damage dengan menembak setiap kali menemukan musuh.
        - Bergerak acak untuk menghindari serangan.
        - Menghindari dinding dan posisi yang rentan terhadap tembakan musuh.
        - Berusaha selalu agresif dengan menembak secara konstan.
    - Fungsi Solusi
        - Jika musuh terdeteksi dalam jarak dekat dan energi cukup, bot memilih menembak dengan *high power*.
        - Jika terkena tembakan atau menabrak, bot segera mengubah arah untuk menghindari serangan selanjutnya.
        - Jika tidak ada musuh yang terdeteksi, bot terus bergerak dan mencari target.
    - Fungsi Seleksi
        - Saat melihat musuh → Menembak dengan kekuatan tergantung jarak dan energi.
        - Saat terkena tembakan → Mundur dan berputar untuk menghindar.
        - Saat menabrak dinding → Mundur dan memutar arah.
        - Saat bertabrakan dengan bot lain → Mundur dan menargetkan musuh dengan turret.
    - Fungsi Kelayakan
        - Bot tidak boleh diam terlalu lama karena akan lebih mudah terkena tembakan.
        - Saat energi rendah, bot mengurangi kekuatan tembakan agar tetap bisa bertahan.
        - Menghindari tembok untuk mencegah posisi yang rentan.
    - Fungsi Objektif
        - Survival Rate → Dengan terus bergerak dan menghindari serangan.
        - Damage Output → Dengan selalu menembak musuh yang terdeteksi.
        - Efisiensi Pergerakan → Dengan kombinasi gerakan acak dan reaksi cepat terhadap serangan.

3.3 Analisis Efisiensi dan Efektivitas Alternatif Solusi Greedy

Dalam menganalisis efisiensi dan efektivitas dari empat alternatif solusi greedy yang telah dieksplorasi, kita akan menilai masing-masing strategi berdasarkan beberapa faktor utama, yaitu:

1. Efektivitas Serangan – Seberapa besar damage yang dapat diberikan kepada musuh.

2. Efisiensi Energi – Seberapa baik bot mengatur energi dalam pertempuran.
3. Kemampuan Bertahan Hidup – Seberapa lama bot dapat bertahan di arena.
4. Kemampuan Menghindar – Seberapa baik bot dapat menghindari serangan musuh.

3.3.1 Greedy by Survival

Strategi ini menyeimbangkan antara serangan dan pertahanan dengan menyesuaikan kekuatan tembakan berdasarkan jarak musuh dan tetap bergerak untuk menghindari tembakan lawan.

- Efektivitas Serangan: Cukup baik karena menyesuaikan damage berdasarkan jarak.
- Efisiensi Energi: Sangat efisien karena tidak selalu menggunakan kekuatan penuh.
- Kemampuan Bertahan Hidup: Sangat baik karena strategi ini memprioritaskan penghindaran dan penggunaan energi yang bijak.
- Kemampuan Menghindar: Baik, dengan pergerakan acak untuk menghindari prediksi lawan.

Kesimpulan: Strategi ini cukup efektif untuk bertahan dalam pertempuran yang berlangsung lama karena pengelolaan energi yang baik, meskipun damage yang diberikan mungkin lebih rendah dibandingkan strategi lain yang lebih agresif.

3.3.2 Greedy by Maximum Firepower

Strategi ini berfokus pada serangan dengan selalu menggunakan tembakan berkekuatan penuh tanpa mempertimbangkan efisiensi energi.

- Efektivitas Serangan: Sangat tinggi karena selalu menggunakan kekuatan maksimal.
- Efisiensi Energi: Rendah karena tidak mempertimbangkan penghematan energi.
- Kemampuan Bertahan Hidup: Cukup baik, tetapi berisiko jika energi cepat habis.
- Kemampuan Menghindar: Cukup baik dengan pola pergerakan pentagon dan gerakan menghindar yang reaktif.

Kesimpulan: Strategi ini sangat agresif dan dapat memberikan damage besar dalam waktu singkat. Namun, jika pertempuran berlangsung lama, bot ini berisiko kehabisan energi dan menjadi lebih rentan.

