*University of Bologna - Department of Mathematics*

*Piazza di Porta S.Donato, 5 - 40127 - Bologna*

# Free Form Deformation: *xcsurf* (Version 3.0) plugin

G. Casciola E. Trevisan

Department of Mathematics
University of Bologna

*Bologna 2005*

**Sommario**

This report describes the *FFD* plugin for *xcsurf*. It is designed for modelling by Free Form Deformation techniques proposed in literature and making use of high 3D interaction.

G. Casciola E. Trevisan
Department of Mathematics, University of Bologna, P.zza di Porta S.Donato 5, Bologna, Italy. E-mail: casciola@dm.unibo.it.

# Indice

**Indice** **i**

**Introduzione** **1**

**1 Il sistema *xcsurf*** **3**
1.1 I problemi dell'interazione utente . . . . . 5

**2 Soluzioni in *xcsurf*** **11**
2.1 Località dei comandi e contatto con la superficie . . . . . 11
2.2 Riferimenti 3D ed esattezza della variazione . . . . . 16
2.3 Intuitività dell'azione . . . . . 24
2.4 Pressioni del mouse e pilotaggio a distanza . . . . . 27

**3 *FFD* User's Guide** **29**
3.1 Modifica del poliedro di controllo . . . . . 29
3.2 Modifica della superficie . . . . . 34
3.3 Modellazione per deformazione . . . . . 35
3.4 Guida ad una deformazione ENFFD . . . . . 35
3.5 Guida ad una deformazione Wire . . . . . 38

**Bibliografia** **43**

# Introduzione

In questa nota si vogliono mettere a disposizione del progettista *xcsurf* tutte le informazioni necessarie per muoversi nel nuovo ambiente di modellazione per deformazione incluso in questa implementazione.
Gli elementi teorici relativi ai metodi di modellazione per deformazione, sono stati oggetto della tesi di Laurea di E. Trevisan [Trevisan00]; nel rapporto [ffdsurvey00] vengono descritti i metodi presi in considerazione nel lavoro sopra citato ed in particolare i metodi implementati in questo plugin.
Di particolare importanza è il modello di interfaccia proposto che si ritiene indispensabile per supportare amichevolmente l'interazione dell'utente con la superficie che si vuole deformare.
Si illustrerà, di fatto, il nuovo ambiente di *xcsurf* per la modifica interattiva di una superficie NURBS, che è stato anche l'ambiente in cui è stata effettuata la sperimentazione sulle tecniche di deformazione stesse.

Capitolo 1

# Il sistema *xcsurf*

*xcsurf* è il modellatore 3D di *xcmodel*. Esso permette di:

1. generare e modificare curve NURBS 3D
2. generare e modificare superfici NURBS
3. generare e modificare liste di superfici
4. generare e modificare superfici gerarchiche mediante NURBS trimmate
5. visualizzare superfici singole o sottoforma di oggetto complesso, cioè ottenuto come lista di superfici semplici o trimmate.

*xcsurf* è scritto interamente in linguaggio ANSI C, e per le chiamate grafiche fa esclusivo uso delle funzioni della libreria Xlib di X Window. Per questo motivo, esso è portabile su piattaforme Unix/Linux.
L'interfaccia utente, cioè le finestre di dialogo, è invece realizzata tramite la libreria *xtools*, creata nell'ambito del progetto *xcmodel*, la quale funge da insieme di macroistruzioni Xlib.

La figura 1.1 mostra una delle possibili configurazioni iniziali di *xcsurf*.

A differenza degli altri pacchetti di *xcmodel*, *xcsurf* non è strutturato in un'unica finestra. Le window Control Net e Curve/Surface, così come le window Persp, Front, Side e Up che in figura risultano di dimensioni ridotte, sono indipendenti tra loro e possono essere riposizionate e/o ridimensionate.

In *xcsurf*, una superficie può essere **caricata** da file o da una banca di primitive integrate nel sistema, quali cubi, sfere, cilindri, ecc.. Nel primo caso tutti i dati che la definiscono vengono letti da un file associato che ha estensione *.db*. In tale archivio, vengono inseriti in formato ASCII tutti i parametri che definiscono una superficie NURBS. La **generazione** di una superficie può avvenire in due modi: utilizzando un editor di griglia dei punti di controllo, o attraverso tecniche avanzate. Una volta creata, la superficie viene conservata in memoria e visivamente presente nel textbox degli oggetti, in modo che sia disponibile per tutta la sessione di lavoro.

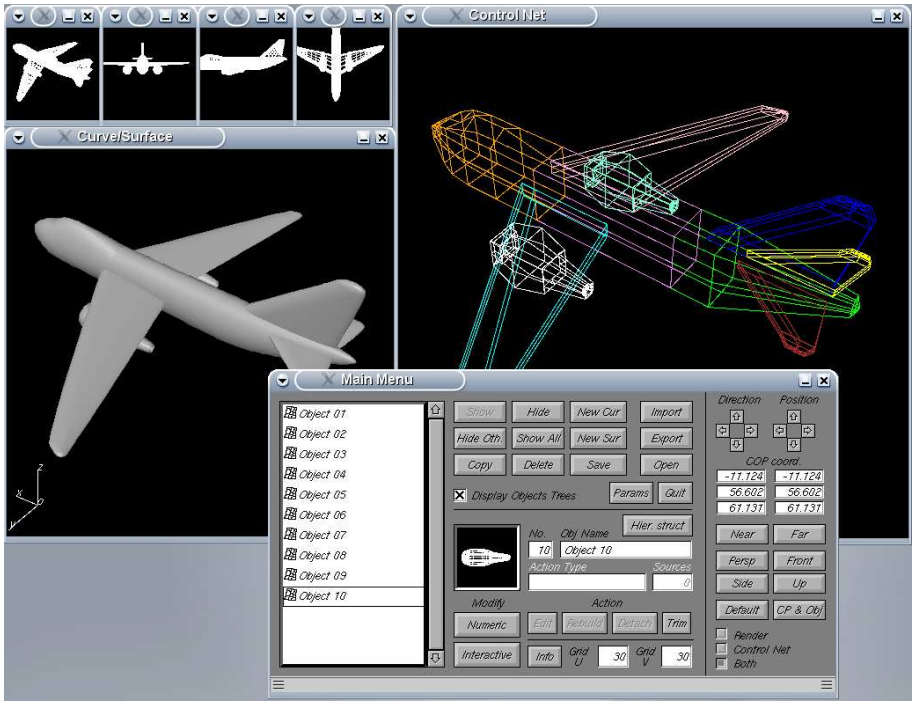


Figura 1.1: Come si presenta *xcsurf*

Può essere visualizzata da sola o insieme alle altre superfici e se viene resa attiva (selezione nel textbox degli oggetti con il mouse) può essere anche modificata.

La griglia di controllo viene visualizzata nella finestra Control Net, mentre nella finestra Curve/Surface viene visualizzata una approssimazione poligonale della superficie.

Mentre la griglia di controllo viene sempre mostrata con tecnica wireframe, per l'approssimazione poligonale viene utilizzata la libreria *trim* di *xcmodel*, che supporta tecniche di real-time rendering quali *wire-frame*, *depth-cueing*, *hidden-line*, *flat-shading*, *Gouraud-shading* e *Phong-shading*.

Per quanto riguarda le modifiche apportabili ad una superficie, esse possono spingersi fino ai livelli più bassi della rappresentazione NURBS, e quindi interessare parzialmente o totalmente il dominio parametrico, le partizioni nodali, i gradi, i punti di controllo, i pesi associati.

Come accennato, è possibile lavorare con liste di superfici, per dar vita ad un oggetto complesso. In questo caso i dati che costituiscono l'elenco sono salvati in file *.obj* e le superfici caricate vengono considerate singoli oggetti. Le griglie di controllo e l'approssimazione poligonale risultante vengono rese nelle finestre specificate sopra, ma solo una delle supercifi, quella selezionata come superficie *attiva*, è modificabile.

## 1.1 I problemi dell'interazione utente

Modellare un oggetto al calcolatore è un compito molto difficile, non solo perché occorre disporre di tecniche potenti e facili da usare, ma anche (e probabilmente soprattutto) perché l'oggetto che si sta creando è virtuale, e l'utente può sentirsi a contatto con esso solo nella misura in cui il calcolatore sa farlo sentire. Gli scultori possedendo l'oggetto tra le mani, possono sfruttare al meglio i propri sensi. In particolare sfruttando il senso tattile hanno la facoltà di compiere azioni con una certa precisione, la quale invece con il calcolatore, è subordinata alla capacità del progettista di indicare un certo valore numerico piuttosto che un altro. Per non parlare del senso visivo, che consente loro di avere una visione immediata ma soprattutto **reale** dell'oggetto. Il computer invece, può solo **far intuire** le tre dimensioni. Azioni come il cambiamento della direzione visiva o l'avvicinarsi all'oggetto sono spontanee e naturali quando si lavora con qualcosa di reale. Anche se i moderni sviluppi della tecnologia, nel campo della realtà virtuale, stanno notevolmente riducendo queste differenze, è probabile che ancora per molto tempo il problema di come deve essere realizzata una ottimale interfaccia CAD permarrà.

Che il problema sia tutt'ora aperto, lo dimostrano due fattori: le continue pubblicazioni sul tema, e il fatto che i pacchetti commerciali non si siano affatto standardizzati in questo senso.

La necessità di rendere di pratico utilizzo alcuni metodi di deformazione non riassumibili con singoli comandi, è stata l'occasione per mettere in discussione il livello di interattività del progetto *xcsurf*, finora lasciato in secondo piano, privilegiando soprattutto l'aspetto della ricchezza di funzionalità e di algoritmi NURBS-based che in campo commerciale non trova riscontro, perché l'aspetto a basso livello (cioè a contatto con la rappresentazione parametrica) viene volutamente celato. La peculiarità di *xcsurf* infatti, è quella di avere anche fini didattici, e per questo motivo, è pure un terreno di sperimentazione sconfinato.

Proprio in questa ottica di sperimentazione si è cercato di fornire delle risposte ai disagi più comuni per un progettista 3D, prendendo ispirazione anche dalle risposte fornite dagli sviluppatori dei maggiori modellatori commerciali, con l'intento di prenderne il meglio, ma adattandolo spesso alle reali esigenze di *xcsurf*.

Poiché però non si vuole perdere di vista il principale obiettivo, ossia quello di ricercare la migliore interazione di supporto alla modellazione per deformazione, si dimostra ora, quanto grande sia l'influenza dell'interfaccia utente su uno dei requisiti che maggiormente sta a cuore, quello della *facilità di utilizzo* del metodo di deformazione.

Per fare questo, si illustra in poche righe come, proprio *xcsurf*, realizza lo spostamento di un control point.

La figura 1.2 mostra la finestra di dialogo, in *xcsurf*, per la modifica di una superficie.

