METODOLOGIA E CRITERI DI PROGETTO PER UN SISTEMA GRAFICO STANDARD

Giorgio Faconti

Rapporto C83-15

Istituto CNUCE
PISA
Settembre 1983

1. INTRODUZIONE

L'importanza dei sistemi grafici risiede essenzialmente nelle possibilita', che essi offrono, di comunicare con un sistema di elaborazione attraverso l'utilizzo di informazioni grafiche. A partire dai primi studi relativi alla computer graphics e' stato infatti messo in evidenza come l'utilizzo di tecniche grafiche migliori enormemente la qualita' dell'interazione tra uomo e macchina.

Non appena una nuova tecnologia, come i dispositivi grafici, diventa disponibile per un grande numero di utilizzatori, si stabiliscono automaticamente delle regole minime, generalmente accettate, relative al suo utilizzo. Tuttavia, per trarre il maggior beneficio possibile dallo stabilirsi di queste convenzioni, e' necessario trovare delle definizioni piu' precise per quanto riguarda i dispositivi hardware, i sistemi software ed i protocolli di comunicazione.

Nella sostanza e' questo lo scopo ed il compito di qualsiasi operazione di standardizzazione.

Nel campo della computer graphics, le prime attivita' in questo senso sono state condotte verso la meta' degli anni '70 dal Graphics Standard Planning Committee in seno all'ACM/SIGGRAPH (Special Interest Group - Computer Graphics) e dal Graphics Subcommittee dell'IFIP Working Group 5.2.

Le attivita' dei due gruppi si sono vicendevolmente influenzate e si sono concretizzate nello Workshop on Computer Graphics di Seillac del 1975, e nello 'Status Report of the Graphics Standard Planning Committee, Siggraph of ACM' pubblicato nel 1977 dalla rivista Computer Graphics. Questo rapporto consiste sostanzialmente in una bozza di proposta da adottare per lo sviluppo di uno standard grafico ed e' conosciuta con il nome di CORE Graphics System. Gli ulteriori sviluppi di questa proposta non hanno modificato i concetti di base sviluppati inizialmente in quanto l'unica, seppure importante estensione della proposta stessa, e' stata l'integrazione nel sistema delle possibilita' fornite dai dispositivi grafici di tipo 'raster scan', che pur imponendo innovazioni funzionali non intaccano le scelte metodologiche.

Fin dalla sua prima comparsa, il CORE e' stato diversamente interpretato. Nell'ambito dell'insegnamento e' stato assunto come base dei corsi di computer graphics, in alcuni ambienti industriali e' diventato 'de facto' lo standard, per alcuni doveva essere il prototipo da utilizzare per la definizione di uno standard ANSI, per altri infine era un veicolo per stimolare la discussione e per uno studio di fattibilita'.

Una possibile spiegazione di questo fatto e' che a seguito dello Workshop di Seillac, alcuni si sono dedicati allo studio di una metologia per lo sviluppo dei sistemi grafici mentre altri hanno pensato allo sviluppo di uno standard. Il risultato e' stato che lo sviluppo dello standard ha anticipato, oscurandolo, quello che veniva indagato a livello metodologico.

Nel seguito del presente articolo vengono discusse le scelte metodologiche e i criteri di progetto del CORE Graphics Systems evidenziando, per quanto possibile, i problemi che sorgono nel momento in cui si passa da una definizione teorica alla realizzazione pratica.

2. MODELLING SYSTEM E VIEWING SYSTEM

La questione piu' importante da risolvere nella definizione di uno standard e' la definizione della materia stessa oggetto di standardizzazione: che cosa si deve includere od escludere dallo standard.

Il problema e' stato risolto per il CORE adottando la risoluzione di Seillac riguardante la distinzione tra 'modelling system' e 'viewing system'.
Il modelling system permette la creazione di oggetti nello spazio di coordinate in cui essi sono definiti e fornisce le funzioni per definire trasformazioni e combinazioni di quegli oggetti in quello spazio. Il risultato consiste tipicamente in una serie di informazioni geometriche definite in un 'sistema di coordinate reali' tali da poter essere interpretate dal viewing system.
Il viewing system consiste essenzialmente in una sequenza di trasformazioni che applicate alle entita' geometriche definite a livello modelling ne consentono la visualizzazione su di un dispositivo grafico; esso include funzioni per disegnare linee e testi, per la definizione delle trasformazioni di visualizzazione, per la gestione dell'input, e funzioni di controllo per la gestione del sistema nel suo insieme.

