Informatica Industriale
Progetto di un'applicazione real time
basata sul kernel S.Ha.R.K

S.Ha.R.K SOCCER SIMULATOR 2001

Autori

Merli Andrea
Zucchetti Alessandro

1. Descrizione generale

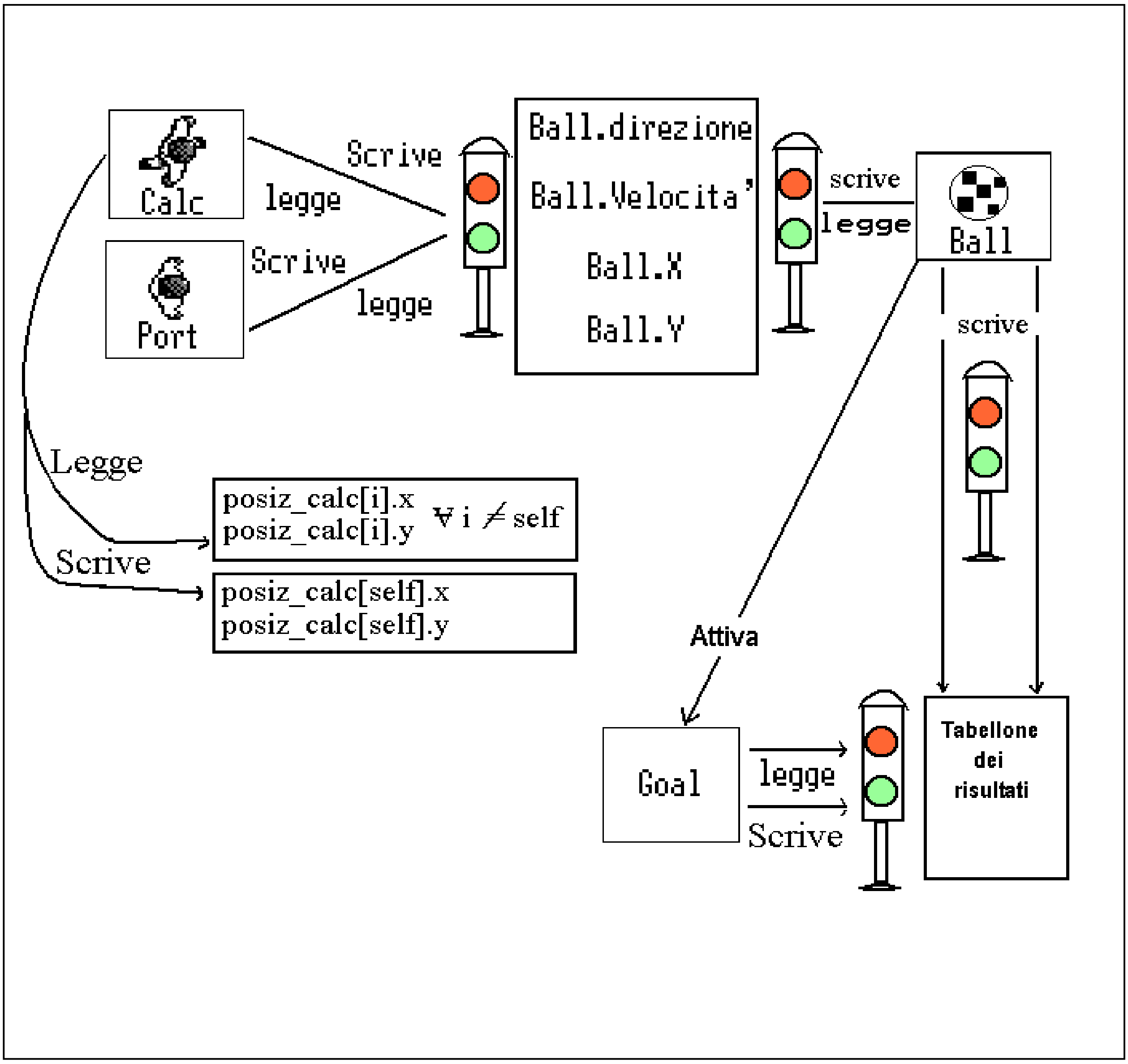
Scopo del progetto è simulare una partita di calcio a cinque realizzando giocatori e palla e simulandone il comportamento in diverse situazioni.