3.3.3 Greedy by Brutal Movement

Strategi ini sangat agresif, di mana bot selalu maju ke arah musuh tanpa mempertimbangkan taktik bertahan atau menghindar.

- Efektivitas Serangan: Sangat tinggi karena selalu mendekati musuh untuk menembak dengan kekuatan penuh.
- Efisiensi Energi: Rendah karena selalu menggunakan kekuatan maksimal.
- Kemampuan Bertahan Hidup: Rendah karena tidak ada strategi menghindar atau pengelolaan energi.
- Kemampuan Menghindar: Sangat rendah karena bot lebih fokus pada mendekati musuh daripada menghindar.

Kesimpulan: Strategi ini efektif dalam skenario serangan langsung, tetapi sangat lemah dalam jangka panjang karena kurangnya pertimbangan untuk bertahan hidup dan pengelolaan energi yang buruk.

3.3.4 Greedy by Aggression

Strategi ini mengkombinasikan serangan agresif dengan beberapa elemen pertahanan, seperti penghindaran ketika terkena tembakan atau menabrak dinding.

- Efektivitas Serangan: Baik karena tetap agresif dengan menembak setiap kali menemukan musuh.
- Efisiensi Energi: Sedang karena menyesuaikan kekuatan tembakan dengan jarak dan energi yang tersisa.
- Kemampuan Bertahan Hidup: Baik karena ada mekanisme penghindaran saat terkena tembakan.
- Kemampuan Menghindar: Baik karena bot bergerak secara acak dan menghindari tembakan musuh.

Kesimpulan: Strategi ini cukup seimbang antara serangan dan pertahanan. Meskipun tidak seefektif "Maximum Firepower" dalam memberikan damage besar secara langsung, bot ini lebih mampu bertahan dalam pertempuran yang lebih lama.

3.4 Strategi *Greedy* yang Dipilih

Strategi *Greedy by Survival* dipilih karena fokus utamanya adalah memastikan keberlangsungan hidup dalam peperangan dengan membuat keputusan yang mengutamakan kelangsungan hidup bot.

3.4.1 Alasan dan Pertimbangan Pemilihan

1. Memaksimalkan Kelangsungan Hidup – Dibandingkan strategi lain yang mungkin terlalu agresif atau pasif, *Greedy by Survival* menjaga keseimbangan antara eksplorasi dan penghindaran risiko, memastikan entitas dapat bertahan lebih lama.
2. Adaptif terhadap Lingkungan – Bot ini dapat menyesuaikan keputusan berdasarkan kondisi yang dihadapi, seperti ketersediaan sumber daya dan keberadaan ancaman.
3. Efektif dan Konsisten – Bot yang menggunakan strategi *Greedy by Survival* diperkirakan mampu memberikan hasil yang stabil karena mengutamakan langkah-langkah yang meningkatkan peluang bertahan.
4. Sederhana namun Optimal – Pendekatan *greedy* yang digunakan tidak memerlukan perhitungan kompleks, sehingga lebih efisien dalam implementasi tanpa mengorbankan efektivitas.

Dengan pertimbangan tersebut, strategi ini dipilih karena memberikan keseimbangan antara efisiensi, adaptasi, dan keberlanjutan dalam simulasi.

