ToyTankWorld Protocol (TTWP) - Specifica tecnica

1. Introduzione

Questo documento descrive le caratteristiche del protocollo TTWP, come utilizzato nel colloquio tra i programmi *client* ed il programma *server* del gioco ToyTankWorld. Il protocollo viene impiegato per lo scambio di messaggi in codifica ASCII un canale TCP/IP.

Si assume infatti che ogni programma *client* abbia una connessione TCP/IP diretta e privata con il programma *server*. La porta di ascolto (di default) del *server* è 2001. Si assume comunque che l'identità di tale porta sia fissa e nota a priori al momento della connessione.

1. Il server - Il campo di gioco

Il campo di gioco è un'arena di forma quadrata in cui sono presenti alcuni ostacoli fissi. La dimensione dell'arena è di 512×512 unità. Il numero, la dimensione e la posizione degli ostacoli non sono stabilite a priori, al fine di evitare che i programmi *client* siano progettati in modo troppo specifico.

I carri armati giocattolo sono posizionati in modo casuale all'inizio di ciascuna *round*, con la garanzia che la posizione sia 'realistica' (i.e. senza sovrapposizioni fisicamente impossibili tra oggetti, pareti dell'arena e carri).

Il programma *server* gestisce l'arena, gli ostacoli e l'insieme dei turni di gioco per ciascun carro. Ciascun programma *client* gestisce un singolo carro. All'inizio di ogni *round*, a ciascun carro viene attribuito una valore iniziale di *energia*. Ogni azione del carro (e.g. movimento, sparo) comporta il consumo di energia. Colpire carri avversari fa guadagnare energia, mentre essere colpiti ne fa perdere. Ogni carro viene rimosso dall'arena quando l'energia residua scende a zero. Lo scopo del gioco è quello di far durare l'energia del carro il più a lungo possibile: il vincitore di ogni *round* è il carro che viene rimosso per ultimo.

Un *torneo* è composto da una sequenza di *round*. Vince il torneo il carro che complessivamente è rimasto in gioco più a lungo.

2. Il carro

Il carro può essere immaginato come un rettangolo con lati indicati come fronte, retro, lato sinistro e lato destro, dotato di una torretta girevole.

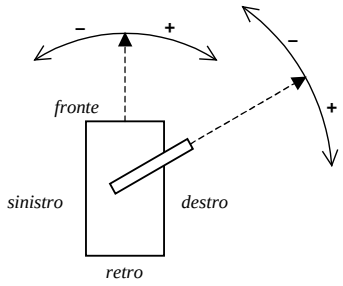

Figura 1 - Lati e convenzioni per gli angoli

In ciascun istante, il carro è contraddistinto dai seguenti parametri:

1) posizione esatta nell'arena (in unità di misura);
2) angolo orientamento del carro (in radianti);
3) angolo orientamento della torretta rispetto al carro (in radianti);
4) energia rimanente (in punti).

La posizione esatta e l'angolo di orientamento del carro rispetto al sistema di coordinate dell'arena sono note solo al *server* e non vengono mai trasmesse al carro, cioè al programma *client* che lo gestisce. Viceversa, vengono continuamente trasmesse dal *server* ad ogni programma *client* l'angolo di orientamento della torretta rispetto al carro e l'energia residua (vedi anche la sezione 5).

Il carro può effettuare le seguenti azioni, comandate dal *client*:

1) movimento lineare;
2) rotazione del fronte del carro (in radianti, relativamente all'orientamento corrente);
3) rotazione della torretta (in radianti, relativamente all'orientamento corrente rispetto al fronte del carro);
4) sparo.

Le azioni di movimento e sparo prevedono un consumo di energia, mentre le azioni di rotazione del fronte del carro e della torretta hanno costo nullo. I parametri di default per l'energia sono:

- **energia iniziale**: 1600 punti;
- **costo di un turno**: 1 punto;
- **costo di un colpo**: 1 punto;
- **guadagno per colpo a segno**: 8 punti;
- **perdita per colpo subito**: 8 punti.

Si assume che il motore del carro sia sempre acceso, pertanto ogni turno ha un costo fisso (vedi sopra) indipendentemente dal fatto che il carro si muova o meno. Al contrario, un colpo sparato costa un punto di energia. La collisione con oggetti, pareti ed altri carri non provoca danni al carro.