2. Strumenti

L'applicazione è stata realizzata basandosi sul kernel S.Ha.R.K, ed è scritta in linguaggio C.Il kernel fornisce le primitive per la creazione e la gestione dei task ed i meccanismi di comunicazione

3. Realizzazione

3.1 Struttura dell'applicazione



3.2 Descrizione struttura

La struttura del progetto consiste in vari tipi di task che realizzano le diverse parti presenti nel simulatore. C'è così un task per ciascun giocatore, un task per la palla, uno per ciascun portiere ed uno per l'aggiornamento del tabellone in caso di goal.
I giocatori, la palla ed i portieri sono realizzati come task hard mentre il processo deputato all'aggiornamento del tabellone è un task soft.
Per quanto riguarda la schedulazione dei task hard, gli algoritmi di scheduling scelti per realizzare il simulatore sono EDF; un server CBS soddisfa invece i task soft, così da garantire che la banda a disposizione per i task periodici HARD sia sempre sufficiente.

Riepilogando

10 task uno per calciatore τc_i
1 task per la palla τb
2 task uno per ciascun portiere τp_i
1 task per il tabellone At

TASK	PERIODO	TEMPO DI CALCOLO
τc_i	100.000	5.000
τb	30.000	5.000
τp_i	30.000	4.000
At	1.200.000	1.000

3.3 Garanzia

Per ciò che riguarda la condizione di schedulabilità per i task periodici essa è abbondantemente soddisfatta

$$\sum_{i=1}^{n} \frac{Ci}{Ti} = \frac{5000}{100000} + \frac{5000}{30000} + \frac{4000}{30000} = 0{,}35$$

La banda residua è assegnata automaticamente ai task aperiodici mediante CBS

$$U_s = 1 - U_p = 1 - 0,35 = 0,65$$

La scelta dei periodi è dettata dalla necessità di eseguire i task ad una frequenza tale che l'applicazione, oltre a funzionare correttamente, sia "realistica" nel senso che i tempi di gioco non siano troppo brevi.

3.4 Modi Di Funzionamento dell'Applicazione

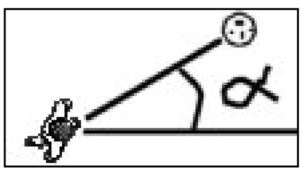
NO_BALL_MODE: Nessun calciatore in possesso della palla
PASS_MODE: La palla è calciata e si muove
PENALTY_MODE_WHITE: Rigore per la squadra Bianca
PENALTY_MODE_BLUE: Rigore per la squadra Blu

4 Processi Attivi nell'Applicazione

4.1 Main

Nel main si apre il sistema grafico e si creano i CAB deputati alla comunicazione tra i processi; si disegnano quindi lo stadio ed il tabellone e si entra in un loop in attesa della pressione di un tasto.

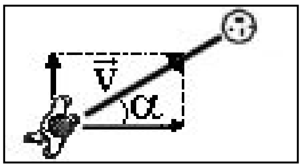
4.2 Task Calciatore



All'avvio questo processo, basandosi sul PID, ricava la posizione iniziale nel campo da gioco; si dispongono allora i calciatori nella zona di campo che a ciascuno di loro compete. Ci sono due difensori, un centrocampista e due attaccanti per ciascuna squadra.

Quando l’applicazione si trova in NO_BALL_MODE ogni task valuta l’angolo tra la sua posizione e quella della palla, verifica se quest’ultima si trova nella sua zona ed in tal caso la seleziona come obiettivo del calciatore. Se invece il pallone è in una zona diversa del campo il punto verso cui il calciatore si dirige è la posizione iniziale.
Lo spostamento del calciatore viene scomposto in due vettori, uno verticale ed uno orizzontale, in modo tale che la velocità del calciatore non dipenda dalla sua direzione di corsa.

dy = -VEL*sin(α);
dx = VEL*cos(α);



Il calciatore poi esegue una verifica per scoprire se sta per entrare in collisione con un altro giocatore o con i bordi del campo. In questo caso la sua posizione viene modificata per far sì che le immagini non risultino sovrapposte.
Quando un calciatore raggiunge la palla, calcola direzione e velocità del tiro quindi setta le variabili angolo, velocità e mode (=PASS_MODE), il tutto utilizzando un CAB.
La direzione di tiro è calcolata diversamente in base al ruolo del calciatore. Un difensore valuterà la posizione del centrocampista e tirerà in sua direzione. Un centrocampista invece sceglierà come obiettivo per il suo tiro la posizione di un attaccante, mentre quest’ultimo cercherà il goal tirando verso la porta.
Il funzionamento dell’applicazione in PENALTY_MODE è molto simile. La principale differenza consiste nel fatto che un unico calciatore per squadra (il centrocampista) ha la possibilità di andare al tiro. Gli altri invece si collocano in posizione d’attesa.