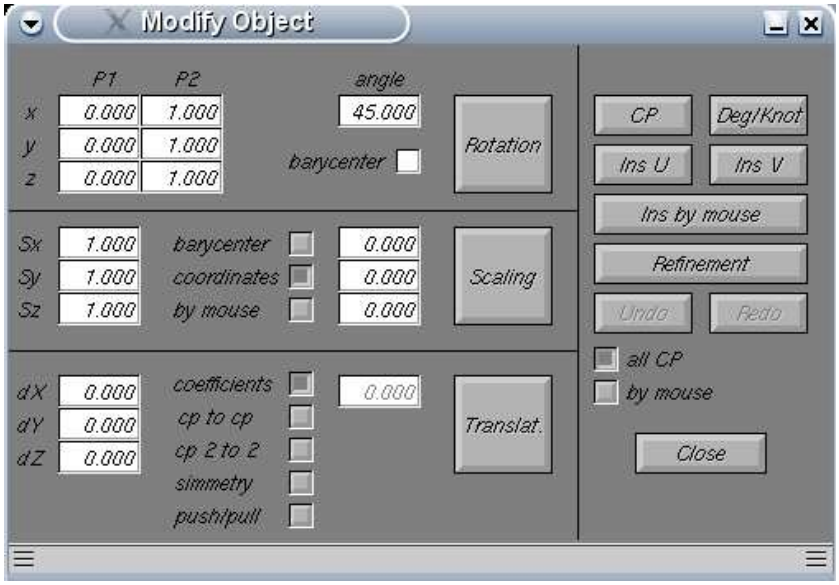


Figura 1.2: La finestra di dialogo per la modifica di una superficie in *xcsurf*

Si immagini di possedere a schermo una superficie e la relativa griglia di controllo della quale si vuole muovere un punto in una posizione idealmente individuata. Il primo passo consiste nel dichiarare al sistema che si intende operare su un sottoinsieme dei punti di controllo e che tale sottoinsieme verrà indicato utilizzando il mouse; quindi, occorre innanzitutto, attivare il check point by mouse. Poi, giacché l'operazione voluta non è altro che una traslazione occorre posizionarsi in corrispondenza dei text box dX, dY, dZ e ivi indicare i coefficienti della traslazione. Ma quanto indicare negli appositi campi di testo? La questione non è di poco conto, perché da un lato ciò che si possiede mentalmente è una informazione visiva, non reale, della **direzione** dello spostamento, e da un altro perché la quantità dello spostamento non è desumibile, se non con moltissima imprecisione, dalle poche informazioni che si conoscono riguardo la superficie: le dimensioni del suo Bounding Box e la posizione del suo baricentro. E' chiaro che l'operazione di spostamento non verrà risolta in poco tempo: nel caso più fortunato una **serie di tentativi** farà convergere gradualmente il punto di controllo nella posizione schermo desiderata. Dopo aver riempito i valori dX, dY e dZ, occorre trasmettere effettivamente al sistema il comando di traslazione. Per fare ciò occorre portarsi in corrispondenza del bottone Translat. e premerlo. A questo punto *xcsurf* chiede che si indichino col mouse i punti di controllo desiderati cliccandovi vicino con il tasto sinistro del mouse e di concludere tale indicazione premendo quello destro. Quindi, occorre spostarsi nella

finestra Control net ed eseguire quanto richiesto. Quando il tasto destro è stato premuto il sistema mostra il risultato della traslazione. A questo punto, facendo tesoro dell'esperienza fatta, si proverà o a ristimare l'entità della traslazione riportandosi alla situazione antecedente al comando impartito (cioè premendo il botton Undo), oppure si cercherà di stimare una traslazione che corregga quella precedente a partire dalla posizione correntemente occupata dal punto di controllo. Nell'uno e nell'altro caso occorrerà ripetere la sequenza di azioni sopra descritte: attivazione del check point- riempimento dei text box-pressione sul bottone Translat.-posizionamento sulla finestra della griglia di controllo-click sul punto di controllo. Ad un certo punto, quest'ultimo sembrerà ben posizionato, ma la conferma di questa ottimistica impressione verrà solo osservando la superficie e la griglia di controllo da diverse posizioni. Quindi, occorrerà portarsi in posizione delle quattro frecce, in alto a destra della finestra di dialogo, e cliccarvi diverse volte per spostare la telecamera in una posizione ideale. In realtà è probabile che questa azione sia stata fatta già più volte durante la serie delle traslazioni.

Con un pò di bravura, non sarà necessario un tempo estenuante per raggiungere il risultato voluto, ma è certo che il più semplice strumento di deformazione non sembrerà affatto poi così semplice.

Provando ad analizzare i fattori di maggior disagio incontrati nell'esperienza descritta emergono alcune critiche alla suddetta interfaccia utente:[1]

**1. Località dei comandi**.

Per impartire più ordini al sistema (es. sposta un CP, muovi la telecamera ..), si è costretti, troppe volte, ad un lungo movimento con il mouse, perché i relativi bottoni sono tra loro distanti. Questo, a lungo, stanca la mano (e il pensiero). Si pensi che in una intera fase di modellazione, di comandi del genere se ne dovranno impartire qualche centinaio.

**2. Pressioni del mouse**.

Per prolungare un comando nel tempo (ad es. effettuare la stessa traslazione per molte volte consecutive) si è costretti a premere continuamente il bottone relativo al comando. Poiché è, a lungo, estenuante **premere** ripetutamente un tasto si preferirebbe notificare al sistema la ripetizione di un ordine ad esempio **spostando** il mouse.

**3. Pilotaggio a distanza (contatto con la superficie)**.

La separazione spaziale tra il luogo in cui si compie l'azione (finestra di

[1] Questo elenco non ha una vera e propria base scientifica: si fonda esclusivamente su osservazioni e opinioni di chi scrive

dialogo) e il luogo in cui questa ha effetto (posizione schermo del control point) penalizza il proprio intuito. Inoltre, costringe ad alternare lo sguardo tra i due luoghi, compiendo di fatto, anche sotto questo punto di vista, un tragitto eccessivamente lungo e stancante. Diversamente, operando direttamente sul luogo della modifica, si può fissare lo sguardo su una piccola regione di monitor.

**4. Intuitività**

Nel compiere un'azione, come ad esempio una rotazione, o una scala, si deve, come al solito, premere un bottone previa indicazione di un valore numerico. Questo è fastidioso, perché verrebbe spontaneo agire con naturalezza, ad esempio spostanto il mouse in senso circolare, per fare una rotazione.

**5. Riferimenti 3D**.

Si deve spingere interamente o solo una parte dell'oggetto in una certa direzione. Si intuisce visivamente quale dovrebbe essere la destinazione, ma non si riesce a tradurla in temine di quantità $\Delta X$, $\Delta Y$, $\Delta Z$ (cioè in pezzi di tragitto spaziale). Come è orientato l'oggetto? C'è una scarsità di riferimenti che suggeriscano dove e come effettivamente l'oggetto stia.

**6. Esattezza della variazione**.

Quali sono le dimensioni reali della parte che si vuole modificare? Si può farne una **stima** in base alle informazioni che vengono fornite sull'oggetto (in genere Bounding Box e posizione baricentro)? Difficilmente. E in base al rapporto tra la dimensione di occupazione-finestra dell' oggetto e quello della parte da modificare? No: il calcolatore offre una visione *prospettica* dell'oggetto; tutte le informazioni di profondità (rispetto al punto di osservazione) vengono scalate. Nulla si può stimare su di esse.

Gli ultimi due problemi sollevati suggeriscono di meditare su una questione cruciale: la scelta di mostrare a schermo una superficie attraverso sua *proiezione prospettica* è la scelta ideale? Quali sono le alternative?

Ad esempio, quando si raffigura sulla carta un oggetto che si ha in mente, lo si fa solitamente, usando la così detta proiezione *parallela*, come in figura 1.3

La proiezione parallela è **irreale**: mai, in nessun caso, nella vita reale, si vedrà un cubo come in figura 1.3! Tuttavia, essa consente di conservare le informazioni di profondità qualunque sia il punto di osservazione. I problemi 5. e 6. sono molto attenuati.

A questo proposito, quasi tutti i pacchetti di modellazione professionali,

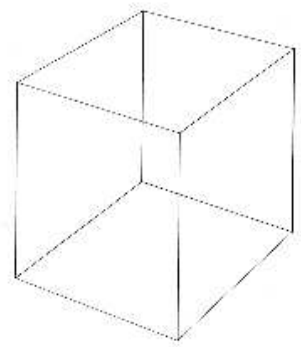

Figura 1.3: Proiezione parallela di un cubo

cercano un compromesso tra le due strategie, cercando di raccogliere i vantaggi dell'approccio parallelo, senza perdere il realismo offerto dalla visione prospettica: l'area dello schermo destinata alla visualizzazione dell'oggetto è divisa in quattro parti, una di esse è riservata alla resa in prospettiva, le altre tre alla visione dall'alto e da due lati della scena (figura 1.4).

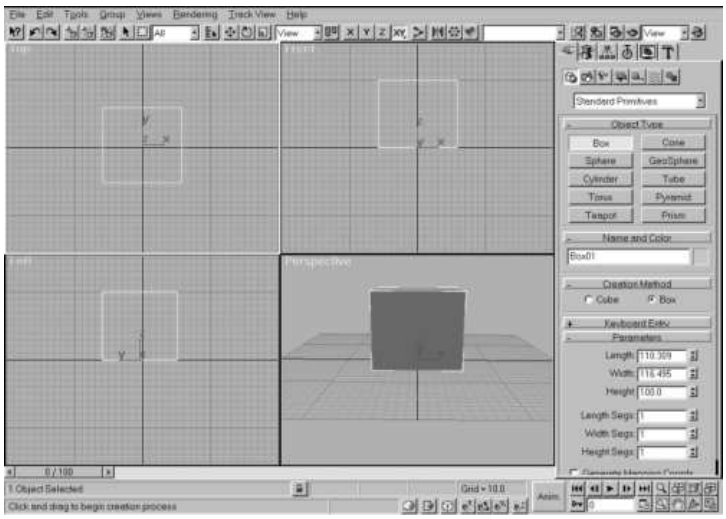


Figura 1.4: Tipica disposizione delle finestre in un pacchetto commerciale

Ciò che essi mirano a fare è risolvere il problema 5. isolando ognuno dei tre piani $XY$, $YZ$, e $XZ$; ogni modifica apportata in uno di essi comporta ovviamente l'aggiornamento immediato delle altre visioni. In pratica, ad esempio, per spostare un control point, l'utente deve compiere una traslazione nella finestra del piano XY e successivamente modificare la z in una delle finestre dei piani $XZ$ o $YZ$ (a seconda di quella che fornisce la visione al momento più comoda). Infine, guardando la finestra della prospettiva egli giudica la bontà del risultato ottenuto.

Con tale sistema, il problema 6. è pure parzialmente risolto, perché lo sfondo viene opportunamente rigato (e di solito ogni intervallo di rigatura ha idealmente ampiezza un metro), sicché l'utente riesce con relativa precisione a dimensionare ogni parte dell'oggetto (in realtà, per certi punti di osservazione, la cosa non è comunque priva di difficoltà).

Sebbene la strategia commerciale rappresenti indiscutibilmente un miglioramento rispetto la rappresentazione a proiezione parallela, in *xcsurf* non è stata adottata, conservando così la originaria strategia di una **unica** finestra[2] e della **sola** resa prospettica. Per i seguenti motivi:

1. benché parzialmente risolva i problemi 5. e 6. l'idea della separazione della modellazione 3D in tre fasi distinte di modellazione 2D va ad aumentare la quantità e la lunghezza degli spostamenti manuali (problema 1.)

2. la strategia dei tre piani è familiare solo a chi ha una certa esperienza di progetto geometrico ed è in generale adatta a chi è solito creare un disegno cartaceo (usando appunto la proiezione parallela) nel quale vengono precisamente indicate le dimensioni di ogni parte dell'oggetto. In questo caso la traduzione delle informazioni cartacee in comandi al programma è quasi sempre immediata

3. in *xcsurf* le entità da rendere a schermo sono **irrinunciabilmente** due: la superficie e il poliedro di controllo, i quali per ragioni di chiarezza è opportuno che, da un lato, **non siano sovrapposti** e da un altro, siano **contemporaneamente visibili**. In tali ipotesi un'adozione della strategia filo-commerciale determinerebbe una non ben chiara ripartizione dello schermo

4. il carattere sperimentale di *xcsurf* spinge a percorrere strade nuove e ad inventare soluzioni alternative ai problemi sopra menzionati

Il prossimo paragrafo mostrerà come a tali problemi si sia cercato in questo rapporto, di dare una qualche risposta, non rifiutando là dove possibile, il confronto con la strategia commerciale.

Si considera importante sottolineare che quello che si delinea in questa sezione è il profilo di un piccolo pacchetto di modifica-deformazione di un oggetto 3D concepito dagli autori per risultare *user-friendly*.

Ora, è al quanto presuntuoso pensare che lo sia effettivamente, e per tutti; anzi, è certo che molti di coloro che avranno modo di usarlo solleveranno critiche ed obiezioni. Probabilmente non c'è concetto più opinabile di amichevole per l'utente.

[2]Per unica finestra si intenda una sola finestra riservata alla resa della superficie; in realtà, un'altra finestra è riservata alla visualizzazione della griglia di controllo

Capitolo 2

# Soluzioni in *xcsurf*

## 2.1 Località dei comandi e contatto con la superficie

La preoccupazione maggiore è rendere immediatamente disponibili le azioni che l'utente dovrà fare con maggiore frequenza. A tal proposito si possono individuare, in base all'esperienza, le seguenti categorie:
**a) azioni di altissima frequenza:** cambio della telecamera, ingrandimento/rimpicciolimento *uniforme* dell'oggetto
**b) azioni di modesta frequenza:** comandi di modifica, trasformazione geometrica, spostamento di control point, deformazioni, ...
**c) azioni di bassa frequenza:** settaggio di parametri: tipo di rendering, step incrementali di rotazione, scala, visione.. e in genere le azioni che l'utente richiede siano fatte digitando **precisamente** dei valori.

