Progetto relativo al corso di Grafica Computerizzata
Franza Giovanni

Scopi

Lo scopo primario del progetto è quello di realizzare una significativa esperienza di programmazione utilizzando openGL e cosí dimostrare il livello di conoscenza maturato.
Uno scopo secondario, personale, è l'esplorazione della possibilità di realizzare interfacce utente per apparecchiature radioamatoriali.

Criteri di scelta

Lo scopo primario del progetto prevede che vengano utilizzate una serie di funzionalità, ovvero:

1. una scena composta da piú di un oggetto,
2. almeno un oggetto modellato gerarchicamente,
3. almeno un oggetto realizzato con l'ausilio di una libreria quali ad esempio libglu o libglut,
4. l'uso della display list,
5. l'uso di luci (almeno di due tipi), materiali, vettori normali e tessiture,
6. la possibilità di muovere l'osservatore nella scena
7. il caricamento di modelli 3D (.obj, 3ds, ...) da files esterni
8. la generazione di modelli usando un software per la grafica 3D (ad esempio: blender3d, wings3d, ...)
9. l'animazione con controllo real time o attraverso canali di comunicazione
10. funzionalità avanzate non trattate nel corso.

Lo scopo secondario prevede l'accesso agli oggetti "toccati" dal puntatore in modo da cambiarne le proprietà (ed è compatibile con il punto 10) in modo che risulti "fluido" e naturale per l'utente.

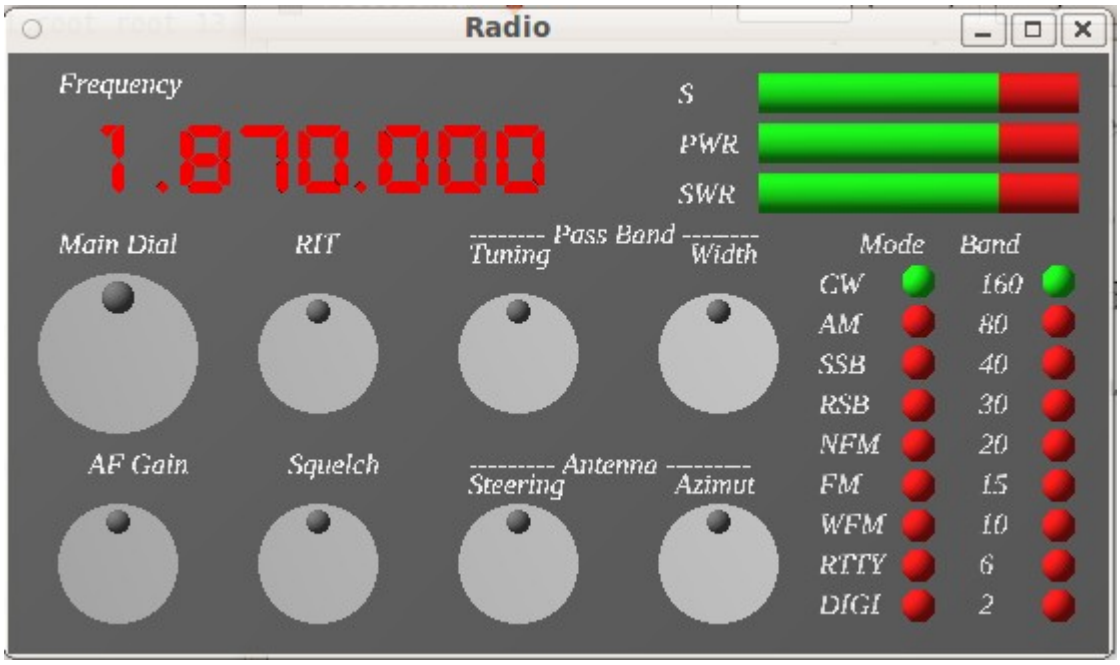
Le funzionalità realizzate sono:

1. La scena, anzi, le scene, sono composte da decine di oggetti.
2. Proprio per renderne possibile la realizzazione, la gran parte gli oggetti è stata costruita gerarchicamente a partire da una semplice funzione che realizza un tubo.
3. Un paio di oggetti sono stati costruiti usando la funzione di disegno delle sfere messa a disposizione da libglut.
4. Non ho fatto uso di display list.
5. Ognuna delle due scene ha delle luci, le texture (di tipo "random") sono state usate in una delle due scene.
6. In una delle due scene l'osservatore puó essere mosso attorno all'oggetto, nell'altra è l'oggetto a ruotare.
7. Non vengono caricati modelli 3d ma, usando una libreria aggiuntva (libftgl) vengono carcati dei font true type per realizzare delle scritte.
8. Il modello utilizzato, reperito in rete, è stato costruito con un prodotto di modellazione.
9. L'animazione è gestibile sia da input tramite mouse/tastiera che tramite tick temporali. (rotazioni programmate) – L'uso del joystick si è dimostrato impossibile per problemi implementativi relativi a libglut.
10. A questo proposito ho gestito due finestre e, in una, ho usato il render mode come GL_SELECT per individuare gli oggetti "toccati dal puntatore".

Descrizione del progetto

Si tratta del prototipo di un “controllo di stazione radio” composto da funzioni semplici di controllo di un ricetrasmettitore con relativa antenna, che viene visualizzata in una differente finestra.

Il “controllo di stazione radio”, da qui in poi RTX, prevede le funzioni base relative alla selezione di banda, alla sintonia, al filtro di media frequenza, alla selezione della modulazione utilizzata, al volume, allo squelch nonchè l'orientamento dell'antenna. A completamento delle funzionalità è previsto un indicatore di sintonia e misuratori di segnale, potenza, ed onda riflessa dall'antenna.



I comandi rotativi sono ruotabili da tastiera usando

Tasto	Effetto
u	Ad ogni tick aumenta il valore di uno e ruota il controllo in senso orario di un grado
U	Ad ogni tick aumenta il valore di 18 e ruota il controllo in senso orario di 18 gradi
d	Ad ogni tick diminuisce il valore di uno e ruota il controllo in senso antiorario di un grado
D	Ad ogni tick diminuisce il valore di 18 e ruota il controllo in senso antiorario di 18 gradi
T oppure t	Inverte il movimento
S oppure s	Ferma il movimento
0	Azzera il valore e riporta il controllo in posizione centrale
Rotella centrale del mouse	Aumenta / diminuisce il valore del controllo e lo fa ruotare di conseguenza. Se mossa in contemporanea con premuto SHIFT o CTRL o ALT l'aumento / diminuzione passa da 1 a 5 / 25 / 125 rispettivamente

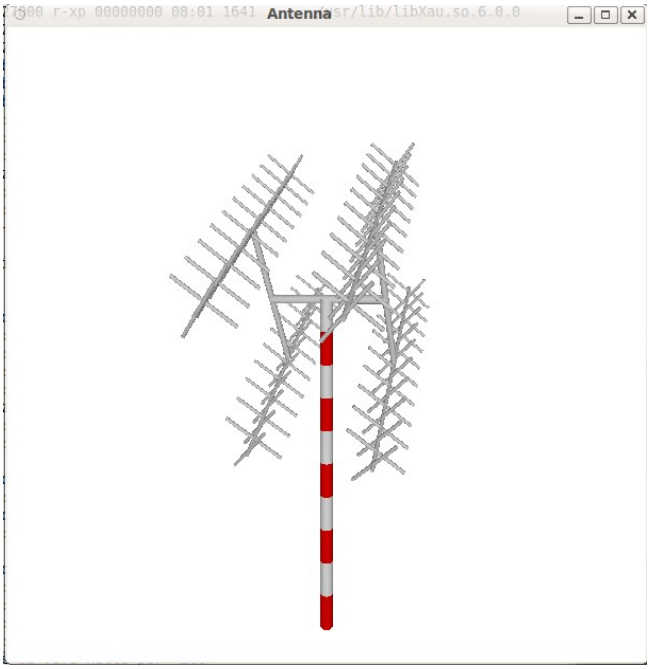
Per quanto riguarda gli interruttori il loro funzionamento risponde alle seguenti regle:

Tasto	Effetto
u	Attiva l'interruttore
U	Attiva l'interruttore
d	Disattiva l'interruttore
D	Disattiva l'interruttore
Rotella centrale del mouse	Attiva / Disattiva l'interruttore

Gli interruttori realizzati fanno parte di due gruppi (controllo banda e controllo modi di emissione) per cui uno solo per gruppo puó essere attivo: attivare uno significa contemporaneamento disattivare gli altri.