Relativamente alle velocità, i valori di default sono i seguenti:

- **velocità massima del carro**: 2 unità/turno;
- **velocità massima di rotazione del fronte**: 2°/turno;
- **velocità massima di rotazione della torretta**: 4°/turno;
- **velocità di tiro**: 1 colpo/turno.
- **velocità dei colpi**: 16 unità/turno.

Si deve tenere presente che i valori di default per movimenti e velocità potranno essere modificati per ragioni di bilanciamento. Si assume comunque che essi siano prefissati e noti prima dell'inizio del gioco.

3. Sensori

Si assume che ogni carro possieda un sensore di collisione e due sensori visivi, uno piazzato sul corpo del carro ed uno posizionato sulla torretta.

Il sensore di collisione riporta una valore 0/1 (dove 1 significa collisione) per ognuno dei quadranti riportati in figura:

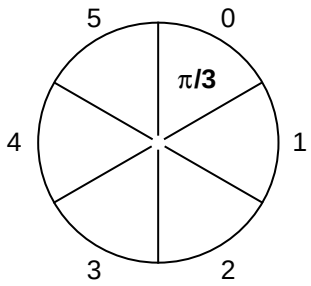

Figura 2 - Sensore di collisione

Il sensore di collisione non è in grado di distinguere il tipo di oggetto con cui è avvenuta la collisione. Si possono avere collisioni in più punti. Il sensore di collisione non rileva i colpi (nel senso di colpi sparati da altri carri) eventualmente ricevuti.

Il primo sensore visivo piazzato sul corpo del carro consente una visione di corto raggio sull'intero arco di 2π radianti per una profondità di 8 unità, misurate dal centro del carro. La lettura del sensore è rappresentata da un vettore di 'pixel' organizzato come in figura:

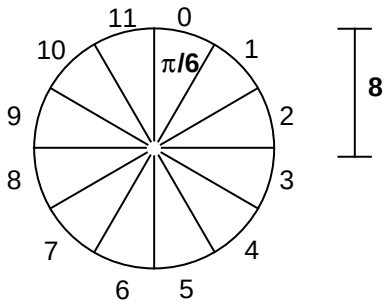

Figura 3 - Visione di corto raggio

L'orientamento della visione del sensore è relativo all'orientamento del *fronte* del carro: si assume che l'asse compreso tra il 'pixel' 0 ed il 'pixel'11 coincida con l'asse del carro.

Il sensore visivo è in grado di distinguere quattro 'colori':

0 nessuna lettura;

1 presenza di ostacolo fisso o parete;

2 presenza di carro avversario;

3 presenza di ostacolo fisso o parete e di carro avversario.

Il sensore visivo, ovviamente, non è in grado di vedere attraverso gli oggetti (ostacoli, pareti, carri). Pertanto il valore 0 (nessuna lettura) può significare sia 'campo libero' che 'fuori dalla linea di vista'.

Il sensore visivo posizionato sulla torretta consente una visione direzionale di lungo raggio, con un'apertura angolare di $2/3\pi$. Il centro dell'angolo di visuale coincide con l'orientamento della torretta. Il sensore riporta una mappa di 6×6 'pixel' organizzati come in figura:

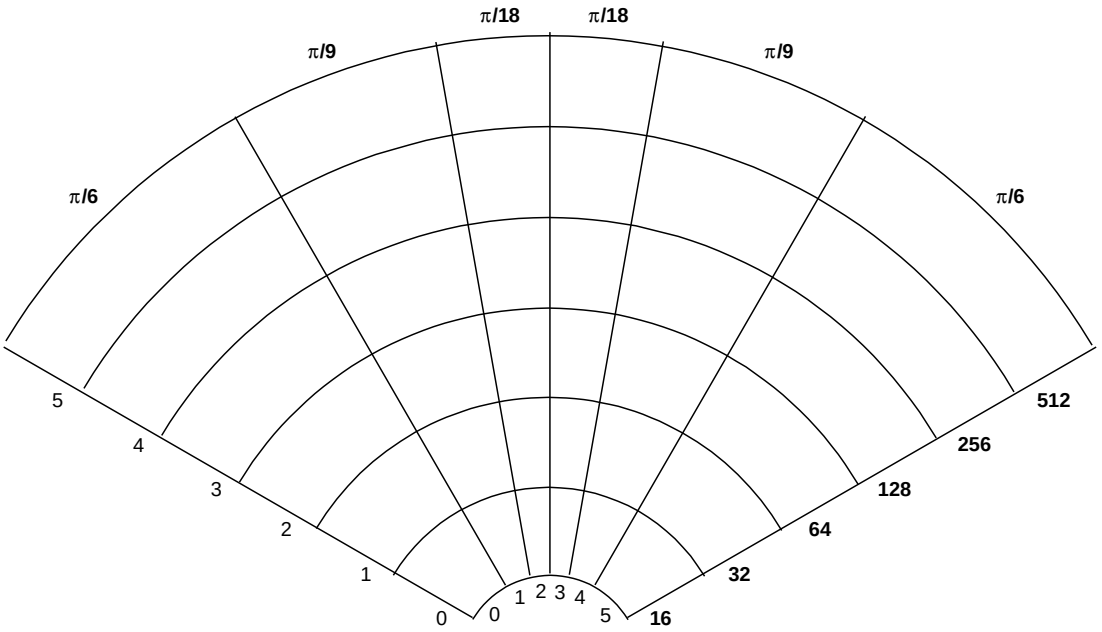

Figura 4 - Visione di lungo raggio

E‘ importante osservare che la figura non rappresenta correttamente i valori di profondità, che crescono esponenzialmente (i.e. 16, 32, 64, 128, 256, 512 unità) e non linearmente. Si noti anche che la risoluzione angolare del sensore aumenta in corrispondenza all’asse della torretta e diminuisce agli estremi.

4. Procedura

Il server gestisce i turni di gioco. Il procedimento adottato dal *server*, in sintesi, è il seguente:

1) il *server* si pone in attesa di connessione dei programmi *client* (con una soglia di tempo massimo);
2) viene sorteggiata la sequenza di gioco tra tutti i *client* connessi;
3) viene effettuato il posizionamento automatico dei carri corrispondenti a ciascun *client*;
4) gestione iterativa dei turni: il turno viene assegnato a ciascun carro in base alla sequenza stabilita al passo 2;
5) ad ogni turno viene interpellato il programma *client* che gestisce il carro: il *server* trasmette un insieme di informazioni (vedi sezione successiva); il *client* risponde trasmettendo le istruzioni per il carro (vedi sezione successiva).
6) le istruzioni trasmesse da ciascun *client* vengono eseguite all’atto della ricezione;
7) il *server* verifica continuamente che ogni carro abbia una energia superiore a zero; ogni carro che abbia esaurito l’energia viene rimosso dal campo di gioco;
8) il *round* ha termine quando tutti i carri sono stati rimossi dal campo di gioco;

9) qualora sia previsto un nuovo *round*, il *server* invia un messaggio di *restart* (vedi sezione successiva) a tutti i *client*, quindi attende un breve periodo di tempo e ripete la procedura a partire dal punto 2. All'inizio di ogni *round*, l'energia dei carri viene riportata al valore nominale.

5. Protocollo

Il protocollo TTWP prevede tre stati:

– ***initial***, quando il server è in attesa di connessioni;

– ***in_play***, durante lo svolgimento del gioco;

– ***wait***, dopo un *restart*, prima dell'effettuazione di un nuovo *round*.

In ogni stato, *server* e *client* si scambiano dei *messaggi* secondo un turno prestabilito e definito dal protocollo. Ogni messaggio tra *client* e *server* è codificato come una stringa di caratteri ASCII terminata da un CR. I campi del messaggio sono suddivisi da spazi.

Ogni errore di gestione del protocollo, se identificato dal *server*, comporta la chiusura immediata della connessione e la rimozione del carro corrispondente dal campo di gioco.

5.1 Stato *initial*

Il *server* è in attesa sulla porta prestabilita. Il *client* stabilisce la connessione. Non appena ciò avviene (i.e. la connessione è stabilita ed accettata dal *server*), il *server* invia il seguente messaggio:

ToyTankWorldProtocol (TTWP) - Vers. <*n*>**.**<*n*>**: READY**<cr>

Dove <*n*>**.**<*n*> è la versione del protocollo. Il messaggio ha un puro valore indicativo ed il client non deve in alcun modo basarsi sul contenuto del medesimo.

Il *client* deve quindi inviare un messaggio di identificazione:

DESCRIPTION <*name*> <*r*> <*g*> <*b*><cr>

dove <*name*> è una stringa ASCII senza spazi, lunga al massimo 32 caratteri, che identifica il *client*. <*r*> <*g*> e <*b*> sono tre numeri interi compresi tra 0 e 255 descrivono il colore di visualizzazione (su schermo) del carro. I valori **0 0 0** (nero) e **255 255 255** (bianco) sono vietati, per ragioni di visibilità e di distinguibilità.Per gli stessi motivi, i tre valori devono avere una somma compresa tra 127 e 512.