Il modello di un oggetto e' per sua natura una entita' astratta che viene rappresentata attraverso un modello logico costituito da una struttura dati e da un insieme di funzioni che interpretano quella struttura; in questo senso e' necessario che differenti applicazioni possano utilizzare differenti modelling systems, mentre e' comunque possibile definire un esempio generale di viewing system indipendente dall'applicazione stessa.

Una parte dei dati contenuti nella struttura di un modello contiene informazioni geometriche che possono essere interpretate da un programma per produrre grafici; sia che si tratti, ad esempio, di mappe territoriali, di circuiti integrati o di parti meccaniche. Il rimanente della

struttura contiene dati relativi ad altri tipi di informazioni quali potrebbero essere la distribuzione della popolazione, alcuni valori di resistenza o il carico di rottura negli esempi sopra citati.
In ogni caso la struttura dati che rappresenta il modello spesso si sovrappone alle strutture dati che sono parte integrale della rappresentazione grafica di un oggetto.

Questa divisione concettuale ha avuto una grandissima influenza nello sviluppo e nell'accettazione degli standards ed e' ancora oggi oggetto di alcune controversie, sebbene la tendenza ormai universalmente accettata sia quella di considerare i modelling systems di pertinenza del campo dell'intelligenza artificiale, dei sistemi per la rappresentazione della conoscenza e dei sistemi inferenziali.

3. METODOLOGIA

Il motivo piu' importante che giustifica lo sviluppo di una metodologia standard per i sistemi grafici e' la nozione di portabilita' dei programmi; cioe' la possibilita' di trasportare applicazioni grafiche da un'installazione ad un'altra con il minimo possibile dei cambiamenti.

La portabilita' di un'applicazione grafica e' profondamente legata ai concetti di 'device-indipendence' e di 'machine-indipendence'; cioe' alla possibilita' di un sistema software di poter essere utilizzato unitamente a terminali grafici diversi e di poter essere eseguito su diversi sistemi di elaborazione in quanto accessibile da linguaggi di programmazione comunemente disponibili su diverse macchine.

In realta', si puo' ragionevolmente ritenere che la portabilita' di un programma in senso assoluto non possa essere raggiunta, per cui questo concetto deve essere necessariamente sostituito con una sua qualche approssimazione.

3.1 Indipendenza dalle periferiche grafiche

Il primo dei problemi legati alla portabilita' e' dovuto all'esistenza di una varieta' estremamente grande di periferiche grafiche ed alla diversita' di funzioni che tali dispositivi possono o meno possedere. Ogni dispositivo grafico tende non soltanto a fornire un insieme diverso di funzioni, ma molto spesso richiede anche l'uso di speciali tecniche di programmazione per accedere a queste funzioni.

Il CORE Graphics System introduce, a soluzione di questo problema, i concetti di 'superficie di visualizzazione virtuale' e di 'dispositivi logici di input'. La superficie di visualizzazione virtuale e' una superficie di output

logica definita in un 'sistema di coordinate normalizzate' su cui sono logicamente memorizzati gli oggetti a cui siano state applicate le trasformazioni di visualizzazione. Al momento della presentazione su di una superficie grafica reale, gli oggetti presenti sulla superficie logica subiscono una trasformazione da coordinate normalizzate a 'coordinate di schermo'.
Analogamente, i 'dispositivi di input logici' utilizzati per l'introduzione dei dati, sono associati ai dispositivi reali soltanto al momento dell'esecuzione di un'applicazione.
La corrispondenza tra dispositivi di input logici, superfici di output virtuali e dispositivi fisici in una data configurazione, e' gestita completamente dal sistema grafico e non richiede alcun intervento da parte del programma applicativo.

In tal modo viene creata una distinzione logica tra la parte indipendente e la parte dipendente dal particolare tipo di dispositivi grafici al momento utilizzati, per cui il sistema risulta distinto in due moduli: uno device-independent e di validita' generale e l'altro device-dependent e valido per un particolare tipo di hardware, il cosidetto 'device driver'.
Allo stesso modulo device-independent possono essere collegati vari device drivers per pilotare diversi tipi di periferiche grafiche.

Per i dispositivi logici sono definite una serie di funzioni che non sempre trovano riscontro nei dispositivi reali e che difficilmente possono essere simulate su qualsiasi periferica grafica. Il CORE Graphics System e', a questo proposito, prevalentemente orientato verso displays grafici interattivi di tipo vettoriale e di prestazioni medie.