Bab 4: Implementasi dan Pengujian

4.1 Implementasi 4 Alternatif Solusi

4.1.1 Implementasi SurvivorBot (Main Bot)

```
// Pseudocode untuk SurvivorBot - Mengimplementasikan Greedy
by Survival

function Run()
    // Inisialisasi warna
    SetupColors(DarkGreen, Red, Yellow, White, Magenta)

    // Loop utama - pergerakan acak untuk bertahan hidup
    while (IsRunning)
        if (movingForward)
            Forward(Random(800, 1600))
            TurnRight(Random(30, 90))
        else
            Backward(Random(800, 1600))
            TurnRight(Random(30, 90))
        end if

        // Gerakkan radar untuk mencari musuh
        TurnGunLeft(180)
        TurnGunRight(180)
    end while
end function

function OnScannedBot(event)
    // Hitung jarak musuh
    distance = DistanceTo(event.X, event.Y)

    // Tentukan kekuatan tembakan berdasarkan jarak
    if (distance < 200)
        firePower = 3
    else if (distance < 500)
        firePower = 2
    else
        firePower = 1
    end if

    // Greedy: Menembak jika memiliki energi cukup atau musuh
sangat dekat
    if (Energy > 30 OR distance < 200)
        Fire(firePower)
    else if (Energy > 10)
        Fire(0.5) // Hemat energi, tapi tetap menyerang
    else
        // Jika energi rendah, menghindar dari musuh
        TurnRight(Random(60, 120))
```

```
        Backward(300)
    end if
end function

function OnHitWall(event)
    // Greedy: Menghindari tembok dengan mundur dan berputar
    Backward(100)
    TurnRight(Random(80, 120))
end function

function OnHitBot(event)
    if (event.IsRammed)
        // Jika bertabrakan langsung dengan musuh, mundur,
berputar, dan tembak
        Backward(150)
        TurnRight(Random(90, 180))
        Fire(3)
    else
        // Jika hanya bersentuhan dengan musuh, tetap
menembak
        Fire(2)
    end if
end function
```

4.1.2 Implementasi SmartGreedyBot (Alt-Bot 1)

```
// Pseudocode untuk SmartGreedyBot - Mengimplementasikan
Greedy by Brutal Movement

function Run()
    // Inisialisasi warna bot
    SetupColors(Red, Black, Red, Red, White)

    // Loop utama - bot tidak berpatroli, hanya mencari musuh
    while (IsRunning)
        TurnGunRight(360) // Terus-menerus mencari musuh
        Go() // Bergerak ke arah terakhir yang dituju
    end while
end function

function OnScannedBot(event)
    // Greedy: Selalu tembak dengan kekuatan penuh
    Fire(3)

    // Mengejar musuh secara agresif
    Forward(300)
    Fire(3)
```

```
    // Jika musuh masih jauh, terus maju dan tembak
    Forward(400)
    Fire(3)
end function

function OnHitBot(event)
    // Dorong musuh dan terus menyerang
    Forward(500)
    Fire(3)

    // Mundur sedikit dan tembak lagi
    Backward(100)
    Fire(3)
end function

function OnHitWall(event)
    // Jika menabrak dinding, mundur lalu belok untuk kembali
menyerang
    Backward(150)
    TurnRight(90)
    Go()
end function
```

4.1.3 Implementasi BotMuter (Alt-Bot 2)

```
// Pseudocode untuk BotMuter - Mengimplementasikan Greedy by
Maximum Firepower
function Run()
    // Inisialisasi warna
    SetupColors()

    // Loop utama - bergerak dalam pola pentagon
    while (IsRunning)
        // Pola pentagon (5 sisi) dengan rotasi 72° setiap
sudut
        Forward(100)
        TurnGunRight(135) // Gerakan gun untuk mencari musuh
        TurnRight(72)
        TurnGunRight(135)
        Forward(100)
        TurnRight(72)
        TurnGunRight(135)
        Forward(100)
```

```
        TurnRight(72)
        TurnGunRight(135)
        Forward(100)
        TurnRight(72)
        TurnGunRight(135)
        Forward(100)
        TurnRight(72)
        TurnGunRight(135)
        TurnRight(72)
    end while
end function

function OnScannedBot(event)
    // Greedy: Selalu tembak dengan kekuatan maksimal tanpa
    // mempertimbangkan jarak atau energi
    Fire(3)
end function

function OnHitByBullet(event)
    // Greedy: Pilih sudut optimal untuk menghindar
    bearing = CalcBearing(event.Bullet.Direction)

    // Berputar tegak lurus (90°) terhadap arah peluru
    TurnLeft(90 - bearing)

    // Bergerak menjauh dengan jarak tetap
    Forward(120)
end function

function OnHitWall(event)
    // Greedy: Berputar dengan sudut tetap untuk menjauhi
dinding
    TurnRight(120)
    Forward(200)
end function
```