Poiché l'obiettivo è di rendere intuitiva, amichevole, ma anche veloce, la modellazione si isola il gruppo c) fuori dall'ambiente di modifica, ossia in una apposita finestra di dialogo da richiamare quando, e se, necessaria. Per le altre azioni si vorrebbe che l'utente non dovesse mai abbandonare col puntatore del mouse la finestra di visualizzazione.

In particolare le azioni del gruppo a) devono essere attuabili nel modo più **naturale e pratico** possibile; si vuole svincolare l'utente addirittura dall'onere di selezionarle da qualche elenco di comandi.
Lo schema proposto è il seguente:

(**praticità**)

a) se l'utente clicca col bottone 1 del mouse [1] , in un punto schermo

[1] Nel sistema grafico XWindow i tasti del mouse vengono definiti bottoni. Analogamente, in questa sezione, per bottone 1 si intenderà il tasto sinistro del mouse, per bottone 3

coperto dell'oggetto visualizzato[2], egli implicitamente dice al sistema che intende operare sull'oggetto.

b) se clicca esternamente all'oggetto, intende modificarne la visione; a seconda del bottone premuto egli intende cambiare il **punto** di osservazione (bottone 1) o la **vicinanza** dell'osservazione (bottone 2).

(**naturalezza**)

se dal click esterno segue uno spostamento verticale, verso l'alto [verso il basso], allora l'utente vuole vedere la superficie da un punto di vista più basso [più alto] (in effetti, idealmente sta prendendo dal basso la superficie e la sta ruotando in modo da vederne il fondo); se il movimento è orizzontale, l'utente vuole conservare l'altezza della telecamera, ma vuole girare attorno all'oggetto in senso laterale.

Le azioni del gruppo b) sono molte. Per esse è inevitabile un elenco, ma si vorrebbe che questo elenco fosse immediatamente disponibile.

Ecco la proposta: se l'utente preme il **terzo** bottone del mouse, dovunque esso si trovi, gli appare un *menu contestuale* (anche noto come *menu rapido*) contenente **il più piccolo insieme di icone** comprendente tutte le possibili azioni che egli può fare sull'oggetto.

Esempio: comando di spostamento di un control point (figura 2.1, parte sinistra), ma, da un punto di osservazione migliore.

Azioni:

1. click in un punto esterno alla superficie col bottone 1 del mouse. Movimento (con mouse premuto) verso il basso e un pò a sinistra (figura 2.1, parte destra).

2. leggero ingrandimento (avvicinamento) dell'oggetto per osservare più in dettaglio il control point: click col bottone 2 verso sè stessi (figura 2.2, parte sinistra).

3. selezione del comando modifica control point: pressione del bottone 3 (non occorre spostare il mouse). Appare il menu rapido. Il cursore viene posizionato al centro del menu (si considerano tutti i comandi equiprobabili) (figura 2.2, parte destra). Quale icona selezionare verrà precisato più avanti.

quello destro, e per bottone 2, se presente quello centrale, altrimenti i bottoni sinistro e destro contemporaneamente

[2]Per oggetto si intende la superficie, il suo poliedro di controllo, ma anche, come si vedrà, il graticolo ENFFD o le curve wire e reference

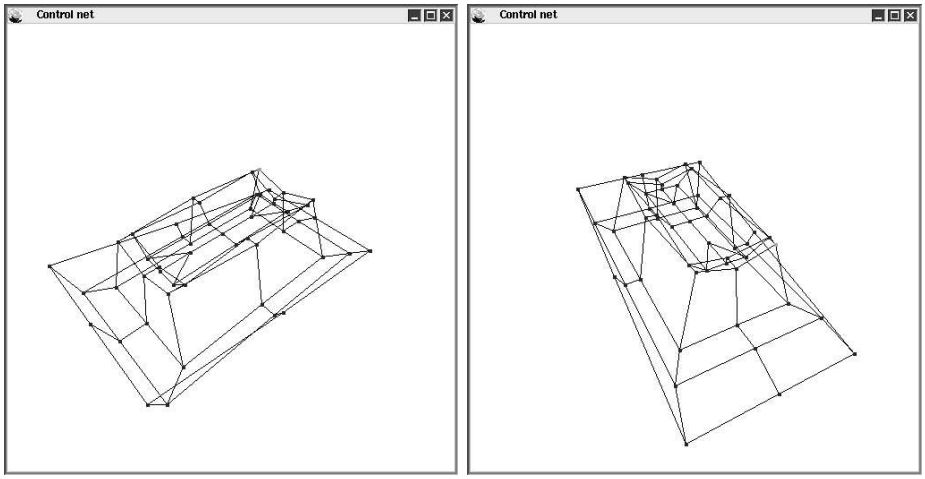


Figura 2.1: Sistemazione della telecamera per meglio osservare un control point

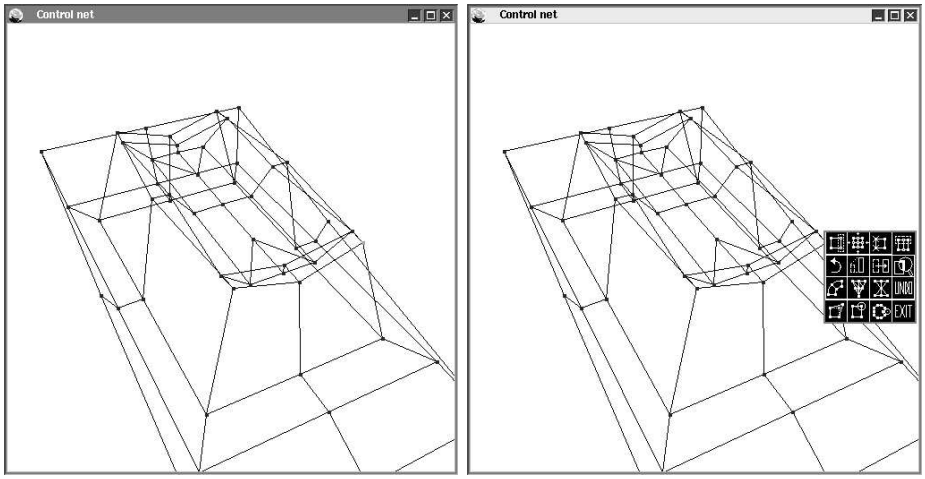


Figura 2.2: Focalizzazione di un control point e richiamo del menu rapido dei comandi

Allo scopo di chiarire meglio questo tipo di interazione si fornisce lo schema pseudo-procedurale che lo realizza. Esso ha una struttura fissata che vale per ogni comando (o come spesso si dirà per ogni *opzione*).

```
//--------------------------
   main loop
//--------------------------
{
 done=0;
 while (!done){
   svuota_coda_eventi(); //verra' chiarita in seguito
   Next_Event(&evento);
   switch (evento){
    case XXXX: menu=opzXXX;
    ...
    case EXIT: done=1;
   }
  if (!done)&&(menu) mostra_menu_rapido();
 }

}//end main loop


//--------------------------
int opzXXX(evento)
//--------------------------
{
 exit=0
 switch (evento){
   case ButtonPress
     switch (tipo_bottone){
        case Button3: exit=1;
        case Button1:
             if "click interno al piu' piccolo rettangolo
                 schermo che racchiude la superficie"
             then  tipo_click=1;
             else if " click su un punto (dipende dall'azione)
                       particolare"
                  then tipo_click=2;
                  else  (dipende dall'azione);
        }

   case MouseMovement:
```

```
      switch (tipo_click){
         case Button3: cambio_telecamera(button_current_press);
         case Button1: do_opzXXX();
         }
 }
 return exit;

}//end opzXXX
```

Il menu rapido è una finestra *modale*: essa cattura tutti gli eventi riferendoli a se stessa. Per uscirvi, l'utente può cliccarvi esternamente (restaurando la situazione precedente all'ingresso) o selezionando una opzione.

In precedenza è stato sottolineata una caratteristica che si vuole imporre al menu di selezione: contenere il più basso ma sufficiente numero di icone, al fine di minimizzarne la complessità e, di riflesso, velocizzare le selezioni.

Per far questo ci si è serviti della seguente osservazione: comandi che si riferiscono all'interazione col poliedro di controllo sono attivati (per il principio di località) nella window dei control point, quelli relativi alla superficie nell' altra finestra. Onde si possono prevedere **due menu rapidi separati**: la pressione del bottone 3 attiverà il menu associato alla finestra in cui avviene la pressione.

Si noterà nel proseguo come si sia cercato di dare ai due menu una struttura razionale comune.

Il fatto di possedere due menu (e quindi di dimezzare il numero di icone/menu) non è sufficiente. Ad esempio, tra i comandi che sicuramente riguarderanno la superficie, figurerà quello di rotazione. Ma come si è visto nel capitolo precedente i metodi di deformazione pur applicati fisicamente ai control point, sono virtualmente applicati ai punti della superficie. In altre parole è nella finestra Render/Icons che l'utente creerà il graticolo FFD e probabilmente vorrà ruotarlo. Si dovrebbero quindi prevedere due icone per la rotazione? No. La strategia addottata in *xcsurf* potrebbe essere definita polimorfismo delle icone: **ogni icona indica quale è l'azione, non su che cosa è applicata; ciò è implicitamente stabilito dal tipo di oggetto con cui l'utente sta lavorando (superficie, graticolo ENFFD, curva..)**. Ancora una volta si demanda al sistema la deduzione del significato di una azione sulla base del contesto in cui è fatta.

Nel prossimo paragrafo si vedrà più in dettaglio ciascuna icona, o meglio, ciascuna funzionalità della nuova sezione di modifica di *xcsurf*.

Si discute, in conclusione, brevemente, la strategia commerciale in riferimento al problema della località. In genere quasi tutti i pacchetti pro-

fessionali vogliono fornire una vastissima varietà di opzioni e sotto opzioni fondendo interattività di tipo contatto con la superficie a quella di tipo numerico. E' molto probabile che un programma commerciale che non possedesse (e non rendesse evidenti) un gran numero di opzioni non darebbbe l'impressione di ricchezza e meritevolezza del costo economico richiesto agli acquirenti. Inoltre quanto più un modellatore è professionale, tanto più deve poter soddisfare ogni possibile richiesta immaginabile da parte dell'utente, il quale dal canto suo, ha il tempo materiale e l'intento professionale di familiarizzare con ogni piccolo tassello dell'applicazione. Non è così per *xcsurf*, nè per i fini, nè per il target umano. Va notato tuttavia, come anche la scelta di velocizzare la selezione di comandi frequenti sia pressocchè scartata a livello commerciale: ad es. il cambio della telecamera e lo zoom è quasi sempre attivabile tramite due icone di una toolbar, ossia in una posizione schermo ben distante dalla finestra di lavoro. Inoltre, non viene mai decifrato il tipo dell' azione dal luogo in cui il click si compie. Probalbilmente, è considerato rischioso lasciare al progettista l'onere di cliccare **esattamente** esternamente alla superficie; in effetti nello schema presentato sopra, viene considerato esterno alla superficie un punto che sta fuori del più piccolo rettangolo schermo che racchiude la superficie stessa; può accadere, in definitiva, che l'utente clicchi in una posizione **apparentemente** esterna alla superficie. Ma la sperimentazione di questo procedimento ha dimostrato che ciò accade molto raramente, e, quando accade, non si hanno mai effetti collaterali.