3.2 Indipendenza dai linguaggi.

Generalmente, anche i linguaggi di programmazione standard presentano differenze, seppur minime, nel passare da un tipo di elaboratore ad un altro: non esiste quindi nessun linguaggio per cui sia assicurata una assoluta portabilita'. Questo significa che un'applicazione necessita sempre di qualche modifica quando i programmi relativi sono trasferiti da un sistema di elaborazione ad uno diverso.

Gli interventi a livello di programma sorgente necessari per permettere la portabilita' di un'applicazione possono essere cosi' distinti:

- semplici modifiche nella chiamata di alcune routines, come ad esempio sostituzione dei nomi delle routines stesse (LINE(...) per DRAWTO(...), ecc.), che possono essere effettuate da un programmatore non esperto del problema

applicativo e per le quali e' richiesta in genere soltanto la disponibilita' di un buon Text-Editor.

- modifiche alla struttura del programma che richiedono una conoscenza approfondita sia dell'applicazione che della sua implementazione particolare per cui sono molto onerose e non sempre esenti da errori.

E' ovvio che nello sviluppo di un'applicazione occorre fare il massimo sforzo affinche' tali modifiche ricadano nella prima delle categorie citate e che il punto focale del problema, dal punto di vista dello standard, sia quello della definizione della struttura dei sistemi grafici, il cui studio porta conseguentemente alla comprensione delle funzioni hardware e software necessarie per lo sviluppo delle applicazioni.

Il CORE Graphics System e' inteso, a questo proposito, piu' come un sistema chiuso ed autosufficiente che non come un insieme di routines da usare come base per l'implementazione di packages ad alto livello piu' adatti per lo sviluppo di applicazioni.
Esso e' accessibile da linguaggi ad alto livello come FORTRAN o PL/I, ma non e' necessariamente implementato utilizzando questi linguaggi. Inoltre, laddove sia reso disponibile in ambienti naturalmente interattivi (APL, CMS, UNIX, ...), esso esiste ed e' funzionale indipendentemente da qualsiasi applicazione, presentandosi come un'estensione del sistema ospite con cui l'utente puo' interagire in modo nativo.

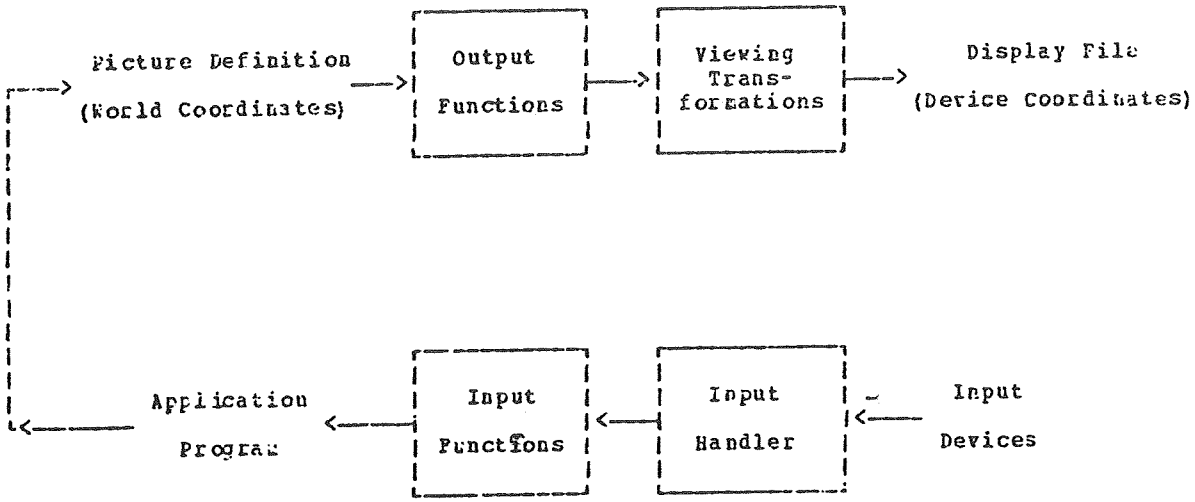
3.3 Indipendenza dai sistemi operativi e dalle macchine.