4.1.4 Implementasi Berserker (Alt-Bot 3)

```
// Berserker Bot - Strategi Agresif dan Random Movement
```

```
function Run()
    // Set warna bot
    SetBodyColor(Gray)
    SetTurretColor(Black)
    SetRadarColor(White)
    SetScanColor(Red)
    SetBulletColor(Black)

    // Loop utama selama bot masih aktif
    while (IsRunning)
        // Putar radar 360 derajat untuk mendeteksi musuh
        TurnGunLeft(360)

        // Tentukan arah dan gerakan bot secara acak
        move = Random(0, 10)

        // Acak arah putaran bot
        TurnLeft(Random(70, 120))

        // Tentukan apakah bot maju atau mundur
        if (move > 5)
            Forward(Random(450, 700))
        else if (move > 0)
            Backward(Random(450, 700))
        else
            StayStill()
        end if
    end while
end function

function OnScannedBot(event)
    // Hitung jarak ke musuh
    distance = DistanceTo(event.X, event.Y)

    // Tentukan kekuatan tembakan berdasarkan jarak dan
energi
    if (distance < 180 AND Energy > 50)
        Fire(3)  // Tembakan kuat jika musuh dekat dan energi
cukup
    else if (distance < 280 AND Energy > 20)
        Fire(1.5)  // Tembakan sedang untuk musuh jarak
menengah
    else if (Energy > 1)
        Fire(1)  // Tembakan lemah jika energi terbatas
    else
        Fire(Energy * 0.5)  // Tembakan terakhir sebelum
energi habis
    end if
end function
```

```
function OnHitByBullet(event)
    // Jika terkena peluru, bergerak untuk menghindar
    Backward(Random(100, 150))
    TurnRight(Random(75, 90))
end function

function OnHitWall(event)
    // Jika menabrak dinding, mundur dan ubah arah
    Backward(100)
    TurnRight(200)
    TurnGunRight(200)
end function

function OnHitBot(event)
    // Jika bertabrakan dengan bot lain, mundur dan targetkan
musuh
    Backward(Random(100, 200))
    TurnGunRight(200)
end function
```

4.2 Struktur Data, Fungsi, dan Prosedur pada SurvivorBot

4.2.1 Struktur Data

SurvivorBot menggunakan beberapa variabel dan struktur data sederhana:

- `Random rng`: Objek dari kelas Random untuk menghasilkan angka acak yang digunakan dalam pergerakan bot.
- `bool movingForward`: Variabel boolean yang menentukan apakah bot sedang bergerak maju atau mundur.

4.2.2 Fungsi dan Prosedur

1. Main()

   ```
   static void Main() {
       new SurvivorBot().Start();
   }
   ```

   Fungsi utama yang menjalankan bot dengan membuat instance dari SurvivorBot dan memanggil Start().

2. Konstruktor SurvivorBot()

   ```
   SurvivorBot() :
   base(BotInfo.FromFile("SurvivorBot.json")) { }
   ```

   Konstruktor ini memanggil informasi bot dari file konfigurasi SurvivorBot.json.

3. Run() - Fungsi Utama Pergerakan Bot

```
public override void Run()
```

- Mengatur warna bot.
- Loop utama memastikan bot tetap bergerak dengan pola acak (maju atau mundur) dan berputar.
- Senjata diputar bolak-balik 180 derajat untuk mencari musuh.

4. OnScannedBot(ScannedBotEvent e) - Menangani Musuh yang Terdeteksi

```
public override void OnScannedBot(ScannedBotEvent e)
```

- Menghitung jarak musuh menggunakan DistanceTo(e.X, e.Y).
- Menentukan kekuatan tembakan (firePower) berdasarkan jarak.
- Strategi menembak:
  - Jika energi bot tinggi atau musuh dekat, menembak dengan kekuatan besar.
  - Jika energi menengah, menembak dengan kekuatan kecil.
  - Jika energi rendah, bot menghindar dengan mundur dan berputar secara acak.

5. OnHitWall(HitWallEvent e) - Menghindari Dinding

```
public override void OnHitWall(HitWallEvent e)
```

Jika bot menabrak dinding, ia akan mundur 100 unit dan berbelok acak antara 80-120 derajat untuk menghindari tabrakan lebih lanjut.

6. OnHitBot(HitBotEvent e) - Menangani Tabrakan dengan Bot Lain

```
public override void OnHitBot(HitBotEvent e)
```

- Jika bot bertabrakan dengan musuh:
  - Jika tabrakan adalah hasil dari dorongan (IsRammed), bot akan mundur, berputar, dan menembak dengan kekuatan maksimal.
  - Jika tidak, bot tetap menembak dengan kekuatan menengah.