4.3 Task Ball

Questo task ha il compito di gestire la palla.
Esso legge la velocità e la direzione impresse da un giocatore e istante per istante calcola la nuova posizione e la nuova velocità della palla e la disegna nella posizione corretta.
Per calcolare la posizione abbiamo assunto che la palla si muova con un moto piano uniformemente decelerato, in particolare abbiamo utilizzato le equazioni:

- $V = Vo - a * T$
- $X = V * T - 0,5 * a * T * T$

Per semplificare abbiamo considerato una forza d’attrito proporzionale al peso della palla, cioè una decelerazione costante pari al coefficiente d’attrito per l’accelerazione di gravità, e un tempo base Tau di durata uno con cui aggiornare le due equazioni sopra riportate.
In questo modo istante per istante abbiamo implementato i seguenti passi:

1. $V2 = V1 - a * Tau$
2. If $(V2 > 0)$ $X = V1 * Tau - 0,5 * a * Tau * Tau$
3. $V1 = V2$

E in base alla direzione teta fornita dal calciatore, abbiamo calcolato le componenti Dx e Dy dello spostamento X:

1. $Dx = X * \cos(teta)$
2. $Dy = X * \mathrm{sen}(teta)$

Abbiamo, inoltre, deciso che la palla rimbalzasse sui bordi del campo.
Una volta calcolate le nuove coordinate, la palla, che è una semplice immagine salvata nel file Image.h, viene disegnata nella posizione desiderata, attraverso l’istruzione grx_putimage() contenuta nella funzione Draw_ball().
Un altro compito di questo task è riconoscere i gol e quindi aggiornare il risultato e inizializzare un contatore interno (la variabile flag) in base al quale si attiva il task “Gol” che gestisce le immagini da rappresentare sul tabellone centrale.
Per gestire la comunicazione tra tutti questi tipi di task vengono utilizzati quattro CAB:

1. Un CAB “modo” per gestire il modo in cui si trova la palla (*PASS_MODE* quando viene calciata, *NO_BALL_MODE* quando non è calciata, *PENALTY_MODE* quando si deve tirare un rigore)
2. Un CAB “direzione” per leggere la direzione della palla
3. Un CAB “velocita” per leggere la velocità della palla
4. Un CAB “azione” per gestire le azioni che devono essere compiute dal task Goal

Mentre il primo CAB viene letto e scritto dal task Ball, il secondo e il terzo CAB sono solo letti, il quarto solo scritto.
Occorre precisare che per poter utilizzare un CAB per trasmettere la direzione della palla convertita in valori char abbiamo considerato un angolo giro composto di 120 gradi applicando a tutti gli angoli da trasmettere la proporzione:

$$360 : 120 = \theta : (\theta \text{ in Char})$$

4.4 Task Portiere

Questo task, come il nome stesso dice, gestisce le azioni che i portieri devono compiere.
Le coordinate del portiere sono calcolate, attraverso le funzioni xportiere() e yportiere(), in base alla posizione della palla: quando questa si trova nella metà campo avversaria il portiere torna nella posizione di attesa e cioè in mezzo alla porta, quando è nella stessa metà campo, ma fuori dall'area di rigore, il portiere si muove solo lungo l'asse verticale e infine, quando la palla arriva in area, il portiere esce verso di essa e se la raggiunge la rinvia immediatamente.
Le funzioni xportiere() e yportiere() restituiscono le nuove posizioni in cui devono essere ridisegnati i portieri dopo aver calcolato, in base alle coordinate obiettivo della palla, gli incrementi da assegnare alle coordinate degli stessi portieri determinate al ciclo precedente.
Gli incrementi assegnati non sono costanti, ma sono calcolati in base alla differenza tra i valori in ingresso alla funzione, in particolare non si ha movimento lungo un asse quando le corrispondenti coordinate del portiere e della palla coincidono.
Abbiamo inoltre deciso che il portiere, una volta giunto ai limiti dell'area in cui può uscire, mantiene la coordinata limite acquisita finché la palla non torna entro una zona del campo oltre il limite precedentemente superato; inoltre, in caso di rigore, il portiere rimane fermo nella sua posizione di attesa fino a quando il calciatore non colpisce la palla.
Una volta calcolate le nuove coordinate si cancella il portiere nella vecchia posizione coprendolo con un rettangolo di colore dello sfondo e lo si