Il sistema di elaborazione influenza l'implementazione e l'utilizzo del CORE Graphics System sotto due aspetti: i sistemi operativi ed i linguaggi di programmazione. Restando valido quanto gia' affermato per i linguaggi di programmazione in generale, e' comunque necessario che tutte le funzioni del CORE siano implementate in modo tale da non richiedere l'utilizzo di particolari capacita' non standard dei sistemi operativi ospiti.
Ad esempio, non esiste alcuna funzione del CORE che richieda al sistema operativo di caricare dinamicamente in memoria parti di programma, sebbene questo sarebbe indubbiamente molto utile. La gestione della memoria ed altri tipi di funzioni dipendenti dal linguaggio devono essere gestiti utilizzando quelle regole che sono stabilite per i singoli linguaggi.

3.4 Il modello del sistema grafico

Il CORE Graphics System si presenta all'utilizzatore con un modello funzionale estremamente semplice e coerente che puo' essere descritto nei seguenti punti:

- la separazione delle funzioni di input e di output
- la minimizzazione delle differenze nell'utilizzo di diversi tipi di periferiche grafiche
- il concetto di due sistemi di coordinate: il sistema di coordinate reali in cui sono definiti gli oggetti da visualizzare ed il sistema di coordinate del dispositivo grafico in cui gli oggetti stessi sono rappresentati per la visualizzazione
- la nozione di 'display file'
- la nozione di 'segmento'
- la nozione di trasformazioni di visualizzazione.



In figura e' rappresentata la struttura generale di un sistema grafico interattivo costruito attorno a questi concetti. Il programma applicativo crea la definizione degli oggetti da visualizzare, eventualmente memorizzata in una struttura dati propria dell'applicazione stessa, utilizzando un sistema di coordinate reali; a tale definizione sono quindi applicate le trasformazioni di visualizzazione, che attraverso meccanismi hardware o software, ne creano una rappresentazione nel sistema di coordinate del dispositivo grafico e la memorizzano nel display file sotto forma di segmento. I singoli segmenti che costituiscono un'immagine possono in tal modo essere modificati senza necessariamente dover ripercorrere tutte le operazioni di trasformazione. Inoltre i dati di input del dispositivo grafico sono interpretati dal programma applicativo per definire nuovi oggetti da rappresentare o per cambiare l'aspetto di oggetti precedentemente definiti ripercorrendo tutta la sequenza delle operazioni di output.

3.5 Livelli di standard

Il modello del sistema grafico puo' essere considerato generalmente valido per qualsiasi applicazione interattiva, ma in particolari circostanze, diventa insoddisfacente sia perche' puo' non prevedere possibilita' che sarebbero necessarie in un particolare contesto sia perche' le troppe funzioni disponibili ne possono degradare indirettamente le prestazioni anche se non sono utilizzate. Ad esempio, in alcuni casi si puo' avere necessita' di gestire in modo efficiente il ricorrere dello stesso elemento in un complessivo, oppure nel caso tipico di utilizzo di un plotter, si ha la necessita' di dedicare una buona parte di risorse alla gestione di un display file.
In ogni caso, le necessita' espresse dalle diverse applicazioni sono talmente varie, che difficilmente la proposta di un unico sistema standard sarebbe in grado di soddisfarne i requisiti.

Questo problema e' di fondamentale importanza nella definizione di uno standard e certamente di non facile soluzione se si considera nella sua interezza. In pratica, e' possibile trovare una soluzione del tutto generale adottando una struttura modulare che preveda diversi livelli di funzionalita'. Al livello piu' elevato tutte le funzioni devono essere disponibili, ma scendendo a livelli piu' bassi un numero diverso di funzioni puo' non essere supportato.

Nel CORE la divisione in livelli riflette la separazione concettuale di funzioni di input e di output, e di spazi di coordinate a due ed a tre dimensioni; all'interno di ogni classe di funzioni sono poi identificabili dei sottolivelli che rendono il sistema piu' o meno ricco.

4. CRITERI DI PROGETTAZIONE

Il CORE Graphics System, in accordo con i criteri metodologici esposti, e' un sistema ricco e chiuso in se stesso, utilizzabile da un gran numero di applicazioni grafiche su di un gran numero di dispositivi grafici. Esso e' accessibile da linguaggi di programmazione ad alto livello e la sua interfaccia verso l'utente e' indipendente dai dispositivi grafici, dagli elaboratori e dai sistemi operativi.

Il raggiungimento di questi obiettivi e' basato sull'accettazione del compromesso de 'il piu': 'quello che e' utile per la maggior parte dei programmatori sulla maggior parte dei dispositivi grafici interattivi per la maggior parte delle applicazioni il piu' delle volte'.
Sulla base di questo compromesso, e' da ritenersi che un criterio per la valutazione delle possibilita' di un sistema grafico sia quello di considerare 'generalmente utili'

quelle funzioni che non 'costano' troppo e non presentano difficolta' nella loro implementazione.