Il *server* risponde accettando la trasmissione tramite il messaggio:

200 OK<cr>

o respingendola, tramite il messaggio:

500 ERROR<cr>

La risposta del *server* termina lo stato ***initial***. Se la risposta del *server* è positiva, il *client* viene correttamente registrato come controllore di un carro. La risposta negativa è invece seguita dalla chiusura del collegamento da parte del *server*.

5.2 Stato *in_play*

Durante ciascun turno, è il *server* a interpellare il *client* inviando un messaggio contenente i dati istantanei del carro:

`SENSOR` <*energy*> <*aim*> <*collision*> <*shortRange*> <*longRange*>`<cr>`

Il contenuto dei campi è il seguente:

- <*energy*> è un numero intero che indica l'energia residua;
- <*aim*> è un numero in virgola fissa[1], con segno, che indica l'orientamento (in radianti) della torretta relativamente all'asse del carro;
- <*collision*> è una stringa di 6 'pixel' che indicano la lettura del sensore di collisione, ciascun 'pixel' è un numero intero compreso tra **`0`** e **`1`**;
- <*shortRange*> è una stringa di 12 'pixel' che indicano la lettura del sensore visivo di corto raggio, ciascun 'pixel' è un numero intero compreso tra **`0`** e **`3`**;
- <*longRange*> è una stringa di 36 'pixel' che indicano la lettura del sensore visivo di lungo raggio, ciascun 'pixel' è un numero intero compreso tra **`0`** e **`3`**; l'immagine è trasmessa per righe ed in ordine crescente di profondità (vedi figura 3).

Il client risponde con un messaggio che descrive l'azione da intraprendere:

`ACTION` <*turn*> <*move*> <*aim*> <*fire*>`<cr>`

Il contenuto dei campi è il seguente:

- <*turn*> è un numero in virgola fissa, con segno, che indica la rotazione in radianti del fronte del carro rispetto all'orientamento corrente;
- <*move*> è un numero in virgola fissa, con segno, che indica la velocità lineare del carro impostata, espressa in unità/turno;
- <*aim*> è un numero in virgola fissa, con segno, che indica la rotazione relativa in radianti della torretta a partire dall'orientamento corrente rispetto al fronte del carro;
- <*fire*> è un numero intero che indica l'intenzione di sparare, con valori ammessi **`0`** e **`1`**.

Per ai campi <*turn*>, <*move*> e <*aim*>, sono permessi valori superiori al massimo consentito. Tali valori vengono comunque interpretati dal *server* come coincidenti con il valore massimo consentito.

Ovviamente, lo scambio **`SENSOR`** / **`ACTION`** viene iterato più volte, fino a quando il carro non viene rimosso. Lo stato ***in_play*** ha termine quando il carro viene rimosso dal gioco. In tal caso il *server* invia il messaggio:

---

[1] In TTWP, i numeri in virgola fissa sono codificati in questo modo:

[<*sign*>]<*integralPart*>**`.`**<*decimalPart*>

Dato che i valori trasmessi sono comunque piccoli, si assume che <*integralPart*> sia composto da una sola cifra. <*decimalPart*> può contenere da una a otto cifre, ma non può essere vuoto. Quindi, l'intero 1 si indica come **`1.0`** e lo 0 come **`0.0`**.

STOP<cr>

Qualora non vi sia un nuovo *round*, il server invia il messaggio:

CLOSE <*finalGap*><cr>

e chiude la connessione. Il valore <*finalGap*> è un numero intero che indica il comportamento globale del carro nel torneo, inteso come sequenza di *round*. Il valore <*finalGap*> deve essere interpretato come una penalità: ad un carro che vince tutti i *round* viene attribuito un valore 0; viceversa, per ogni *round* perduto, il carro subisce un punteggio di penalità pari alla durata residua del *round*, espressa in numero di turni, a cui il carro non ha potuto partecipare avendo terminato l'energia.

5.3 Stato *wait*

Al termine dello stato ***in_play***, il server può effettuare un nuovo *round*. In tal caso, il server invia il messaggio:

RESTART<cr>

Da questo momento, il protocollo si trova in uno stato ***wait*** che termina quando il server invia il primo messaggio **SENSOR** relativo al nuovo *round*.