Giovanni Orlotti – 2025

Company V3.0

A competitive Crobots project - manual

1. Panoramica

Company è un robot strategico progettato per adattarsi dinamicamente alle situazioni di combattimento, alternando tra fasi offensive e difensive in base alle condizioni del campo di battaglia. A differenza di molte strategie statiche, che si basano su comportamenti predefiniti e ripetitivi, Company utilizza un approccio più flessibile, reagendo agli eventi in tempo reale e ottimizzando il proprio posizionamento, la scansione dei nemici e la gestione del fuoco in modo da massimizzare l'efficacia e la sopravvivenza.

Le sue fondamenta si basano sul celebre robot *Goblin*, ma anziché limitarne le capacità, Company ne rivoluziona ogni aspetto. Le funzioni originali vengono ripensate e adattate per ottenere un comportamento più intelligente ed efficiente. Se *Goblin* era incentrato su una strategia aggressiva con attacchi continui e movimenti rapidi, Company integra una logica più raffinata, bilanciando momenti di aggressione pura con momenti di difesa strategica.

Uno degli aspetti chiave di Company è la sua capacità di muoversi sul campo di battaglia in modo calcolato. Il robot non si limita a seguire un percorso fisso, ma analizza costantemente la propria posizione e quella dei nemici per trovare il punto ottimale da cui attaccare o difendersi. Questo è reso possibile da un avanzato sistema di scansione che gli permette di identificare la presenza di avversari nelle vicinanze e valutare la necessità di un attacco immediato o di una manovra evasiva.

Il sistema di combattimento di Company è altrettanto sofisticato. A differenza di molti robot che si affidano a una strategia di fuoco costante e indiscriminato, Company ottimizza ogni sparo per massimizzare le probabilità di colpire il bersaglio e ridurre lo spreco di munizioni. Grazie a una gestione intelligente dell'angolazione del cannone e della distanza del bersaglio, il robot è in grado di aggiustare il tiro e correggere eventuali errori di mira, aumentando così l'efficienza dell'attacco.

Inoltre, Company non è solo un cecchino preciso, ma anche un combattente resiliente. Quando subisce danni significativi, il robot attiva una serie di contromisure difensive per ridurre l'esposizione al fuoco nemico. Questo può includere movimenti evasivi, cambi di direzione improvvisi e una maggiore attenzione alla scansione dell'ambiente per evitare ulteriori colpi. Il robot, quindi, non si limita a subire passivamente gli attacchi, ma cerca attivamente di mitigare i danni e riposizionarsi per ottenere un vantaggio tattico.

---

2. Spiegazione delle Variabili

All'inizio del codice vengono dichiarate diverse variabili intere:

```
int b, rng, orng, tt, deg, odeg, dir, t, q, dam, reg;
```

- `b` - Determina il quadrante in cui si trova il robot.
- `rng` - Distanza di un nemico rilevato.
- `orng` - Un'altra variabile di distanza, probabilmente per memorizzare risultati di scansione precedenti.
- `tt` - Flag di controllo (probabilmente indica lo stato del ciclo principale).
- `deg` - Angolo di scansione del cannone.
- `odeg` - Angolo precedente usato per sparare.
- `dir` - Direzione del movimento del robot.
- `t` - Contatore per varie condizioni.
- `q` - Identificatore del quadrante.
- `dam` - Memorizza il danno ricevuto.
- `reg` - Usato per la scansione.

---

3. main() - Punto di Ingresso

```
main()
{
    if (loc_x()<500) {sx(70);
    if ((!scan(90,10))&&(!scan(70,10))) up(930); else dw(70);}
    else {dx(930);
    if ((!scan(90,10))&&(!scan(110,10))) up(930); else dw(70);}
```

- Il robot verifica se si trova nella metà sinistra o destra del campo di battaglia.
- Se è a sinistra (`loc_x() < 500`), si sposta a sinistra (`sx(70)`) e poi decide se muoversi su o giù in base ai risultati della scansione.
- Se è a destra (`loc_x() >= 500`), si sposta a destra (`dx(930)`) e decide se muoversi su o giù.

Ciclo Principale

```
deg = 3600;
tt = 1;
while(1)
{
    dam = damage();
    while ((!orng || orng > 450) && (damage() < dam + 5))
    {
        if (t > 12)
            if (Radar()) End();
        dam = damage();
        Fire(1);

        if (UpDown())
        {
            Angle();
            if (UpDown()) Move();
        }
    }
```

- `deg = 3600;` inizializza la direzione di scansione del cannone.
- `dam = damage();` memorizza il livello di danno attuale.
- Se non ci sono bersagli ravvicinati (`orng > 450`) e non si subisce molto danno (`damage() < dam + 5`):
    - Spara (`Fire(1)`).
    - Se necessario un movimento verticale (`UpDown()`), regola l'angolo e si sposta.

---

4. Sistema di Movimento

Il robot si muove in modo da rimanere nel campo di battaglia ed evitare di rimanere bloccato.

Navigazione Basata sul Quadrante

```
b = (loc_x() > 500) + 2 * (loc_y() > 500);
```

- Determina il quadrante del robot:
    - **0** → Alto-sinistra
    - **1** → Alto-destra
    - **2** → Basso-sinistra
    - **3** → Basso-destra

Ogni quadrante ha la propria logica di movimento.

Funzioni di Movimento

```
up(l) {dir = 90; while (loc_y() < l) { drive(90,100); Fire(); } drive(270,0);}
dw(l) {dir = 270; while (loc_y() > l) { drive(270,100); Fire(); } drive(90,0);}
dx(l) {dir = 0; while (loc_x() < l) { drive(0,100); Fire(); } drive(180,0);}
sx(l) {dir = 180; while (loc_x() > l) { drive(180,100); Fire(); } drive(0,0);}
```

- Muove il robot in quattro direzioni (su, giù, destra, sinistra).
- Chiama `Fire()` mentre si muove, suggerendo che il robot continua a sparare.

---

5. Rilevamento del Nemico ed Evasione

Scansione Radar

```
Radar()
{
    reg = 10; t = 0;
    while((reg += 20) != 730) if (scan(reg,10)) if (++t > 2) {t = 0; return 0;}
    return 1;
}
```

- Scansiona in incrementi di 20 gradi fino a 730 gradi.
- Se più di due scansioni rilevano nemici, ritorna 0 (nemico rilevato).
- Altrimenti, ritorna 1 (nessun nemico rilevato).

Evasione del Fuoco

```
End(){
    tt = 0;
    dw(550); up(450);
    while(1) {
        dx(850); sx(150);
    }
}
```

- End() viene attivato se il robot determina di essere in pericolo.
- Si muove su e giù tra due posizioni indefinitamente.

---

6. Sistema di Attacco

Uso del Cannone

```
Fire(flag)
{
    if (orng = scan(deg,10))
    {
        if (!scan(deg -= 5,5)) deg += 10;
        if (tt) if (orng > 770)
        {
            if (!scan(deg -= 3,3)) deg += 6;
            cannon(deg, orng);
            ++t;
            deg += 40;
            return;
        }
```

- Scansiona il nemico alla direzione deg.
- Se lo trova, aggiusta leggermente la mira prima di sparare.

---

7. Riassunto del Comportamento

1. Si posiziona strategicamente nel campo di battaglia.
2. Scansiona continuamente alla ricerca di nemici.
3. Spara e aggiusta la mira per migliorare la precisione.
4. Si muove per evitare di essere colpito troppo facilmente.
5. Se danneggiato gravemente, attiva una strategia evasiva.