## 2.2 Riferimenti 3D ed esattezza della variazione

Come osservato, queste problematiche sono accentuate in *xcsurf* dal fatto che si dispone della sola resa prospettica, che obbliga a ragionare in tre dimensioni, pur fornendo una scarsa cognizione della profondità. Un modo per simulare l'approccio dei pacchetti commerciali è quello di avere quattro bottoni che posizionano direttamente la telecamera in alto, di lato ecc. e agire in successione su alcune di tali viste. Ad es. per effettuare uno spostamento $(\Delta x, \Delta y, \Delta z)$ si potrebbe vedere la superficie dall'asse $z$ (piano $XY$) ed effettuare uno spostamento 2D $(\Delta x, \Delta y)$ e quindi posizionare l'osservazione a lato (ad es. sul piano $XZ$) ed effettuare $\Delta z$. Tuttavia avendo difficoltà ad intuire l'esatta quantità $\Delta x, \Delta y$ e $\Delta z$ si dovrebbe saltare da una vista all' altra molte volte. Inoltre, l' obiettivo è quello di effettuare un'azione con un'unica sequenza pressione mouse-trascinamento-rilascio. Infine, poichè comunque, la bontà del risultato è verificabile solo visionando la superficie in modo 3D e da molte angolature diverse, tanto vale agire direttamente sull'immagine prospettica. Si pone quindi il problema di **come muovere un punto nelle tre direzioni attraverso un dispositivo (mouse) bidimensionale**. Probabilmente questo è stato uno degli aspetti più interessanti affrontati in

questo lavoro. Per altro risulta pure un aspetto poco esplorato nella letteratura scientifica. Sicuramente, essendo un segmento 2D (movimento schermo del mouse) non mappabile in uno 3D (spostamento reale) il numero minimo di spostamenti che si dovranno effettuare con il mouse saranno due.

Si è cercato allora un metodo che limitasse, nel caso medio, a due, gli spostamenti, ma che fosse nel contempo facilmente utilizzabile. Una soluzione è quella di pensare idealmente che la posizione corrente del mouse, nel tappetino, sia il centro di un cerchio suddiviso in otto settori. Ogni settore corrisponde ad un semipiano: $+X+Y, +X-Y, +Z+Y$, ecc. A seconda del settore nel quale avviene il movimento e a seconda della sua inclinazione si riesce a far corrispondere nello spazio uno spostamento planare; con due movimenti si riesce ad effettuare una qualunque variazione $\Delta x, \Delta y, \Delta z$. Purtroppo il difetto principale di questa tecnica, sta nella immensa precisione cui si richiede all'utente (i settori circolari hanno una ampiezza troppo piccola). A tal proposito è opportuno notare che per dare al movimento del mouse un carattere di inclinazione occorre decidere come identificare la fine di un movimento (e quindi procedere al calcolo della sua angolazione) e l'inizio del successivo (non si vuole che il bottone venga rilasciato). Si chiamino $x, y$ le coordinate del pixel corrente puntato dal mouse e *xprec* e *yprec* quelle del pixel da cui è iniziato il movimento. Se si calcola uno spostamento 3D ogni volta che $(x \neq xprec)$ oppure $(y \neq yprec)$ allora, poichè si lavora con pixel e quindi $||(xprec, yprec) - (x, y)|| = 1$, si può contare solo su una inclinazione pari a $k \times 45$ gradi $(k = 0, 1, ...)$ . La soluzione adottata in *xcsurf* è stata quella di calcolare uno spostamento 3D ogni qualvolta $|x - xprec| \geq 3$ oppure $|y - yprec| \geq 3$.

Un'altra idea per realizzare un movimento 3D è quella di **identificare con due bottoni diversi, uno spostamento in un piano, e uno spostamento nella direzione restante**. Ad es. in *xcsurf* è stata implementata la seguente regola:

1. se si trascina il mouse col bottone 1 premuto, si determina uno spostamento nel piano $XY$

2. se dal tasto 1 si passa al tasto 2, lo spostamento avviene, conservando le quote $\Delta x$ e $\Delta y$ raggiunte, nella direzione $z$

Come vengono calcolate le quote $\Delta x$ e $\Delta y$? In realtà basta che esse rispecchino l'andamento a schermo del mouse:

1. sia $(x, y, z)$ la locazione 3D del punto che si vuole spostare. Si fissi una quota di spostamento $\Delta S$.

2. si calcoli l'angolo $\theta$ tra il vettore dell'asse $x$ (o $-x$) proiettato e quello tracciato (idealmente) dal puntatore del mouse sullo schermo

3. si ponga $\Delta x = \Delta S \cos\theta$, e $\Delta y = \Delta S \sin\theta$

4. si calcoli il nuovo punto $(x + \Delta x, y + \Delta y, z)$ e lo si proietti, ottenendo $(x_s, y_s)$

5. si sposti il puntatore del mouse in $(x_s, y_s)$

Tale schema dà efficentemente l'**illusione** all'utente di muoversi in 3D secondo la direzione di movimento del mouse.

La strategia di separare uno spostamento 3D in due spostamenti, uno planare e uno lineare, si è dimostrata in molti casi molto valida, ed è di solito la più adatta per creare un parallelepipedo (ad es. di control point ENFFD) giacchè è comodo definirne prima le dimensioni $XY$ (fissando un estremo e spostando l'altro in un piano) e successivamente la dimensione restante (spostando in $z$ uno dei due estremi). Inoltre, essa consente di raggiungere una qualunque localizzazione 3D con al più una coppia di movimenti. Tuttavia, ha il difetto di utilizzare due bottoni, uno dei quali potrebbe essere adoperato per compiere altre funzioni.

Per cui si è cercato di scovare una regola che necessitasse di tre (e non più due) movimenti, ma che utilizzasse un solo bottone.

Una regola realizzata è stata la seguente: l'utente **con un solo movimento, può spostare un punto in una sola delle tre direzioni $X, Y$ e $Z$, positive o negative, e lo può fare trascinando il mouse parallelamente alla proiezione dell'asse relativo alla direzione desiderata**.

L'implementazione di questa regola richiede un modo per determinare a quale asse proiettato si riferisca un certo movimento del mouse. Ciò può essere fatto semplicemente così :

1. si assuma l'origine del sistema d'assi proiettato come il centro di una circonferenza sullo schermo

2. si calcoli, per ognuno dei sei assi proiettati, l'angolo descritto in tale circonferenza ($\theta_i \quad i = 1..6$)

3. si porti il vettore idealmente tracciato sullo schermo dal movimento del mouse nel centro della circonferenza e se ne determini analogamente l'angolo descritto ($\theta$)

4. l'asse di movimento è quello per cui la distanza angolare tra $\theta$ e il rispettivo $\theta_i$ è la più piccola

Questa tecnica si è dimostrata molto efficiente, perchè sebbene implichi tre spostamenti, la loro facilità fa si che l'utente arrivi in poco tempo alla posizione desiderata. Tuttavia non è esente da inconvenienti: quando due assi vengono proiettati molto vicini e magari uno di loro non viene mostrato (gli assi negativi non sono resi) l'utente può incorrere in errori, oppure deve cambiare il punto di osservazione (e quindi le proiezioni dei vari assi).

La ricerca di una regola di movimento è quindi proseguita, nel tentativo di risolvere anche quest'ultimo problema. In effetti esso non si presenterebbe mai, se, sempre, gli assi fossero proiettati come nella figura 2.3.

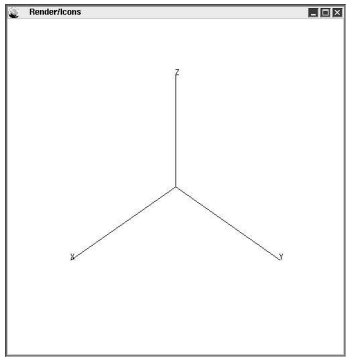


Figura 2.3: Proiezione del sistema d'assi virtuali, di riferimento per il movimento in tre dimensioni

L'idea è di celare del tutto all'utente la reale orientazione dell'oggetto nella scena, e di fare in modo che egli si muova in 3D seguendo la direzione degli assi virtuali in figura 2.3. Quando l'utente ad esempio sposta il mouse nella direzione dell'asse virtuale $x$ egli vede spostarsi effettivamente il punto in quella direzione, mentre fisicamente questo può aver subito una traslazione in tutte le tre direzioni reali.

Implementare questa astrazione può sembrare banale; in realtà non lo è. La difficolta stà nel calcolare le reali quote $\Delta x$, $\Delta y$ e $\Delta z$ corrispondenti ad un certo movimento lungo un asse virtuale. La cosa può essere risolta **determinando a quale sistema di riferimento corrisponda in un certo momento la proiezione degli assi della figura 2.3**.

E' chiaro che l'utente sta muovendo il punto in una sola direzione $x, y$, o $z$ di tale sistema di riferimento. Per calcolare quale siano poi, effettivamente le quote $\Delta x$, $\Delta y$, $\Delta z$ nel sistema reale della scena, basta effettuare una conversione di coordinate da un sistema di riferimento ad un altro. Ma è la determinazione di quale sia questo sistema di riferimento virtuale, a rappresentare l'aspetto intricato dell'idea.

Esso non è fisso, anzi, deve essere rideterminato ogni qualvolta cambia il punto di osservazione della scena (COP).

Si tratta allora di capire la legge che lega una variazione delle coordinate del COP ad una variazione del sistema di riferimento virtuale.

Innanzitutto, va detto che per definire un sistema di riferimento bastano quattro punti: l'origine e tre punti che stanno sugli assi principali (gli assi $X$, $Y$ e $Z$) del sistema in questione. Dicasi $O, a_x, a_y$, e $a_z$ le coordinate di questi quattro punti nel sistema reale, cioè quello della scena. Siano inoltre $cs + cart(D, \theta, \phi)$ le coordinate del COP (dove $cs = (cs_x, cs_y, cs_z)$ è il centro

della scena e $cart(s)$ è l'operazione di conversione in coordinate cartesiane di un punto $s$ espresso in coordinate sferiche).

Si può facilmente notare che quando $\theta = \phi = 45$ gradi, ogni sistema di riferimento che viene proiettato come nella figura 2.3 soddisfa le seguenti:

$$\begin{aligned} O &= cs \\ a_x &= cs + cart(\alpha, 0, 90) \\ a_y &= cs + cart(\alpha, 90, 90) \\ a_z &= cs + cart(\alpha, 45, 0) \end{aligned} \tag{2.1}$$

dove $\alpha$ è un valore arbitrario (per semplicità si pone $\alpha = 2$).

In pratica, quando l'osservatore è posto ad una quota $\theta = \phi = 45$ gradi, i tre versori principali del sistema di riferimento della scena vengono proiettati secondo la figura 2.3.

Ora, si varino $\theta$ e $\phi$ (cioè l'osservatore si posta). Come devono variare $a_x, a_y$ e $a_z$ affinchè essi esprimano un sistema di riferimento che viene proiettato come nella figura 2.3?

Poichè la ( 2.1) mostra gli sfasamenti tra le componenti $\theta$ e $\phi$ di $a_x$, $a_y$ e $a_z$ rispetto quelle dell'osservatore, è naturale pensare che se tali sfasamenti restano immutati allora anche la proiezione del nuovo sistema di riferimento deve restare immutata. Cioè dovrebbe senz' altro essere:

$$\begin{aligned} a_x &= cs + cart(2, \theta - 45, \phi + 45) \\ a_y &= cs + cart(2, \theta + 45, \phi + 45) \\ a_z &= cs + cart(2, \theta, \phi - 45) \end{aligned} \tag{2.2}$$

La ( 2.2) è tanto intuitiva quanto errata! In realtà, è difficile osservarlo ma risponde al vero, solo $a_z$ è corretto nella ( 2.2). Per semplificare le cose si fissi $\theta = 45$ gradi. Allora, $a_x$ al variare di $\phi$ (cioè se l'osservatore si sposta verso l'alto o verso il basso della scena) si muove lungo un cerchio giacente in un piano parallelo a quello del cerchio lungo cui si muove il COP. Una vista, dall'alto, di tale cerchio è illustrata in figura 2.4. Se l'osservatore si sposta di una quantità $\Delta\phi$, $a_x$ deve certamente percorrere un arco $\Delta\phi$, ma, del cerchio di raggio $R$ e centro $C$ in figura 2.4.

