Giulio Urlini Matr. 019618 Corso di Robotica

Rapporto sulle modifiche al progetto Robot Swarm

1.Modifiche apportate al programma Robot Swarm

Il codice del programma Robot Swarm è stato modificato nelle parti di gestione dei file (caricamento e salvataggio), nell'aspetto e nel funzionamento dei dialoghi di LOAD e SAVE dei file di tipo .world, concernenti l'ambiente simulato.

Le modifiche sono state eseguite nei seguenti passaggi:

- Identificazione del ciclo principale di esecuzione, che è attivo quando il programma è in attesa di comandi da parte dell'utente
- Nel ciclo principale sono presenti due blocchi che verificano la richiesta da parte dell'utente di un'operazione di caricamento o di salvataggio (effettuati con il premere il pulsante corrispondente)
- Modifica del codice vera e propria, solo per quanto riguarda il codice di salvataggio e caricamento, senza toccare il ciclo principale o la gestione dell'ambiente nel suo complesso

Il codice di salvataggio e caricamento era diviso tra alcune istruzioni presenti nel ciclo principale (in *sim.c*), la gestione della finestra di dialogo invece era affidata ad alcune funzioni accessorie (di gestione dell'ambiente grafico e di lettura degli eventi da esso prodotti, come per esempio gli eventi di mouse clicked, key pressed, ed altri). Tali funzioni erano contenute nel file *graphics.c*.

La parte di codice nel ciclo principale forniva le coordinate per il display dei componenti il dialogo modale i LOAD/SAVE, la lettura dei file presenti nella directory predefinita (che era solo una, e non poteva essere modificata). A questo punto i metodi di *graphics.c* gestivano la finestra modale, interpretando le azioni compiute dall'utente tramite mouse e tastiera. La pressione dei pulsanti SAVE/LOAD provocava la chiusura del dialogo modale (tale chiusura non era effettuata dal ciclo di esecuzione principale, ma dal metodo di gestione che in quel momento aveva il controllo completo dell'esecuzione del programma, come è proprio di un dialogo modale).

I problemi da risolvere nel comportamento di tali dialoghi erano i seguenti:

- L'attivazione e disattivazione del dialogo da parte dello stesso pulsante era un comportamento non intuitivo, e assolutamente nono standard
- Il dialogo, anche se grafico, poteva essere gestito solo dalla tastiera , tranne che per la chiusura del dialogo stesso (i file venivano elencati nella finestra, ma non potevano essere selezionati col mouse, bisognava digitarne il nome completo)
- La directory di lavoro era unica, e i file presenti in essa non potevano essere più di 15.

Le modifiche apportate sono state le seguenti:

- Nella parte di programma principale in cui vengono riconosciuti i comandi principali per quanto riguarda i comandi di LOAD e SAVE sono stati spostati in funzioni a parte i blocchi di codice che realizzavano il disegno della finestra: dato che tale disegno era molto simile per i due dialoghi, la funzione suddetta è una sola, a cui è stato aggiunto un parametro per distinguere i due tipi di dialogo.
- Come secondo passaggio, è stata modificata la funzione che veniva chiamata all'attivazione del dialogo : ReadText1(); in essa ho aggiunto un certo numero di parti per riconoscere più azioni compiute all'interno della finestra, cioè i pulsanti di OK e CANCEL, i pulsanti di scorrimento per la lista dei file, la conseguente possibilità di visualizzare più di 15 file (nella versione originale se la directory dei file di tipo WORLD avesse contenuto più di 15 file, sarebbe stato generato un errore).

- Come corollario al punto precedente, sono state aggiunte delle funzioni per la navigazione attraverso l'albero delle directory. In questo caso si fa riferimento ad un altro file sorgente .c per l'implementazione di tali funzioni.

L'idea guida nelle modifiche è quella do costruire delle funzioni particolari per i singoli dialoghi, e per le diverse azioni possibili, in maniera più possibile modulare, in modo da poter gestire facilmente eventuali modifiche al blocco principale del programma.