La strategia della suddivisione del CORE in livelli e' stata opportunamente studiate per evitare che questo criterio sia interpretato in modo tale che il sistema diventi prolisso invece che ricco. Le prove condotte per verificare l'implementabilita' del CORE confermano infatti che esso deve contenere soltanto quelle funzioni logiche che possono essere facilmente supportate dalla maggior parte dei dispositivi grafici esistenti e che non possono essere implementate utilizzando altre funzioni del sistema stesso.

4.1 Effetti sulla struttura dei programmi applicativi

In accordo con quanto convenuto a Seillac, lo sviluppo del CORE e' stato effettuato concendrando l'attenzione su quelle risoluzioni che influenzano la struttura dei programmi applicativi. Nello stesso tempo tuttavia, si e' cercato di raggiungere come scopo collaterale quello di fornire una guida anche in altre aree dove e' necessario risolvere problemi tecnici e prendere decisioni, come nel caso della gestione dei testi. All'interno delle linee di guida circa le funzioni da implementare nel sistema e' stato fatto un tentativo di fornire un dettaglio sufficiente delle loro specifiche funzionali e di concentrarsi su quelle risoluzioni particolarmente importanti o controverse.

4.2 Pratica di utilizzo

Alcune funzioni, per quanto potenzialmente desiderabili, non sono completamente accettate nella pratica di utilizzo dei sistemi grafici e non sono state per questo prese in considerazione. Uno standard, in linea di principio, non deve spingersi troppo oltre rispetto allo stato dell'arte in materia: non possono ad esempio essere incluse funzioni non ancora ben conosciute ed investigate.

4.3 Inclusione parziale di funzioni

Qualsiasi funzione presente nel sistema deve essere inclusa od escusa nella sua interezza per evitare di dover definire regole complesse sull'applicabilita' delle funzioni stesse. Un'inclusione parziale di alcune funzioni conduce all'accettazione di compromessi spesso insoddisfacenti e ad implemetazioni che mancano della necessaria chiarezza. A questo proposito, e' stata escusa l'ipotesi della definizione di un numero ristretto di funzioni per la gestione dei testi in quanto avrebbe introdotto una serie di interazioni difficili da realizzare con altre funzioni ed avrebbe di fatto inibito lo sviluppo futuro di un modulo completo.

4.4 Chiarezza e solidita'

E' frequentemente possibile identificare insiemi diversi di funzioni che rendono possibile il raggiungimento dello stesso risultato. Un criterio per scegliere l'insieme di funzioni da implemetare e' quello che tali funzioni siano ben strutturate e che gli effetti prodotti siano quanto piu' ovvi possibile: ad esempio, ogni inizializzazione deve necessariamente avere una terminazione esplicita, gli effetti nascosti od impliciti devono essere evitati, casi speciali ed eccezioni non sono ammessi.

4.5 Ortogonalita' e assenza di interazioni

Il principio dell'ortogonalita' e' stato utilizzato innanzitutto per separare le attivita' di modelling e di viewing, ma si trova applicato anche in un gran numero di casi all'interno del CORE stesso: separazione di input ed output, di attributi dei segmenti ed attributi primitivi, ecc. L'ortogonalita' permette di definire le funzioni indipendentemente le una dalle altre, una volta constatato che non esistono inferenze ed aiuta a chiarire proprio i problemi relativi alla possibilita' di interazioni tra funzioni diverse.

4.6 Completezza e consistenza

Un'opzione che e' valida per un elemento di una classe di funzioni deve essere valido per tutti gli elementi appartenenti a quella classe. Ad esempio, il meccanismo di definizione del valore di un tipo di attributo primitivo deve essere applicabile a tutti gli attributi primitivi e se il valore di uno di essi puo' essere ottenuto attraverso una richiesta questo deve accadere anche per tutti gli altri.

5. SPECIFICHE FUNZIONALI

Come gia' accennato nei paragrafi precedenti, la struttura del CORE Graphics System e' basata sulla separazione delle funzioni nelle classi di controllo, output ed input e sulla definizione di livelli funzionali ottenuti combinando in vari modi le possibilita' offerte da ciascuna classe.