Dalla stessa figura sono ricavabili i valori di $C$ ed $R$:

$$C = (1, -1, 0) + cs \quad R = \sqrt{2}$$

Per determinare le coordinate di $a_x$ localmente al cerchio della figura 2.4 si può far coincidere C con l'origine di un piccolo sistema di riferimento, in cui i punti della circonferenza hanno coordinate $(\sqrt{2}, 45, \phi)$. In definitiva, se l'osservatore si sposta di una quantità $\Delta\phi$, per calcolare $a_x$ si procede come segue:

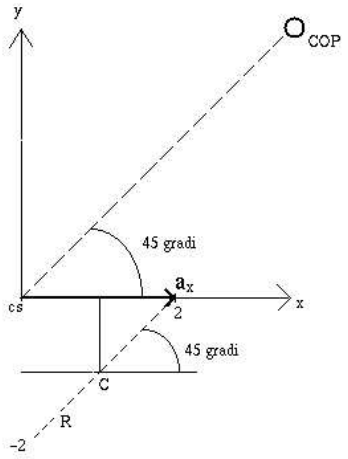


Figura 2.4: Vista dall'alto del percorso effettuato dall'asse $x$ del sistema virtuale

1. si calcola il punto 3D $P = cart(\sqrt{2}, 45, \phi)$
2. si trasla $P$ in modo che il cerchio risulti centrato in $C$: $a_x = P + C$

   $a_y$ si tratta in modo del tutto analogo.

Per considerare anche uno spostamento $\Delta\theta$ dell'osservatore, basta (ed è facile rendersene conto) ruotare il sistema così determinato, di un angolo pari a $\Delta\theta$ rispetto l'asse $z$, centrando tale rotazione nel punto $cs$.

Riassumendo, per ogni movimento dell'osservatore si è determinato un nuovo sistema di assi virtuale che viene proiettato sempre nella stessa posizione, ma che in realtà definisce un nuovo sistema di riferimento. Quando l'utente muove il mouse in direzione dell'asse virtuale $x$, ad esempio, il sistema determina un punto $(\Delta x, 0, 0)$ e lo converte nel sistema di riferimento della scena. Tale conversione dà il reale incremento da assegnare al punto che si sta muovendo.

Ciò di cui si è finora parlato, è come si possa muovere un punto nelle tre dimensioni con due o tre movimenti del mouse. Non si è mai parlato invece, di quello che si potrebbe chiamare feedback del movimento. Poichè l'utente non **riesce ad intuire in che direzione spostare un punto, se non gli si dà facoltà di capire, sia in che modo egli lo sta spostando, sia in che posizione della scena il punto si trovi, occorre fornirgli un qualche responso grafico di ogni spostamento da egli compiuto**. In altre parole si tratta di risolvere il già menzionato problema della necessità di riferimenti 3D.

Un altro sforzo di inventiva, ha condotto a realizzare per *xcsurf* tre tipi di feedback del movimento Il primo è illustrato in figura 2.5.

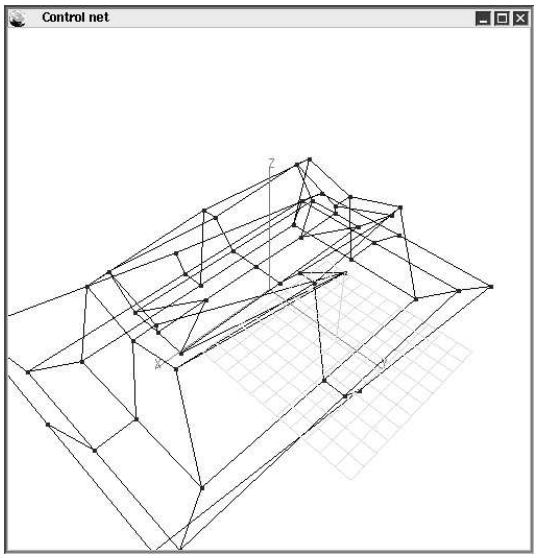


Figura 2.5: Responso grafico della posizione di un punto 3D attraverso grigliatura planare nello spazio

Viene creata una griglia nel piano $XY$ con origine nel punto 3D corrispondente al punto di pressione del mouse. Una linea collega la coordinata $z$ raggiunta, con la griglia (nelle posizioni $x$ e $y$ correnti). Quando è utile questo responso visivo? La grigliatura dà l'idea abbastanza precisa di quanto il punto venga mosso di una quantità $x$ e una quantità $y$ (basta contare i trattini nell'una e nell'altra direzione); se l'utente deve sottoporre allo stesso spostamento $(\Delta x, \Delta y, \Delta z)$ più punti, tali informazioni potrebbero risultargli preziose.

Un secondo feedback si propone di fornire una informazione visiva più globale, ossia di mostrare in quale punto della scena il punto si trovi (figura 2.6). Tre semirette congiungono il punto con altrettante facce di un cubo immaginario che racchiude la scena.

L'ultimo feedback si ispira al precedente ma vuole rispecchiare la tecnica di movimento 3D basata su assi virtuali. In effetti abbinare il feedback precedente a tale tecnica può confondere l'utente, che verrebbe indotto a seguire le inclinazioni dei piani del cubo scena, anzichè quelle degli assi di figura 2.3.

La figura 2.7 illustra il feedback creato.

Come si osserva, gli assi del sistema virtuale non vengono visualizzati, in quanto superflui: essi hanno la stessa inclinazione dei tre piani virtuali

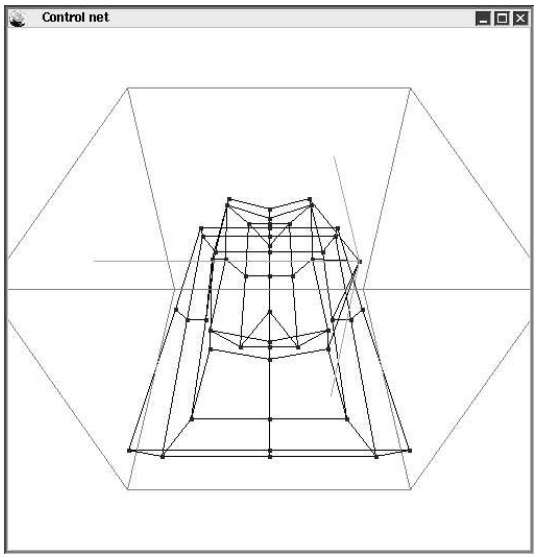


Figura 2.6: Responso grafico della posizione di un punto 3D attraverso visualizzazione del cubo che racchiude la scena

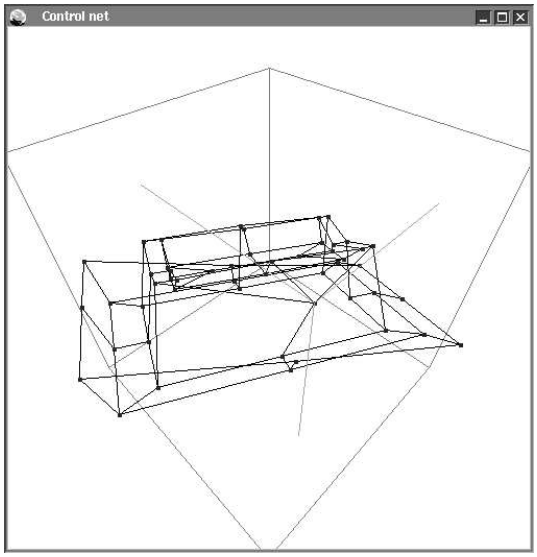


Figura 2.7: Responso grafico della posizione di un punto 3D attraverso visualizzazione dei piani virtuali

mostrati in figura. Questi ultimi restano fissi, qualunque sia la posizione della telecamera.

Tutti i metodi di movimento inventati, nonchè i relativi feedback, hanno un ruolo non piccolo nella nuova sezione di modellazione di *xcsurf*: essi saranno utilizzati non solo per spostare control point della superficie, ma anche per spostare il centro delle trasformazioni geometriche (rotazione, scala, traslazione) e quello delle deformazioni di Barr, per posizionare i control point delle curve reference e wire, per editare gli estremi del graticolo ENFFD, e per altro ancora.

## 2.3 Intuitività dell'azione

Si vorrebbe stabilire una qualche relazione tra un comando e l'azione utente che lo realizza (in un certo senso come è stato per il cambio della telecamera, quando la direzione di movimento del mouse indicava la nuova direzione visiva). Inoltre, è apprezzabile che il sistema dia un responso visivo anche quando un comando non determina modificazioni visibili dell'oggetto. Come primo esempio si considerino i comandi di rotazione e scala. Sarebbe piacevole effettuarli con un gesto spontaneo, cioè che rispecchi quello che idealmente si farebbe se la superficie fosse un oggetto sulla propria scrivania. L'idea in questa tesi è stata la seguente: per fare una rotazione, ad es. attorno l'asse $x$, si immagina di afferrare l'asse delle $x$ e torcerlo come se si dovesse avvitarlo. Per fare una scalatura, ad es. nella direzione $z$, si afferra idealmente una estremità dell'oggetto e la si tira nella direzione $z$, appunto. Le figure 2.8 e 2.9 mostrano come queste metafore siano state tradotte in pratica: per compiere una rotazione l'utente clicca in prossimità dell'asse (figura 2.8, parte sinistra) e compie lo spostamento che, visto l'orientamento dell'asse, maggiormente si avvicina all'azione di torsione (nella figura 2.8, parte destra, la freccia indica il percorso fatto dal mouse); per compiere una scala, l'utente fissa l'estremità di una superficie vicina all'asse di scala (figura 2.9, parte sinistra), e quindi tira il mouse nella direzione desiderata (figura 2.9, parte destra).

Intuitività, spesso significa anche rinunciare a qualche funzionalità; ad esempio, non si è prevista la possibilita' di effettuare un rotazione attorno ad una asse *qualunque* (cioè anche diverso dai tre principali), perchè non si è trovato un modo abbastanza intuitivo (e preciso) per creare un asse arbitrario. D'altro canto, non è affatto scomodo, con il metodo di sopra, compiere una rotazione attorno ad un asse arbitrario servendosi di due rotazioni diverse.

Come già accennato, si vorrebbe che il sistema avvertisse dell'esito di un comando anche quando quest'ultimo non determina un'evidente modifica dell'oggetto (si pensi, ad esempio, all'operazione di selezione dei control point). La figura 2.10 mostra come selezionare tutti i punti di controllo nei bordi laterali di un cubo, in modo rapido: l'utente visiona il cubo

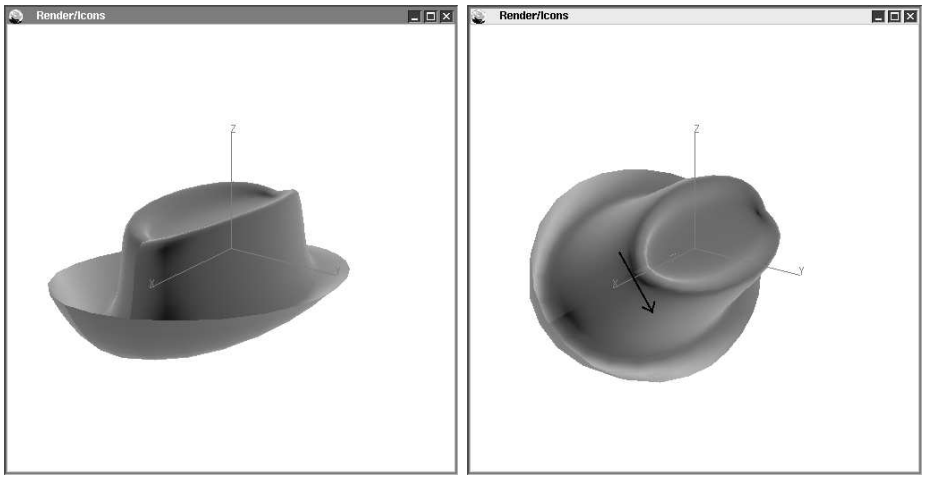


Figura 2.8: Rotazione di una superficie

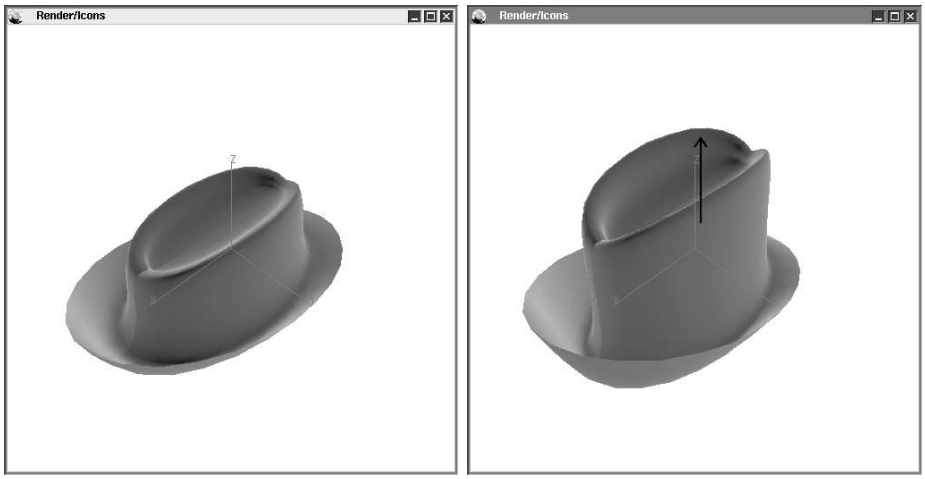


Figura 2.9: Scalatura di una superficie

dall'alto, e crea un rettangolo schermo; ogni control point proiettato internamente al rettangolo viene selezionato (il programma lo colora di rosso: figura 2.10, parte sinistra); successivamente, l'utente crea un rettangolo interno al precedente e i control point visivamente appartenentivi vengono quindi deselezionati (e colorati di giallo: figura 2.10, parte destra). Il colore dei punti di controllo suggerisce il buon esito dell'operazione.