2.Linee guida per la ridefinizione del suddetto programma

La prima cosa da fare per modificare sostanzialmente il programma è quella di isolare concettualmente le funzioni che tale programma svolge, e inserirle poi nel codice in funzioni distinte (questo riguarda anche le funzioni grafiche e non di dialogo con l'utente). Tali operazioni possono essere il risultato di una sequenza di azioni semplici: anche in questo caso bisogna separare logicamente tali funzionalità dal codice nel suo complesso, e riscriverle in modo da poterle considerare dall'esterno come atomiche. Il problema principale infatti del programma così come è adesso è la difficoltà di capire intuitivamente la corretta sequenza di esecuzione di determinate operazioni: la mancanza di dialoghi modali, o comunque di effetti grafici ben visibili dello stato corrente del sistema, rendono difficile il riconoscimento della giusta sequenza di azioni, e il passo corrente di tale sequenza. Le uniche informazioni che vengono fornite sono date da una stringa di caratteri nell'area in basso a destra della finestra di programma, e sono assolutamente insufficienti. Un altro problema legato sempre allo stato corrente del sistema è che in molti casi, se il programma è in un certo stato, molte delle funzioni sono bloccate (per esempio in fase di inserimento di oggetti non si possono inserire dei dati diversi, etc.). La linea guida della ristrutturazione del programma dovrebbe isolare i diversi moduli, come già detto, e rendere visibile la condizione di blocco delle funzioni (aggiungendo per esempio dei pulsanti che vengano ombreggiati quando la funzione relativa è disabilitata).

Inoltre sarebbe il caso di prevedere un'area unica per l'inserimento dei dati da parte dell'utente, che riceva di volta in volta i dati necessari, senza dover costringere l'utente a cercare l'area di inserimento, a seconda della operazione che si sta svolgendo.

3.Esempi caratteristici per effettuare tali modifiche

Viene presentato adesso in breve il codice modificato, evidenziando le parti modificate, in modo da fornire una linea guida di modifica.

Codice originale:

```
case LOAD_WORLD:
  if (sim==1)
  {
    system ("ls ./WORLD > worlds");
    for (i=0; i<10000; i++) ;
    ffdir = fopen ("worlds","r");
    if (!ffdir) {exit(-1); }
    else {
      buff = (char **) malloc (sizeof(char*) * 30);
      if (!buff) { }
      else {
        cont = 0;
        while (!feof(ffdir) && (cont<30)) {
          buff[cont] = (char *) malloc (sizeof (char) * 15);
          if (!buff[cont])
            exit(-1);
```

```
      else {
        fscanf(ffdir," %s \n",buff[cont]);
        cont++;
       }
     }
   }
   fclose(ffdir);
}


DrawPlate(300,130,400,20,BLACK,PLATE_UP);
Color(WHITE);
DrawText(420,145,"ROBOT SWARM - LOAD");
DrawPlate(300,150,400,330,GREY_69,PLATE_UP);
DrawPlate(320,175,360,30,GREY,PLATE_DOWN);
Color(BLACK);
DrawText(322,171,"enter a file name and press return:");
DrawText(322,225,"avaliable files...");
DrawText(325,195,"../WORLD/");
 DrawText(536,240,"HINT:");
 DrawText(536,260,"to abort loading");
 DrawText(536,275,"operations, unpress");
 DrawText(536,290,"the LOAD button in");
 DrawText(536,305,"the window below.");
 DrawPlate(320,230,200,230,GREY,PLATE_DOWN);

 if (buff) {
    Color(BLACK);
    for (i=0; i<cont; i++) {
     if (buff[i]) {
       DrawText (325,245+i*15,buff[i]);
          free(buff[i]);
     } else { exit(-1); }
   }
   free (buff);
 } else { exit(-1); };

 strcpy(name,ReadText1(context,FILE_NAME_TEXT,button));
 if (name[0]!='\0')
 {
  sprintf(text,"loading %s.world",name);
  DisplayComment(context,text);
  strcpy(text,name);
  strcat(name,".world");
  strcpy(file_name,"WORLD/");
  strcat(file_name,name);
  ffile = fopen(file_name,"r");
  if (ffile)
   {
    WaitCursor();
    CreateEmptyWorld(context);
```

```
        ReadWorldFromFile(world,ffile);
        strcpy(world->Name,text);
        fclose(ffile);
        DrawWorld(context);
        if (!(robot->State & REAL_ROBOT_FLAG)) InitSensors(context);
        DrawRobotIRSensors(robot);
        sprintf(text,"%s loaded",name);
        PointerCursor();
       }
      else sprintf(text,"unable to find %s in WORLD directory",name);
      DisplayComment(context,text);
     }
     else DisplayComment(context,"nothing done");
     DrawWorldSquare(world);
   }
   break;
```