4.2.3 Kesimpulan

SurvivorBot menggunakan pendekatan greedy dengan strategi utama:

1. Bertahan dengan pergerakan acak: Menghindari serangan dengan terus bergerak.
2. Menyerang dengan efisiensi energi: Menembak berdasarkan jarak dan energi tersisa.
3. Menghindari dinding dan tabrakan: Menggunakan strategi mundur dan berputar untuk tetap bertahan dalam permainan.

4.3 Pengujian dan Analisis

Setiap bot akan diuji coba oleh 2 skema percobaan. Hasil pengujian ini akan digunakan untuk menganalisis dan membuktikan strategi greedy terbaik diantara 4 bot yang telah dibuat.

4.3.1 Skema 1: Battle Royale

Skema ini menguji coba 4 bot (yang telah diimplementasikan dengan setiap solusi greedy yang dibuat) untuk melawan satu sama lain dan juga 11 bot sampel untuk mengetahui kemampuan setiap bot apabila dihadapi lawan yang banyak.

Bot sampel terdiri dari:

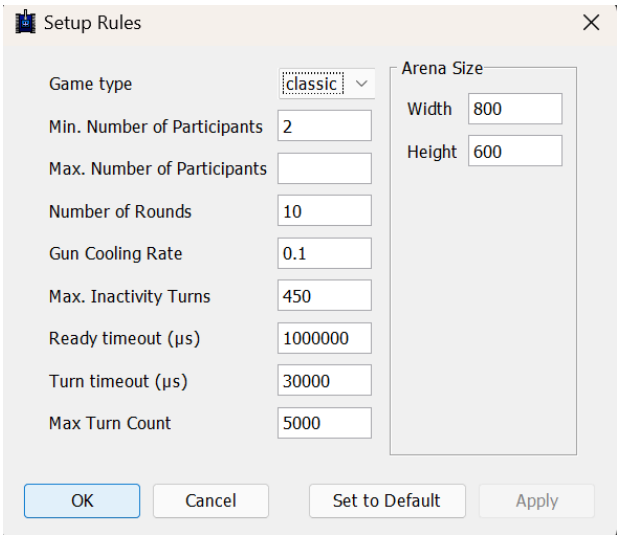
1) Corners: Bot yang menyerang dari corner.
2) Crazy: Gerakannya tidak dapat diprediksi.
3) Fire: Menembak musuh.
4) MyFirstBot: Bergerak maju-mundur dan memutar turret untuk menembak.
5) PaintingBot: Menyimpan posisi musuh yang terdeteksi dan menggambar lingkaran merah di lokasi terakhirnya.
6) SpinBot: Berputar terus-menerus sambil menembak dengan kekuatan tinggi saat mendeteksi musuh.
7) Target: Bot target untuk latihan menembak.
8) TrackFire: Bot stasioner yang fokus pada akurasi tembakan.
9) VelocityBot: Mengubah kecepatan dan arah secara berkala untuk menghindari tembakan.
10) RamFire: Mengutamakan serangan jarak dekat dengan menabrak musuh.
11) Walls: Bergerak di sepanjang dinding sambil menembak ke tengah arena.

Sistem penilaian:

- Setiap bot bertarung 25 kali dalam satu arena berisi 11+4 bot total.
- Skor diberikan berdasarkan posisi akhir dalam pertandingan.

Sistem skor battle royale:

- Juara 1 → 5 poin
- Juara 2 → 3 poin
- Juara 3 → 2 poin
- Juara > 3 → 0 poin



4.3.1.1 Gambar setup rules untuk pertarungan skema 1.

4.3.2 Skema 2: 1v1v1v1

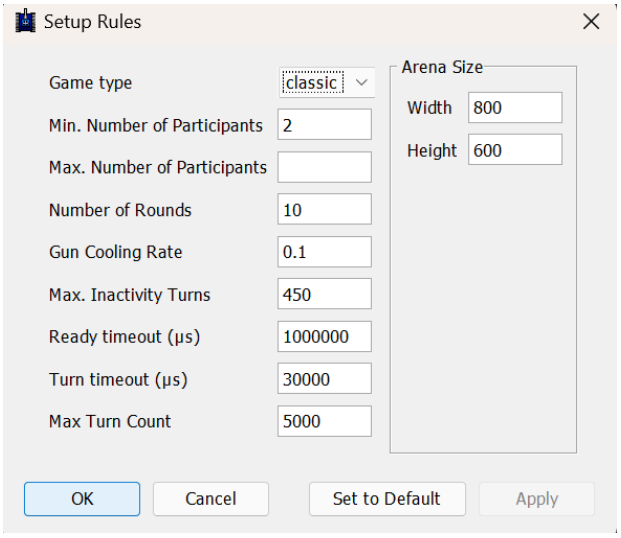
Pada skema ini, setiap bot (yang telah diimplementasikan solusi greedy yang dibuat) akan bertarung satu sama lain.

Sistem penilaian:

- Pertarungan dilakukan sebanyak 25 kali
- Skor diberikan berdasarkan posisi akhir dalam pertandingan.

Sistem skor 1v1v1v1:

- Juara 1 → 3 poin
- Juara 2 → 2 poin
- Juara 3 → 1 poin
- Juara 4 → 0 poin



4.3.2.1 Gambar setup rules untuk pertarungan skema 2.

4.3.3 Skor Akhir

Skor akhir dan peringkat akan ditentukan berdasarkan (total skor battle royale + total skor 1v1v1v1).

4.3.4 Hasil Uji Coba

Bot	Skor 1v1v1v1 (25 match)		Skor Battle Royale (25 match)		Skor Total	Peringkat
SmartGreedyBot	1+1+1+1+1+1+1	7	4.00	4	11.00	4
BotMuter	1+1+1+1+1+2+1+2+1+3+1+1+1+1+1+1+2+1+1+1	25	3+3+3+3+2+3+2+5+3+3+2	32	57.00	3

SurvivorBot	3+3+2+3+3+3+2+2+3+1+2+3+2+2+1+3+3+2+3+3+3+3+3+3+2+2	65	5+3+3+3+2+2+3+5+5+5+3+3+2+2+3	49	114.00	1
Berserker	2+2+3+2+1+2+3+3+2+3+3+1+3+3+2+2+2+3+2+2+2+2+2+3+3	58	2+2+2+3+3+3+2+3+3+5+3+5+2	38	96.00	2

4.3.5 Analisis Hasil Uji Coba

1. SmartGreedyBot

Kelebihan:

- Menghasilkan damage yang maksimal karena setiap tembakan bernilai 3 efektif untuk menghabisi musuh secara langsung.
- Mendapat keuntungan di medan perang yang ramai karena kemungkinan tembakan kena lebih besar.
- Ketika mengenai musuh secara langsung, bot akan selalu mengejar/menahan musuh tersebut menyebabkan musuh kalah dengan telak

Kekurangan:

- Gerakannya yang terlalu lambat dan prediktif membuat bot mudah terkena peluru serangan
- Rotasi turret kurang cepat dan akurat sehingga damage output tidak sebaik bot-bot yang lain.
- Di arena yang sedikit musuh akan sangat kesusahan karena hanya bergerak lurus menuju musuh yang terakhir di-scan dan kemungkinan meleset sangat besar.
- Rawan menabrak dinding dan kehabisan energi secara sia-sia jika musuh yang di-lock menghindar.

Kesimpulan: SmartGreedyBot belum dapat memberikan solusi greedy dengan nilai optimal. Bot ini memilih solusi terbaik tanpa mempertimbangkan efisiensi energi, akurasi, atau kemungkinan serangan balik. Oleh karena itu, strategi ini kurang optimal gerakannya mudah ditebak dan rentan terkena serangan bot lain.

Perbaikan:

- Menambahkan sistem penghematan energi dalam algoritma.

- Meningkatkan fleksibilitas gerakan agar tidak rentan terhadap serangan lawan.