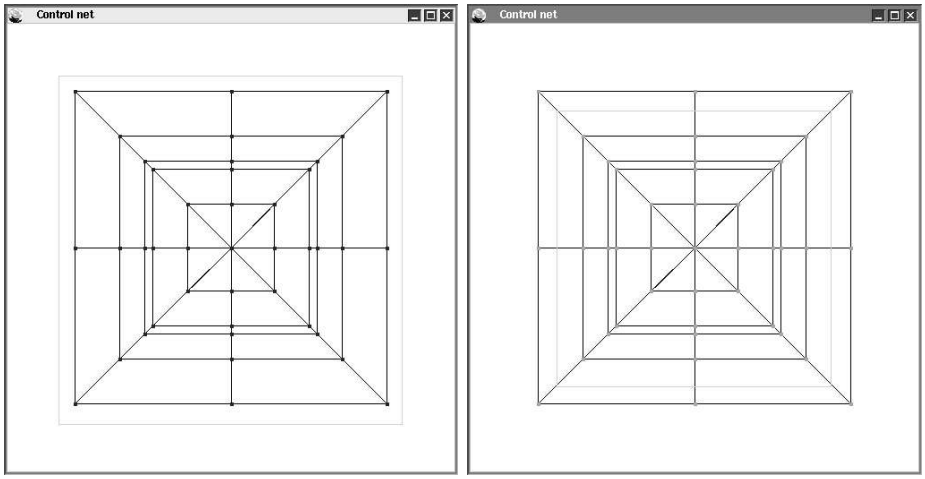


Figura 2.10: Selezione dei control point nei bordi laterali di un cubo

Come ultimo esempio (ma l'utente di *xcsurf* ne scoprirà molti) si consideri l'operazione di freezing di un graticolo ENFFD: quando il graticolo viene congelato esso assume il colore più intuitivo per una congelazione, ossia il grigio chiaro (il bianco è riservato).

Sebbene i colori in *xcsurf* siano facilmente riconfigurabili ciò che si è cercato di fare è di introdurre un vero e proprio **linguaggio dei colori**: **bianco**: è il colore della superficie correntemente in fase di modifica (colore il quale, su sfondo nero dà il maggior contrasto); **grigio**: è il colore delle superfici parzialmente oscurate perchè non coinvolte nel processo di modifica; **rosso**: è il colore della **selezione**; qualunque strumento di selezione (selezione di control point ENFFD, o della curva wire, o di un'area di zoom..) utilizzerà tale colore per marcare ciò che si è selezionato (o si sta selezionando); **verde**: è il colore degli assi; **grigio chiaro**: è il colore del congelamento (del graticolo ENFFD, o della curva reference); **blu**: è il colore dello spostamento; con tale colore è mostrato lo spostamento di un punto 3D, i piani virtuali, il cubo scena..

Come viene gestita l'intuitività nelle applicazioni commerciali? Non si è riscontrato un linguaggio dei colori. Per quanto riguarda le operazoni di trasformazioni elementari la strategia adottata è abbastanza comune e discende direttamente dalla caratteristica altrettanto comune di visionare

l'oggetto attraverso le quattro viste fondamentali. Una rotazione in $z$, per intendersi, equivale ad una rotazione nel piano $XY$, pertanto l'utente deve posizionarsi nella relativa finestra e compiere un movimento intuitivo. Se però il progettista decide di compiere una rotazione direttamente dalla finestra della prospettiva allora deve selezionare dal menu principale la voce Rotation e quindi l'opzione XY Plane. Ancora una volta, questa strategia, ci sembra più scomoda.

## 2.4 Pressioni del mouse e pilotaggio a distanza

Lo schema pseudo-procedurale illustrato sopra risolve abbastanza efficientemente il problema del pilotaggio a distanza. Ciò dovrebbe emergere anche dagli esempi finora trattati: per spostare un control point, l'utente agisce direttamente su di esso, restandone a contatto dall'inizio alla fine dell'azione; per ruotare una superficie egli punta direttamente sull'asse interessato, ecc. Quanto al numero di pressioni del mouse si vede dallo stesso schema procedurale che il prolungamento di una certa azione è determinato dal non cambio del tipo click ossia da un movimento del mouse mantenendo il bottone premuto. In altre parole, la tecnica che è parsa migliore è stata quella del rimpiazzare un continuo *press* del mouse con un suo *drag* (trascinamento). Quando un bottone non è premuto, il mouse è libero di spostarsi senza che il sistema catturi alcun evento. Raramente si notificherà il termine dell'azione con il rilascio del bottone, in quasi tutti i casi ciò sarà implicitamente dichiarato iniziandone una nuova.

Non era, in vero, l'unica scelta strategica che si poteva adottare: molti pacchetti riconoscono l'inizio di una azione con la pressione del tasto e la sua fine con la successiva pressione dello **stesso** tasto. Con questa soluzione l'utente viene esentato dalla fatica di tenere premuto un bottone, tuttavia, da un lato è meno intuitiva (non vi è una netta separazione tra uno stato di comando e uno di **non** comando), da un altro, può indurre un utente distratto a dimenticarsi di notificare la fine dell'azione prima di iniziarne una nuova.

Vi è un inconveniente legato al far riconoscere al sistema (X Window) un evento per ogni movimento del mouse; in generale il buffer degli eventi nè accoderà molti più di quanti ne riuscirà a smaltire, ossia soddisfare. L'effetto è che, soprattutto se l'applicazione è fatta girare in rete, l'utente vede proseguire il comando per un tempo anche molto successivo all'istante in cui ha rilasciato il mouse. La perdita di praticità è devastante. Un modo per ovviare a ciò è invocare una procedura che svuoti la coda degli eventi subito prima di accettarne uno nuovo. Questo è il ruolo che nello schema ha la chiamata

```
svuota_coda_eventi().
```

Capitolo 3

# *FFD* User's Guide

In questa sezione si illustrerà l'intero ambiente di modifica di una superficie progettato e realizzato in questo lavoro. Poichè le funzionalità create sono state davvero molte, è impossibile fornire una immagine illustrativa per ciascuna di esse. Ci si soffermerà pertanto, in modo più approfondito su alcune e meno su altre, lasciando al lettore stesso il compito di intuirne gli effetti pratici e visivi. Va precisato che l'ambiente di modifica in questione, non va a sostituire il suo predecessore (e quindi la finestra di dialogo di 1.2). Infatti data la sua natura e concezione di ambiente privo di interazione di tipo numerico si è convenuto di affiancarlo (e non di sostituirlo) a quello preesistente; *xcsurf* così, non è davvero un sistema incompleto.

## 3.1 Modifica del poliedro di controllo

Quando l'utente clicca col tasto destro del mouse sulla finestra di visualizzazione del poliedro di controllo, in corrispondenza del punto di pressione viene fatto apparire il menu di figura 3.1, del quale per agevolare la trattazione che segue vengono mostrate tutte le icone (mentre in realtà le opzioni in un certo momento non richiamabili sono oscurate).

Seguendo la disposizione dei numerelli e delle lettere poste nella figura 3.1 si procede ora alla spiegazione delle varie funzionalità associate alle icone.

**1A. Comando Control point selection**

Attivando questa icona l'utente può selezionare i control point su cui vuole agire. Egli ha due possibilità:

1. cliccare (bottone 1) sopra un control point, selezionandolo, o deselezionandolo se era stato in precedenza selezionato.
2. disegnare un rettangolo schermo; i control point ivi proiettati vengono selezionati (o deselezionati). Il rettangolo viene definito cliccando su un estremo (bottone 2) e rilasciando il mouse in corrispondenza dell'altro estremo.

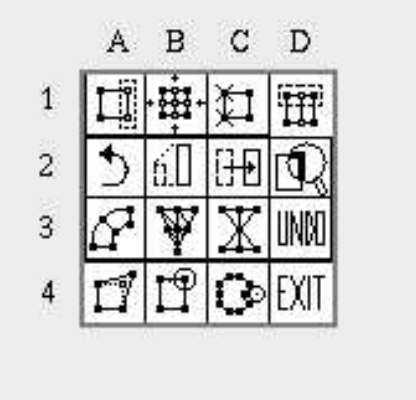


Figura 3.1: Menu dei comandi per la modifica del poliedro di controllo

I control point selezionati sono chiaramente distinguibili dagli altri, per il fatto che i primi sono colorati di rosso, i secondi di giallo.

**1B. Comando Refinement**
Effettua un processo di raffinamento del poliedro di controllo.

**1C. Comando Removal**
Effettua un processo di Knot-Removal in ogni posizione in precedenza interessata a Knot-Insertion. In pratica esegue l'operazione inversa del comando 1B.

**1D. Comando Selective Insertion**
Produce un raffinamento delle parti di poliedro di controllo in cui i control point risultano selezionati. Più precisamente, inserisce un control point nel punto di mezzo di due control point adiacenti entrambi selezionati.

**2A. Comando Control Net Rotation**
Con tale comando, l'utente ruota i punti di controllo selezionati con il comando 1A. Viene disegnato il parallelepipedo che racchiude i control point selezionati; osservando la rotazione assunta da tale parallelepipedo, risulta più chiara l'ampiezza della trasformazione geometrica. Il centro della rotazione (inizialmente posto nel baricentro del parallepipedo suddetto) è reso evidente perchè in corrispondenza ad esso viene disegnato il sistema degli assi della scena. Tale centro può essere cambiato cliccandovi vicino e spostandolo con uno dei sistemi di movimento 3D visti in precedenza (figura 3.2).

**2B. Comando Control Net Scaling**

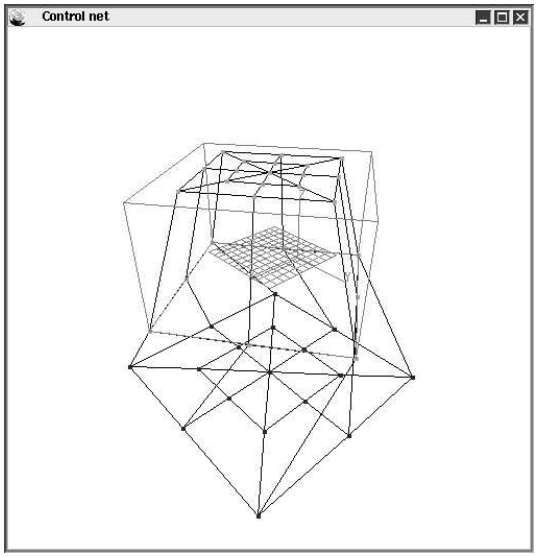


Figura 3.2: Rotazione di più control point e spostamento del centro di rotazione

Realizza la funzionalità di scalatura dei control point selezionati. A parte il tipo di trasformazione geometrica associata, quanto detto per il comando precedente si può qui ripetere.

**2C. Comando Control Net Translation**

Permette la traslazione dei control point selezionati. L'utente clicca in corrispondenza del centro del parallelepipedo contenente tali control point ed effettua uno spostamento in tre dimensioni.

**3A. Comando Control Net Bending**

Realizza la deformazione di Barr omonima (limitandosi ai control point selezionati). L'interattività è la stessa di quella usata per la trasformazione di scala: muovendo il mouse in direzione di un asse proiettato si ottiene l'effetto di curvatura per l'asse medesimo.

**3B. Comando Control Net Twisting**

Realizza la torsione della parte di poliedro di controllo selezionata. L'interazione segue lo schema dell' azione di rotazione.