Nuovo codice:

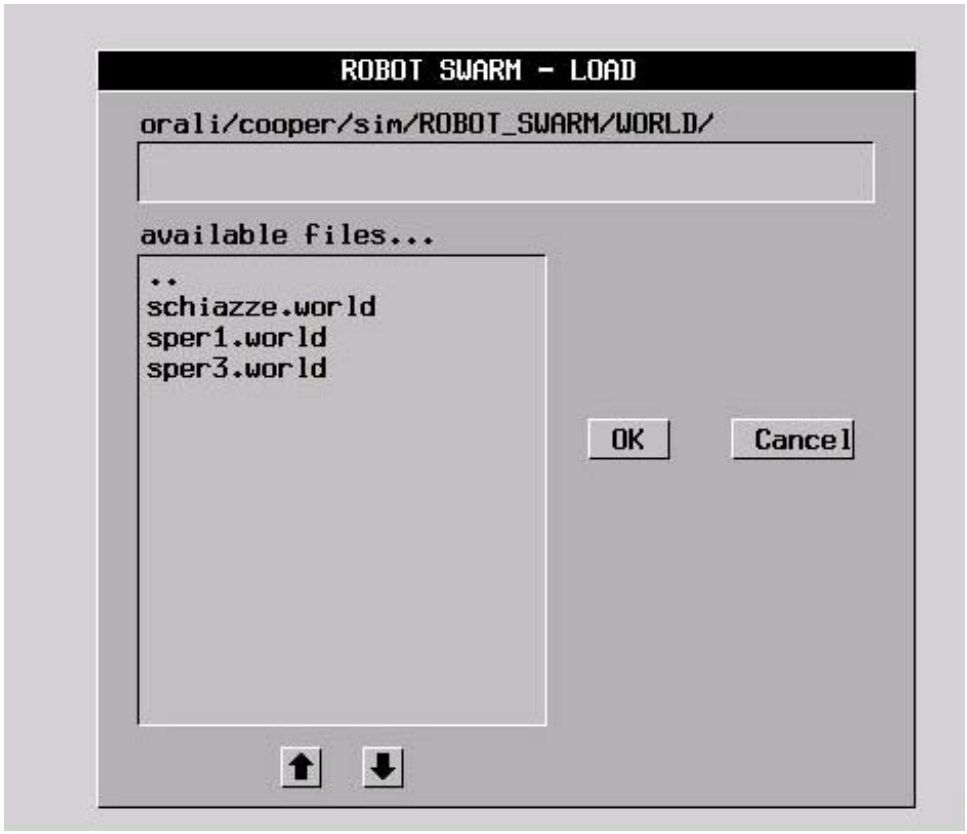
```
case LOAD_WORLD:
  if (sim==1)
   {
        strcpy(name,LoadDialog(DIALOG_LOAD));
            if (!exit_comm) strcpy(default_directory,pres_dir);
                strcpy(temp_string,pres_dir);
                 if (temp_string[myLen(temp_string)-1] != '/'){
                        myCat(temp_string,"/");
                 }
                strcat(temp_string,name);
                temp_file = fopen(temp_string,"r");
                if (!temp_file) {
                        sprintf(text,"File %s not found",name);
                        DisplayComment(context,text);
                        DrawWorldSquare(world);
                        break;
                }
                close(temp_file);
                if ((strcmp(SubStr(name,strlen(name)
                6,6),".world"))&&(name[0]!='\0')) {
                        sprintf(text,"Invalid format for %s file",name);
                        DisplayComment(context,text);
                        DrawWorldSquare(world);
                        break;
                }
                if (name[0]!='\0')  {
                  sprintf(text,"loading %s",name);
                  DisplayComment(context,text);
                  strcpy(text,name);
```

```
                        strcpy(file_name,pres_dir);
                        if (file_name[strlen(file_name)-1] != '/'){
                                strcat(file_name,"/");
                        }

                        strcat(file_name,name);
                        printf("%s \n",file_name);
                        ffile = fopen(file_name,"r");
                        if (ffile)   {
                           WaitCursor();
                           CreateEmptyWorld(context);
                           ReadWorldFromFile(world,ffile);
                           strcpy(world->Name,text);
                           fclose(ffile);
                           DrawWorld(context);
                           if (!(robot->State & REAL_ROBOT_FLAG)) InitSensors(context);
                           DrawRobotIRSensors(robot);
                           sprintf(text,"%s loaded",name);
                           PointerCursor();
                        }
                        else sprintf(text,"unable to find %s in current directory",name);
                        DisplayComment(context,text);
                }
                else DisplayComment(context,"nothing done");
                DrawWorldSquare(world);
        }
        break;
```