La frequenza visualizzata è determinata dalla banda scelta, dalla manopola di sintonia e dal RIT (receiver incremental tuning), i valori mostrati dagli strumenti sono ricavati dai due controlli di filtro banda e dal controllo di volume (ad ovvio scpo dimostrativo).
I quattro tasti freccia permettono di ruotare il punto di osservazione attorno al RTX in modo da osservarlo in prospettiva (è attivata la proiezione assonometrica).

A completamento dell'esercizio viene visualizzata un array di 4 antenne a polarizzazione incrociata ognuna ottenuta tramite l'accoppiamento di 2 antenne Yagi-Uda con polarizzazioni perpendicolari tra di loro.



Le antenne vengono mosse dal controllo presente sul RTX. Con i tasti 'a' e 'p' si possono vedere le due antenne in proiezione assonometrica oppure prospettica.

Note di realizzazione

Il progetto è l'unione di due progetti piú semplici, uno relativo al solo RTX e l'altro alla sola antenna. Questa derivazione spiega il motivo per cui vi sono due set completi di callback relative alle due finestre. La piú importante funzione di timer, presente sul RTX (ovvero agganciata alla relativa finestra), quando necessario scatena il refresh dell'altra finestra usando il frammento di codice seguente

```
glutSetWindow( antenna );
glutPostRedisplay();
glutSetWindow( radio );
```

La costruzione dell'antenna è gerarchica: la costruizione complessiva si ottiene con la funzione quad() che usa la funzione tube() per disegnare i tubi di collegamento e la funzione turn() per disegnare ognuna delle quattro antenne. La funzione turn() a sua volta richiama due volte la funzione array() che è responsabile della creazione della singola antenna. Ed alfine la funzione array() usa la funzione tube() per disegnare gli elementi cilindrici.

Anche la costruzione delle manopole è gerarchica, essendo effettuata con la funzione handle() che richiama la funzione tube(), e lo stesso succede per gli interruttori e per i misuratori.

Una cosa da far notare è come alle funzioni handle(), button(), meter(), etichetta() e sintonia() venga passato il render mode in modo da disegnare tutto solo se il render mode è pari a GL_RENDER. Il motivo di questa scelta è duplice: in primis per una questione di velocità, in secundis per rendere non selezionabili gli oggetti non interessanti (ad esempio le partei della scatola, gli strumenti, le etichette...). Alcuni oggetto non vengono disegnati, altri (come le manopole) vengono disegnati con maggior semplicità e non completamente (in questo contesto il termine “disegnati” tende a generare confusione in quanto il render mode a GL_SELECT non disegna alcunchè ma verifica solo se un determinato oggetto sarebbe disegnato in un intorno del cursore).

Per movimentare il disegno i pomelli delle manopole e gli indicatori relativi ai pulsanti sono stati disegnati utilizzando la funzione glutSolidSphere() messa a disposizione da libglut.

La luce relativa al RTX viene messa da avanti/sinistra/alto, il RTX puó essere visualizzato sia guardandolo da “davanti” (visualizzazione iniziale) sia ruotandovi attorno (si utilizzano i tasti freccia) per esaminare il suo aspetto da altre posizioni. Per ció che riguarda l'antenna la luce viene dalla posizione dell'osservatore.

Nel disegnare le pareti normalmente “non in vista” del RTX è stata usata un texture. La particolarità é che questa texture è realizzata facendo ricorso a dei numeri random e che è stata applicata “ruotata di 45 gradi” in modo da non allineare le ricorrenze (dovute alle ripetizioni della texture) con i contorni delle superfici.

Le scritte sono state disegnate utilizzando la libreria libftgl che permette di utilizzare dei font True Type per realizzare delle scritte in rilievo. Data la particolarità del lavoro le scritte non sono state fatte particolarmente “in rilievo”, dovendo simulare quanto appare sul pannello di una radio. I font usati sono stati reperiti con una rapida ricerca su Internet (e non sono tra i migliori disponibili).

In un primo tempo si era pensato di utilizzare i dati provenienti da un RTX reale (IC7000 ICOM) ma il lavoro di impementazione del relativo protocollo non è stato sufficentemente rapido per cui vi si è rinunciato, limitandosi a permettere la rotazione cdei controlli (e delle antenne) sulla base dei comandi dati con la tastiera.

Altra rinuncia è stata quella della gestione del joystick, in quanto impossibile. Le funzioni da utilizzare sono banali (glutJoystickFunc()) ma all'attivazione della relativa callback tutto ció che si ottiene è un esplosione del programma. Nessuna ricerca in rete ha dato risultati differenti da “non implementato su Linux” e/o “usa una libreria differente”. Disarmante il fatto che la funzione glutJoystickFunc() sia regolarmente documentata e che vi sano esempi di codice, nessuno dei quali funzionante.