**3C. Comando Control Net Tapering**

Realizza l'ultima deformazione di Barr. L'interazione segue lo schema del comando di scala.

**4A. Comando Control Point Translation**

Permette lo spostamento di un punto di controllo, azione che è stata al centro di molte discussioni in questo capitolo. L'utente clicca direttamente sul control point desiderato (che quindi non è necessario sia stato selezionato con il comando 1A) e lo muove con uno dei tre sistemi di movimento 3D. Si noti come questa importante operazione di modifica della superficie, sia stata notevolmente facilitata dopo questo lavoro di tesi.

**4B. Comando Weight Changing**

Con tale comando l'utente può modificare il peso di un control point. Il punto di controllo viene selezionato cliccandovi vicino (bottone 1) e il suo peso viene incremetato in senso positivo o negativo a seconda che l'utente sposti il mouse in direzione di un asse positivo o negativo.

**4C. Comando Heat**

Questa funzionalità è stata proposta nel corso del lavoro pensando all'effetto di un oggetto di plastica in presenza ravvicinata di una forte fonte di calore: l'oggetto si ritira. In effetti questo comando realizza la convergenza simultanea di control point (quelli che vengono a contatto con un immaginario disco di fuoco) verso un centro definibile dall'utente (figura 3.3).

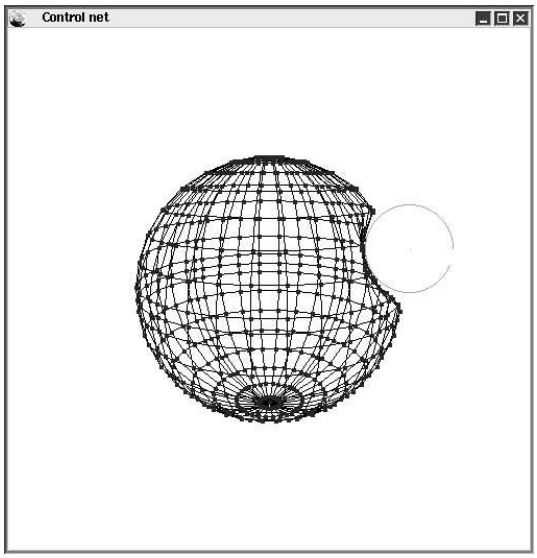


Figura 3.3: Effetto del comando Heat sul poliedro di controllo

L'utente può spostare il disco di fuoco, senza che esso determini alcun effetto, cliccando sul centro del medesimo e tenendo premuto il bottone 1. Tenendo premuto invece, il bottone 2, il disco eserciterà la sua azione di

spinta su tutti i control point che sono in un certo momento a contatto con esso.

La dimensione del disco può essere variata: in questo caso la pressione del bottone 1 deve avvenire in prossimità del bordo del disco e la direzione di movimento del mouse indicherà un suo eventuale allargamento o rimpicciolimento. L'utente, inoltre, può spostare il centro di convergenza (fissato inizialmente nel centro della superficie) cliccando nell'origine del sistema d'assi proiettato e muovendosi in modo 3D come ormai noto.

**2D. Comando Zoom 2D**

A differenza dello zoom menzionato in precedenza (che ingrandisce/rimpicciolisce uniformente la visione della superficie rispetto il centro della scena) questo tipo di zoom permette di ingrandire la visione di una parte soltanto della superficie (o del poliedro di controllo). L'utente delinea la parte interessata disegnando un quadrato sullo schermo. Come per la selezione dei control point, il quadrato viene definito cliccando in corrispondenza di un suo estremo e rilasciando il mouse in corrispondenza dell'altro estremo. La figura 3.4 mostra un utilizzo di tale zoom.

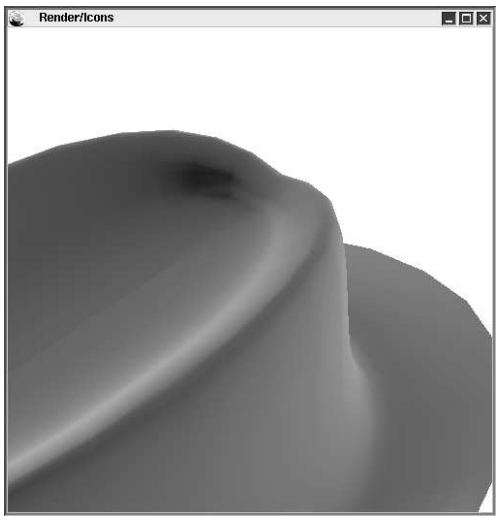


Figura 3.4: Zoom di un particolare di un cappello

**3D. Comando Undo/Redo**

Realizza la ben nota funzionalità di UNDO: la scena viene ripristinata allo stato precedente a quello dell'ultima modifica impartita. Se a seguito di un UNDO si riseleziona questo comando, allora gli effetti dell'UNDO vengono annullati (REDO).

**4D. Comando Exit**

Chiude la sezione di modifica della superficie. Il controllo ritorna agli altri moduli di *xcsurf*.

## 3.2 Modifica della superficie

La pressione del bottone destro del mouse nella finestra per la resa della superficie produce l'apparizione del menu della figura 3.5:

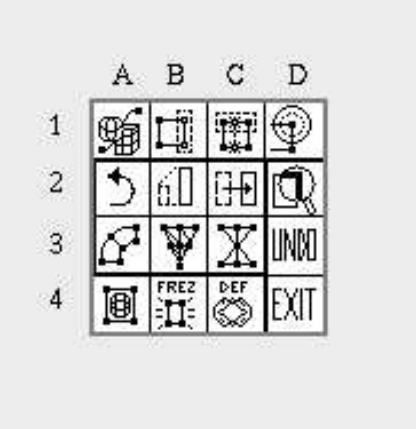


Figura 3.5: Menu dei comandi per la modifica di una superficie

Il comando 1A. (**Object Selection**) è il comando fondamentale di questo menu. Attraverso esso l'utente indica al sistema su quale tipo di oggetto egli intenda lavorare: la superficie (o meglio, la globalità dei suoi punti di controllo), un graticolo ENFFD, una curva. In pratica, come si vedrà nel prossimo paragrafo egli inizierà qualunque procedimento di deformazione attivando questo comando.

I comandi 2A...2D, 3A..3D, 4D, eseguono le analoghe funzioni descritte nel paragrafo precedente, ma applicate appunto, all'oggetto selezionato. Le altre, sono specifiche dei procedimenti di deformazione (e in effetti alcune di esse saranno opportunamente celate).

Si noti la struttura simile dei due menu finora presentati: nella parte centrale (opzioni 2A..2C, 3A..3C) sono collocate le **trasformazioni** elementari e quelle di Barr. Nella prima riga (opzioni 1A..1D) sono collocati i comandi di **selezione e modifica dei control point** (in questo specifico caso sarranno significativi solo per i control point del graticolo ENFFD e delle curve reference e wire). L'ultima riga (opzioni 4A..4C) è riservata alle **azioni deformative o di supporto alla deformazione**.

## 3.3 Modellazione per deformazione

Come già accennato nell'introduzione di questa nota, nel lavoro di tesi di Laurea [Trevisan00] si sono analizzati e sperimentati molti metodi di deformazione proposti in letteratura. Nella fase di progettazione di questo plugin si è deciso di includerne solo alcuni e per l'esattezza i metodi di deformazione di Barr, il metodo Wire Deformation e il metodo ENFFD (Extended NURBS Free Form Deformation); quest'ultimo è una nostra estensione e generalizzazione di quelli proposti in letteratura.
In questa sezione non verranno illustrati tali metodi, per una spiegazione dei quali si rimanda a [Trevisan00] o alla nota [ffdsurvey00]; qui verranno assunti già noti e ci si limiterà a guidare l'utente nel loro utilizzo nell'ambiente realizzato.

## 3.4 Guida ad una deformazione ENFFD

Per prima cosa occorre creare il graticolo dei control point. Per fare ciò si attivi, la già citata opzione 1A del menu rapido precedente. Tale menu viene conseguentemente rimpiazzato dal nuovo menu di figura 3.6.

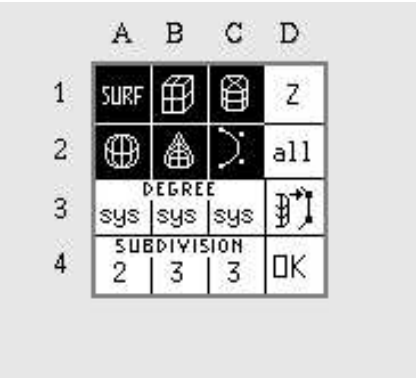


Figura 3.6: Menu di selezione degli oggetti

In questo nuovo menu, le opzioni 1B..1C, 2A..2B, rappresentano i graticoli primitivi da cui si può partire. Le icone mostrano la forma che assumerà la disposizione dei control point del graticolo, non il volume effettivamente descritto, che come noto, dipenderà dalla scelta delle partizioni nodali e dei gradi delle B-spline razionali. Come si vede dalla figura, sono state previste quattro forme primitive: il parallelepipedo, il cilindro, la sfera e il cono.

Si scelga ad esempio, il graticolo a forma di cono.

Occorre adesso, stabilire i gradi delle B-spline. Nella figura, le icone 3A..3C indicano che il volume sarà creato con i gradi e le partizioni nodali stabilite dal **sistema** (da cui la scritta sys), cioè quelle per cui il volume

medesimo è effettivamente un cono. Per stabilire dei gradi diversi basta cliccare col bottone 1 su tali icone. Ovviamente la prima icona rappresenta il grado delle B-spline in $u$, la seconda il grado delle B-spline in $v$, ecc. Ogni click determinerà l'avanzamento di un contatore e il suo valore (cioè il grado) verrà mostrato nell'icona. Cliccando con il bottone 3 tale contatore verrà decrementato. Si lascino, per questo esempio, invariati i valori di tali icone.

Occorre, ora, decidere il numero delle suddivisioni. Per fare ciò si può procedere come nel caso dei gradi, ma cliccando sulle icone 4A..4C. Si stabiliscano ad es. due suddivisioni in direzione $u$, sette in quella $v$, tre in quella $w$.

L'icona 1D stabilisce l'orientamento del graticolo. Si lasci il valore preimpostato, cioè z. Ciò significa che la punta del cono verrà orientata verso l'asse $z$).

Infine si selezioni OK per terminare la fase di caricamento del graticolo.

Il passo successivo consiste nel posizionare il graticolo nella/sulla superficie. Per farlo, basta cliccare su un punto di quest'ultima. Tale punto viene considerato un estremo di un immaginario parallelepipedo che racchiude il cono dei control point. Trascinando il mouse si può muovere l'altra estremità di tale parallelepipedo e conseguentemente determinare il Bounding Box del graticolo ENFFD (figura 3.7)

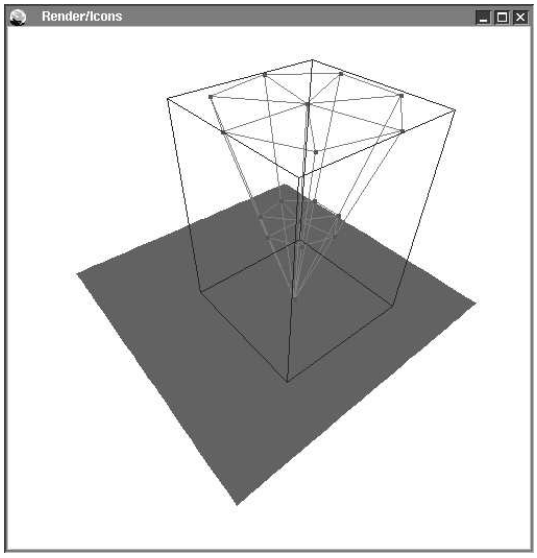


Figura 3.7: Creazione di un graticolo di control point ENFFD

Anche quando si è rilasciato il mouse, si possono modificare le dimensioni del parallelepipedo cliccando in corrispondenza di uno dei suoi vertici e compiendo quindi un nuovo spostamento 3D (ovviamente a seconda del vertice cliccato si modificheranno uno o l'altro estremo del parallelepipedo).