Vediamo ora le diverse modifiche:
per prima cosa, in questa porzione di codice, come in quella riguardante il salvataggio dei dati (*case SAVE*) tutta la parte di visualizzazione del dialogo è nel programma principale (in grassetto nel codice originale). Tale dialogo può essere considerato comune ai due casi LOAD e SAVE, a parte alcune differenze, che possono essere passate come parametri ad una funzione apposita, che realizzi tale dialogo. La mia scelta è stata di specificare un solo parametro (DIALOG_LOAD o DIALOG_SAVE) e lasciare alla funzione LoadDialog il compito di distinguere, in base al parametro suddetto, i casi di LOAD e SAVE. In questo modo in un secondo tempo è sarà possibile aggiungere degli altri casi, che ricadano in questo insieme di dialoghi, senza eccessive modifiche.
Un altro problema è quello di avere delle coordinate assolute, specificate esplicitamente nel codice. In questo caso la mia scelta è stata quella di istanziare delle costanti, in modo da rendere più agevoli le modifiche. In questo modo è stato molto più semplice inserire nuovi oggetti al dialogo, come le frecce a scorrimento, e i pulsanti di OK e CANCEL (cfr. figura a pagina seguente).
Ogni modifica del dialogo inoltre non doveva essere riportata in diversi punti del codice, ma solo nella funzione LoadDialog.
Tale dialogo, con l'aggiunta dei pulsanti suddetti, e con alcune modifiche alla funzionalità, ha assunto un aspetto e un comportamento più facilmente comprensibile da un utente generico. Infatti per esempio, nella funzione originale, il dialogo veniva chiuso completando correttamente l'operazione, o premendo di nuovo un tasto LOAD (o SAVE) in un'altra parte del programma. Nel nuovo dialogo i pulsanti di OK e CANCEL fanno parte integrante del dialogo stesso, comportandosi quindi come lo standard dei programmi esistenti.

Questo discorso segue il principio, già citato in precedenza, di utilizzare il più possibile blocchetti di codice in altrettante funzioni, in modo da dover usare diversi moduli per svolgere insiemi di azioni, senza dover utilizzare un blocco di codice monolitico, di difficile lettura da chiunque altro se non la persona che lo ha scritto.
Nella mia versione modificata, la chiarezza non è ancora totale. Quello che è necessario adesso è isolare tutte le funzioni principali, utilizzando anche una nomenclatura delle funzioni coerente, e in seguito costruire l'ambiente contenitore, in modo da poter vedere bene la sequenza delle diverse funzionalità, in modo da renderle maggiormente fruibili dall'utente.



Nella versione precedente mancavano le frecce, che servono per lo scorrimento della lista dei file, che nella versione potevano essere un massimo di 15 solo nella directory ./WORLD/ del programma. Discorso analogo per OK e Cancel.