2. BotMuter

Kelebihan:

- Gerakannya yang relatif lambat membuatnya lebih mudah mengarahkan pelurunya.
- Walaupun lambat, manuver bot ini dalam medan masih cukup efisien untuk menghindar serangan.
- Setiap tembakan pelurunya memiliki damage maksimal. Hal ini sangat efektif dalam pertempuran jarak dekat. Sehingga, bot ini lebih efektif dalam pertempuran yang sempit.

Kekurangan:

- Gerakannya yang relatif lambat membuatnya mudah ditarget oleh bot-bot yang memiliki sistem targeting yang canggih
- Tidak ada gerakan apabila bertabrakan dengan bot.
- Pada medan perang yang tidak sempit, BotMuter mengalami kesulitan apabila dihadapi bot dengan Gerakan yang cepat. Hal ini ditunjukkan dengan skor 1v1v1v1 yang masih kalah jauh dengan bot Berserker and SurvivorBot.
- Gerakan tidak susah untuk diprediksi
- Penembakan selalu dengan energi maksimal tanpa pertimbangan jarak dan efisiensi energi.

Kesimpulan: Bot ini memiliki strategi yang lebih baik daripada SmartGreedyBot, tetapi masih kalah jauh dibanding SurvivorBot dan Berserker. BotMuter kurang mempertimbangkan penghematan energi dan mekanisme pertahanan yang lebih efektif, sehingga belum dapat menghasilkan solusi greedy dengan nilai optimal.

Perbaikan:

- Menambahkan faktor *random* dalam pergerakannya agar lebih sulit untuk ditarget oleh musuh
- Menambahkan sistem penghematan energi dalam menembak
- Menambahkan strategi menghindar yang lebih efektif saat diserang atau bertabrakan.

3. SurvivorBot

Kelebihan:

- Unggul dalam skema Battle Royale dan 1v1v1v1.
- Gerakan memutar arena yang random tetapi cepat sehingga lebih sulit untuk terkena serangan musuh serta diprediksi musuh.
- Energi penembakan menyesuaikan berdasarkan jarak dan energi yang dimilikinya sehingga penembakan lebih efisien.
- Memiliki *counter movement* yang baik ketika menabrak dinding atau menabrak bot lain.
- Memiliki metode penyerangan dan pertahanan yang paling seimbang di antara bot lain.

Kekurangan:

- Gerakan terkadang repetitif apabila medan lumayan luas sehingga mudah ditarget oleh bot yang memiliki sistem targeting yang canggih.

Kesimpulan: SurvivorBot adalah bot paling efektif, baik dalam pertarungan kecil maupun besar. Meskipun tidak sempurna, SurvivorBot sudah memiliki algoritma greedy yang memiliki solusi yang paling dekat dalam memberikan nilai optimal.

Perbaikan:

- Menambahkan pola gerakan agar lebih sulit diprediksi di medan luas.

4. Berserker

Kelebihan:

- Memaksimalkan kesempatan mendapat poin dalam pertarungan jarak dekat dengan menembak dengan energi maksimal.
- Pergerakan yang cepat dan memiliki aspek *random* sehingga lebih sulit untuk ditarget oleh bot lain.
- Menyesuaikan energi penembakan apabila jarak target jauh untuk menghemat energi.

Kekurangan:

- Kadang pergerakan terlalu random yang menyebabkan *self inflicted damage* dengan menabrakan dirinya ke dinding berkali-kali.
- Gerakan yang terlalu cepat berpengaruh buruk terhadap akurasi penembakan.
- Lemah jika menghadapi bot yang memiliki akurasi penembakan tinggi.

Kesimpulan: Bot ini memiliki strategi agresif yang efektif. Namun, bot Berserker pergerakan bot Berserker kurang teratur sehingga pertahanannya kurang maksimal.

Perbaikan:

- Membuat gerakan lebih terstruktur agar tidak terlalu random untuk menghindari kerugian energi akibat menabrak dinding.
- Meningkatkan akurasi penembakan yang dapat menyesuaikan kecepatan gerakan.