Si modifichi ora il cono in modo da creare il vero graticolo ENFFD iniziale. Attraverso l'opzione 1B del menu di figura 3.5 (che come noto si richiama con un click del bottone 3) si proceda a selezionare i punti di controllo di media altezza (in modo analogo a quanto fatto per i control point del poliedro di controllo). Quindi, si effettui una scala di tali control point (selezionando l'icona 2B del menu).
Si voglia anche, effettuare un raffinamento locale del graticolo: si selezioni una coppia di control point del bordo (solita opzione 1B) e si attivi l'opzione 1C.
In figura 3.8 è illutrato il risultato di quanto detto finora:

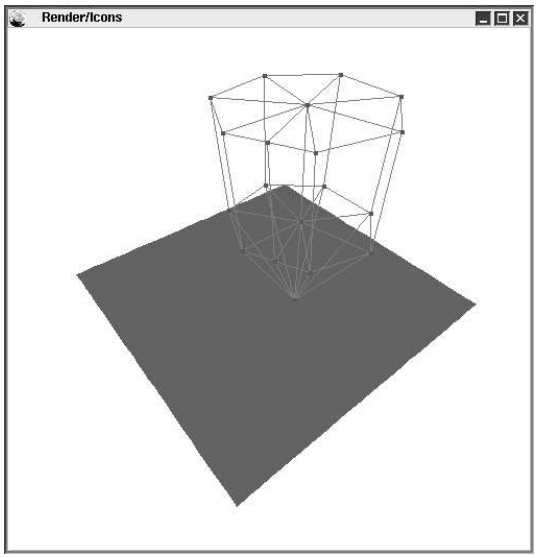


Figura 3.8: Raffinamento e deformazione di un graticolo ENFFD

Si veda ora, quale è il volume effettivamente generato dal graticolo: opzione 4A. Si supponga che il volume sia soddisfacente e quindi si proceda a congelare il graticolo. Questa operazione è fatta selezionando l'icona 4B. Il graticolo diventa grigio. Occorre un piccolo tempo di attesa, al termine del quale il sistema mostra nella finestra dei control point quali di essi sono stati selezionati (cioè quali risultano interni al volume generato).

La prima parte della deformazione è terminata. Ora resta da deformare il graticolo iniziale. Si potrebbe procedere come sopra magari effettuando rotazioni, piuttosto che traslazioni, piuttosto che torsioni ecc.
Invece, si decida di modificare un peso, ad es. di un control point della base del cono. Per fare tale modifica, si selezioni l'icona 1D e si proceda come spiegato a proposito dell'analogo comando riferito al poliedro di controllo della superficie. Il sistema mostra il volume che si sta ora definendo (figura 3.9).

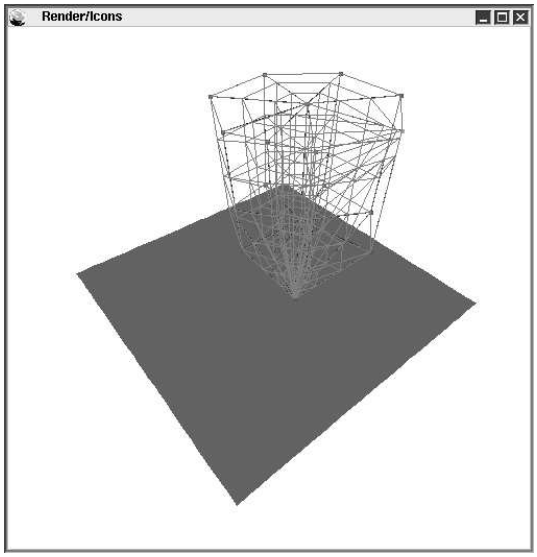


Figura 3.9: Modifica di un peso in un graticolo ENFFD

Può bastare: si supponga che il graticolo sia sufficientemente deformato e si dia il comando di deformazione. L'icona relativa è la 4C.

La nostra esperienza è che, con un pò di pratica, le azioni descritte sopra saranno fatte in modo così spontaneo che un procedimento ENFFD risulterà veloce e, probabilmente, piacevole.

## 3.5 Guida ad una deformazione Wire

La prima cosa da fare è la creazione della curva di riferimento. Come prima, si attivi il menu degli oggetti. Ma in questo caso si scelga l'icona 2C.

Per editare la curva reference si proceda nel modo seguente: si clicchi in corrispondenza di ogni punto della superficie nel quale si voglia posizionare un control point della curva.

Per ogni coppia di control point, il sistema ne genera un altro, esattamente nel punto medio del segmento di estremi i due punti di controllo. Ogni control point creato (dall'utente o dal sistema) è riposizionabile cliccandovi sopra e compiendo uno spostamento 3D.

La figura 3.10 mostra un esempio di creazione della curva di riferimento cliccando su **quattro** punti della superficie. Si noti che i control point sono sette.

Ogni volta che si introduce un nuovo control point o se ne modifica uno, la curva viene aggiornata, cioè rivalutata e resa a schermo.

Si supponga che il risultato soddisfi e si voglia adesso creare la curva wire. Per fare questo occorre congelare la curva di riferimento (l'icona per

analogia, è la stessa prevista per il congelamento del graticolo ENFFD: la 4B). Tale curva diviene, come al solito, grigia.

Ora, per creare la curva wire si hanno due, e non una, strade:

1. cliccare, come sopra, in corrispondenza di un punto della superficie
2. cliccare in corrispondenza di un control point della curva di riferimento.

L'ultima strada serve di fatto a creare inizialmente la wire come copia della reference; in tale caso nessun control point aggiuntivo è creato dal sistema.

Ad es. si copino i primi e gli ultimi due punti di controllo della curva di riferimento, mentre si stabiliscano gli altri in posizioni molto distanti da quest'ultima, come in figura 3.10.

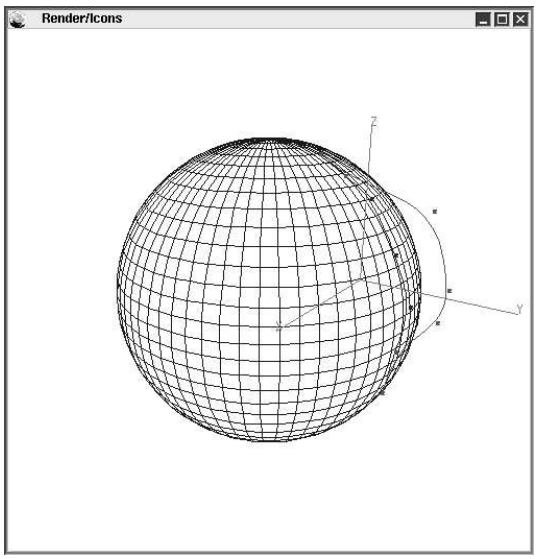


Figura 3.10: Creazione di una coppia reference-wire su una sfera

Non è ancora il momento di procedere con la deformazione. E' opportuno dare una occhiata al volume di influenza associato alla curva iniziale. Si selezioni perciò l'icona view (la 4A, come nel caso ENFFD), e si consideri il poliedro di controllo della superficie: i control point che stanno nel volume di influenza risultano selezionati (figura 3.11).

Per variare tale volume si agisca sul sistema d'assi che l'opzione ha proiettato. Trascinando il mouse nel senso di un asse positivo si ottiene un aumento del raggio di influenza, diversamente, una sua diminuzione. L'insieme dei punti di controllo selezionati viene immediatamente aggiornato e mostrato. Si osservi allora, come l'area colorata nel poliedro si allarga o si restringa al variare del raggio.

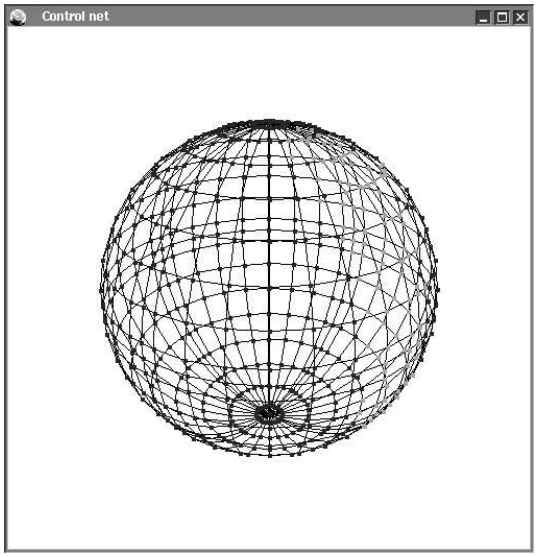


Figura 3.11: Control point selezionati da una curva reference

Si supponga ora che il volume sia soddisfacente, e si proceda con la deformazione (icona 4C).

La deformazione è calcolata considerando un fattore di scala pari a 2 (figura 3.12).

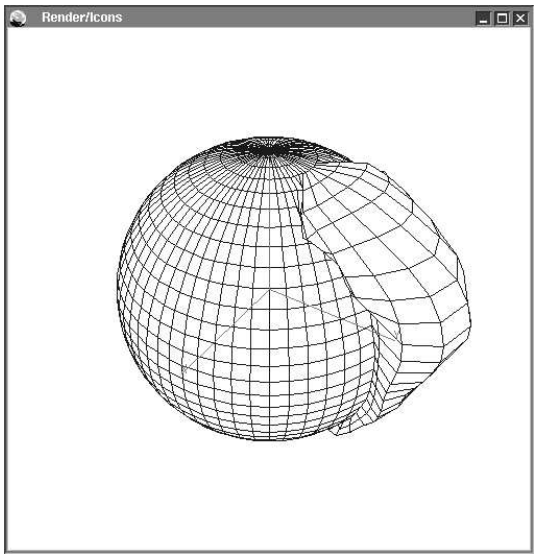


Figura 3.12: Deformazione Wire su una sfera

Si può variare tale coefficiente agendo sul sistema d'assi che viene proiettato nel centro della scena: muovendo il mouse in una direzione positiva [negativa] il fattore viene incrementato [decrementato]. La deformazione è subito ricalcolata e rimostrata.

Alcune osservazioni aggiuntive. Può risultare utile fare in modo che la curva di riferimento combaci con un bordo del poliedro controllore (e non con un bordo della superficie). Per fare questo, occorre attivare l'icona 3D nel menu di caricamento della curva. Attivando tale icona, quando l'utente clicca su un vertice della superficie al control point della reference vengono assegnate le coordinate del control point della superficie più vicino a tale vertice.

Come discusso nel secondo capitolo, creare una curva 3D è assai più scomodo di quanto non sembri. Il motivo sta nel fatto che posizionare dei punti (control point) nello spazio, esattamente come si vorrebbe, è un'impresa non di poco conto. D'altro canto la maggior parte delle volte sia la curva reference che quella wire sono nell'intenzione dell'utente, curve 2D, cioè stanno in un piano. Sarebbe pertanto comodo disporre di una funzionalità che limitasse gli spostamenti dei control point in un piano, agevolando di molto la facilità dell'editazione della curva. In *xcsurf* si è pensato anche a questo: attraverso l'icona 2D del menu degli oggetti (la quale, nella figura 3.6 riporta la dicitura all) può essere disabilitato il movimento 3D in una direzione. L'icona 2D può essere al riguardo settata a -x, -y, -z, all; quest' ultimo settaggio indica l'abilitazione di tutte le direzioni di spostamento; il segno - indica l'asse per cui disabilitare il movimento. Allineando opportunamente la superficie in un piano fondamentale e disabilitando gli spostamenti nell'altra direzione, creare una curva diventa una operazione abbastanza semplice.

Probabilmente, non si sono risolti tutti i disagi di una modellazione di curve, disagi che avevano prodotto grossi dubbi sulla facilità d'utilizzo dei metodi di deformazione basati su tale strategia, ma sicuramente, anche grazie l'ultima opzione spiegata, se ne sono risolti molti.

# Bibliografia

[ffdguide00] G.Casciola, E.Trevisan. *Free Form Deformation: xcsurf (Version2.0) plugin*. Department of Mathematics, University of Bologna, 2000.

[ffdsurvey00] G.Casciola, E.Trevisan. *FFD survey and ENFFD method*. Department of Mathematics, University of Bologna, 2000.

[Trevisan00] E.Trevisan, Modellazione per deformazione di superfici NURBS, Tesi di Laurea in Scienze dell'Informazione, Università di Bologna, (2000)

[xcsurf05] G.Casciola. *xcsurf: the 3D modeller, User's Guide - Version 3.0*, (2005)
http://www.dm.unibo.it/∼casciola/html/xcmodel.html