disegna nella posizione attuale chiamando la funzione Draw_port() che ha anche il compito di ricostruire l'intera area di rigore.
Questo task comunica attraverso i CAB "modo", "velocita" e "direzione" con il task Ball a cui trasmette velocità, direzione e modo della palla quando riesce a rinviarla.
Un'importante considerazione deve essere effettuata sul periodo di questo task che abbiamo posto uguale a quello del task Ball, per evitare che l'esecuzione del task portiere, troppo ritardato rispetto alla palla, causasse il fatto di non riuscire ad evitare neanche un gol.

4.5 Task Gol

Questo task è aperiodico, viene attivato ogni volta che si verifica un gol e ha il compito, in base all'azione richiesta dal task Ball, di rappresentare sul tabellone la grafica "GOAL !!" o il menu iniziale.
Come sopra accennato, il task gol viene attivato dal task ball all'interno della funzione goal_behaviour() che, a seconda del valore in ingresso, scrive in un CAB l'azione che deve essere compiuta sul tabellone.
Il task Gol chiama la funzione draw_goal() che legge il contenuto del CAB: se il valore letto è uguale a GOAL_ACT si procede, attraverso delle chiamate ricorsive dell'istruzione grx_plot(), alla rappresentazione sul tabellone della scritta "GOAL !!" che è un'immagine contenuta nel file stadio.h, mentre se il valore è MENU_ACT si procede alla riconfigurazione iniziale del tabellone cancellandovi qualsiasi immagine è riportandovi il menu iniziale di scelta.
Per evitare, in occasione di un gol, comportamenti strani dei giocatori e della palla si assegna, all'interno della funzione Draw_goal() e tramite il CAB opportuno, il modo NO_BALL_MODE alla palla che inoltre non viene fatta muovere dal centro del campo finché non è stato aggiornato sul tabellone il risultato della partita.
Il task gol comunica quindi con tutti i task calciatori e con il task Ball attraverso il CAB "modo", mentre è collegato solo al task Ball tramite il CAB "azione".

5 Organizzazione dei file

L'applicazione è composta dai seguenti file:

a. initfile.c
b. iniziali.h
c. soccer.c
d. calc.h, calc2.h
e. images.h
f. stadio.h
g. position.h

5.a *initfile.c*

Si occupa della registrazione dei servizi del kernel (scheduler, porte, driver) e richiama il *main* contenuto in *soccer.c*

5.b *iniziali.h*

Contiene tutte le definizioni di costanti utilizzate dal programma e le variabili di uso globale.

5.c *soccer.c*

In questo file è presente la funzione *main* che si occupa di inizializzare la tastiera, il video, i cab per i messaggi e inizializzare i task.

5.d *calc.h,calc2.h*

Contengono gli header che rappresentano le immagini dei calciatori.

5.e *images.h*

Contiene gli header che rappresentano le immagini della palla.

5.f *stadio.h*

Contiene gli header che rappresentano le immagini dello stadio e del tabellone.

5.g *position.h*

Contiene la definizione della posizione iniziale dei calciatori.

6 Interfaccia e comandi utente

All'avvio si presenta l'immagine di uno stadio. I comandi disponibili sono riassunti su di un tabellone.

Spazio: Chiama tre funzioni che creano i TASK dei calciatori, della palla e dei portieri.(Da il via alla partita)
W o *E*: Settano il modo in PENALTY_MODE_WHITE o PENALTY_MODE_BLUE (In pratica assegnano un rigore ad una delle squadre)
K: Uccide tutti i TASK e reinizzializza l'applicazione
ESC: Esce dall'applicazione.