Bab 5: Kesimpulan dan Saran

5.1 Kesimpulan

Keempat bot yang telah diuji menerapkan pendekatan algoritma greedy dengan strategi yang berbeda-beda. SmartGreedyBot dan BotMuter memiliki kelemahan dalam fleksibilitas gerakan serta efisiensi energi. Hal ini membuatnya kurang optimal dalam berbagai kondisi pertempuran. BerserkerBot menunjukkan hasil yang lebih baik dengan gerakan cepat dengan metode penembakan yang agresif. Hal tersebut menjadikannya efektif dalam pertempuran medan luas maupun sempit. Namun, pola gerakannya yang terlalu acak dan kurang terstruktur seringkali menyebabkan pemborosan energi. Selain itu, tidak ada penyesuaian arah penembakan untuk menjaga akurasi peluru saat bot bergerak dengan cepat.

SurvivorBot dipilih sebagai bot terbaik karena strategi yang dimilikinya paling seimbang dalam segi penyerangan dan pertahanan. Dengan gerakan yang cepat dan sulit diprediksi, bot ini mampu menghindari banyak serangan serta tetap mempertahankan akurasi tembakan. Selain itu, penyesuaian energi berdasarkan jarak dan kondisi pertempuran membuatnya lebih efisien dibanding bot lainnya. SurvivorBot berhasil mengimplementasikan strategi greedy dengan nilai solusi paling optimal, memaksimalkan peluang bertahan hidup serta damage output tanpa mengorbankan terlalu banyak energi.

Secara keseluruhan, SurvivorBot memberikan performa terbaik karena mampu mengadaptasi strategi penyerangan dan pertahanan secara dinamis, menjadikannya bot paling efektif dalam berbagai skenario pertempuran.

5.2 Saran

Algoritma greedy yang diterapkan pada bot dapat ditingkatkan agar lebih optimal dalam berbagai skenario pertempuran. SmartGreedyBot dan BotMuter sebaiknya memiliki strategi yang lebih fleksibel agar tidak terpaku pada keputusan jangka pendek yang kurang menguntungkan. BerserkerBot dapat memperbaiki efisiensi energi dengan menyesuaikan strategi penembakan, sehingga tetap agresif tanpa boros energi.

SurvivorBot, meskipun sudah menunjukkan performa terbaik masih bisa dikembangkan dengan mempertimbangkan aspek prediksi gerakan lawan agar keputusan greedy yang diambil lebih akurat. Selain itu, mengombinasikan greedy dengan pendekatan lain, seperti *heuristic-based decision making* (mengombinasikan pendekatan greedy dengan heuristik yang lebih kompleks atau lebih adaptif), dapat meningkatkan keseimbangan antara serangan dan pertahanan tanpa kehilangan efektivitas keputusan cepat.

Lampiran

Github Repository: https://github.com/fliegenhaan/Tubes1_tembak2

Video Penjelasan: https://youtu.be/MX5GjThlAic

No	Poin	Ya	Tidak
1	Bot dapat dijalankan pada Engine yang sudah dimodifikasi asisten.	✓	
2	Membuat 4 solusi greedy dengan heuristic yang berbeda.	✓	
3	Membuat laporan sesuai dengan spesifikasi.	✓	
4	Membuat video bonus dan diunggah pada Youtube.	✓	

Daftar Pustaka

Alabi, T . (2023). What is a Greedy Algorithm? Examples of Greedy Algorithms.

https://www.freecodecamp.org/news/greedy-algorithms/

Amazon Web Services. (2024). What is an API? https://aws.amazon.com/what-is/api/

Microsoft Corporation. (2024a). .NET documentation. https://dotnet.microsoft.com/en-us/

Microsoft Corporation. (2024b). C# documentation.

https://learn.microsoft.com/en-us/dotnet/csharp/

Microsoft Corporation. (2024c). Learn to code C#.
https://dotnet.microsoft.com/en-us/learntocode

Munir, R. (2025a). Algoritma greedy (Bagian 1). Bahan kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika ITB.
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf

Munir, R. (2025b). Algoritma greedy (Bagian 2). Bahan kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika ITB.
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf

Munir, R. (2025c). Algoritma greedy (Bagian 3). Bahan kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika ITB.
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf

Robocode Development Team. (2024a). Robocode Tank Royale. GitHub.

https://github.com/robocode-dev/tank-royale

Robocode Development Team. (2024b). Robocode Tank Royale documentation.

https://robocode-dev.github.io/tank-royale/