Positivo il fatto che si riescano a gestire piú finestre in modo molto semplice senza dover fare particolari equilibrismi, semplicemente ricordandosi di postare le corrette richieste di ridisegno alle finestre che debbono eseguirlo. In soldoni: quando il timer “muove” il controllo di posizione dell'antenna non si puó limitare a richiedere il redraw del controllo ma deve anche chiederlo per la finestra dov'è renderizzata l'antenna stessa.

Non banalissima la gestione del cursore, se si vuole conoscere su quali oggetti è posizionato. Il tutto si basa sulla possibilità di “ridisegnare” la scena solo per capire quali oggetti sono posizionati sotto il cursore. Lo si fa usando una matrice di proiezione che individua una zona nei pressi del cursore e poi ridisegnando il tutto con il render mode a GL_SELECT. Nel farlo openGL accoderà l'indicatore degli oggetti realmente disegnati in un buffer da cui potranno essere letti.
Occorre quindi creare un buffer per il nome degli oggetti (nome che, in realtà, è un numero intero), impostare la matrice di proiezione, il render mode, eseguire il ridisegno degli oggetti ed infine leggere i dati.

```
// mi prendo le dimensioni della viewport
glGetIntegerv (GL_VIEWPORT, viewport);
// buffer per i dati selezionati
glSelectBuffer ( XtNumber( selectBuf ), selectBuf );
// mi metto in modo selezione
(void) glRenderMode (GL_SELECT);
glInitNames();
glPushName(0);
// uso la matrice di proiezione
glMatrixMode( GL_PROJECTION );
glPushMatrix();
glLoadIdentity();
// creo un areola nei dintorni del mio mouse
gluPickMatrix ((GLdouble) x , (GLdouble) viewport[3] - y  , 5, 5, viewport);
// ridisegno la scena in modo selezione in modo da poter
// catturare gli id degli oggetti  ridisegnati nell'areola
glOrtho ( ... );
// okkio che se non mi metto in modelview le rotazioni vanno a pallini
glMatrixMode( GL_MODELVIEW );
glPushMatrix();
// il parametro viene passato per semplificare il disegno
drawScene( GL_SELECT );
// ripristino modi e matrici
glPopMatrix();
glMatrixMode( GL_PROJECTION );
glPopMatrix();
glutSwapBuffers();
// leggo i dati degli oggetti ridisegnati nell'areola di cui sopra
hits = glRenderMode( GL_RENDER );
// richiedo il ridisegno della scena
glutPostRedisplay();
printf( "%d %d\n", hits, selectBuf[3] );
```

Normalmente le buffer vi sono ben piú di un indicatore, dato che sotto il puntatore potrebbero essere alineati piú oggetti (anche se noi ne vediamo solo il piú vicino). Nel nostro caso peró questa situazione capita solamente in casi estremi (ad esempio quando il RTX è ruotato molto e qundi le manopole si sovrappongono) e quindi non vogliamo gestirla.

Compilazione e librerie utilizzate

Il Makefile è piuttosto semplice:

```
LIBS=-lglut -lGL -lGLU -lftgl

all: esercizio2

.c : $*.c
     cc -o $* $*.c ${LIBS} -I/usr/include/freetype2
```

Le librerie utilizzate sono libGL, libGLU, libglut e libftgl.
Questa ultima fa parte del pacchetto libftgl2 (per compilare serve anche il pacchetto libftgl2-dev)

Il programma è stato sviluppato su un sistema Ubuntu 10.04LTS quindi dovrebbe essere installabile sulle macchine SUPSI.

Conclusioni

Ovviamente questo è un lavoro incompleto: il numero di manopole utilizzate è largamente inferiore a quanto necessario per la gestione di un apparato professionale. Purtuttavia dall'esperienza escono delle indicazioni interessanti che indicano come la realizzazione di un'interfaccia in openGL non sia un mero saggio di abilità fine a se stessa ma possa realmente trovare impiego in una realizzazione funzionante.
Il lato interessante è la buona reattività dell'interfaccia e l'occupazione della CPU rimane attorno al 50% anche in condizione di ridisegno globale (come si puó vedere dal display sottoriportato). In condizioni operative normali (senza manopole in rotazione continua, senza display antenne) il carico macchina è decisamente ridotto.