5.1 Funzioni di Controllo

L'ambiente operativo del CORE e' controllato da alcune funzioni che definiscono lo stato del sistema stesso: inizializzazione del sistema ad un dato livello, inizializzazione e selezione delle superfici di visualizzazione,

definizione dei valori per gli attributi relativi ai segmenti ed alle primitive di output, ecc. Fanno inoltre parte di questa classe tutte le funzioni per richiedere informazioni circa lo stato corrente del sistema, i valori delle variabili e le possibilita' dei dispositivi di output. Da ultimo e' possibile definire una particolare modalita' di esecuzione definita 'batch of updates' che permette di eseguire una seguenza di operazioni visualizzando sulla superfici di output soltanto il risultato finale senza passare attraverso tutti i risultati intermedi.

5.2 Funzioni di Output

5.2.1 Primitive di output

Il 'mondo grafico' che l'applicazione descrive al CORE consiste di uno o piu' oggetti, definiti in un sistema di coordinate reali (del mondo), che vengono rappresentati mediante l'utilizzo delle funzioni 'primitive di output'. Queste funzioni descrivono essenzialmente i movimenti che farebbe la punta di una matita per disegnare su di un foglio di carta l'oggetto o gli oggetti in esame: tali movimenti producono quindi spostamenti, linee, seguenze di linee, stringhe di caratteri (testi) ed identificatori che vengono definiti come 'primitive di output'. Ad ogni primitiva di output vengono associati i valori correnti degli 'attributi delle primitive di output' che ne definiscono l'aspetto in termini di colore, stile di linea, spessore del tratto, intensita' ed ancora, tipo di alfabeto, grandezza, spaziatura e qualita' dei testi. Ogni primitiva di output puo' inoltre essere associata ad un particolare identificatore (pick) che ne permette il riconoscimento attraverso i dispositivi di input grafico.

5.2.2 Segmentazione

Tutte le primitive di output che reppresentano un oggetto sono memorizzate dal CORE in un'entita' specificata dall'applicazione detta 'segmento': ogni segmento rappresenta una 'immagine', cioe' una vista dell'oggetto, che e' parte del 'complessivo' presente sulla superficie di output.

Ogni segmento deve essere specificato dal programma applicativo ma la sua creazione e gestione sono completamente demandate al sistema grafico che provvede a 'riempirlo' con le primitive di output che vengono invocate dall'applicazione finche' esso resta 'aperto'. Al momento in cui viene invocata la funzione che provoca la sua 'chiusura', il segmento viene definito come un'entita' unica e non e' piu' modificabile al suo interno.

All'interno del sistema grafico possono essere definiti due tipi di segmenti:

- i segmenti 'temporanei' la cui caratteristica e' quella di esistere esclusivamente al momento della loro creazione e di cui non viene memorizzata alcuna informazione per possibili utilizzi futuri

- i segmenti 'retained' per i quali vengono memorizzate dal sistema grafico tutte le informazioni in essi contenute e che possono quindi essere utilizzati successivamente piu' volte nel corso dell'esecuzione di un'applicazione.

Come per le primitive di output, anche per i segmenti possono essere definiti alcuni 'attributi' che hanno la caratteristica della dinamicita' nel senso che il valore di un attributo puo' essere modificato piu' volte nel corso dell'esecuzione di un'applicazione: gli attributi 'visibilita'' ed 'evidenziazione' sono usati per controllare l'aspetto o l'apparenza del segmento, mentre l'attributo 'identificabilita'' ne rende possibile il ritrovamento sulla superficie di output attraverso le primitive di input.

In aggiunta a questi attributi, esiste ai livelli piu' elevati del CORE la possibilita' di definire per un segmento la 'trasformazione di immagine' che permette di applicare trasformazioni topologiche sul segmento stesso nello spazio di coordinate della superficie di output reale.

5.2.3 Trasformazioni di visualizzazione

La separazione delle funzioni di modelling da quelle di viewing, fornisce una comoda analogia per la comprensione dei meccanismi propri delle trasformazioni di visualizzazione: quella della macchina fotografica.

Per la rappresentazione di oggetti in uno spazio a due dimensioni le trasformazioni di visualizzazione sono definite da una 'window' espressa in coordinate reali e da un 'viewport' sulla superficie di output attualmente selezionata. Nel caso di oggetti definiti in uno spazio a tre dimensioni, oltre alla window ed al viewport, e' necessario definire il 'piano di proiezione', il 'tipo di proiezione' (parallela o prospettica) e la 'direzione di proiezione' od il 'punto di vista' a seconda del tipo di proiezione scelto.

In entrambi i casi si puo' considerare che la window sia la porzione di scena inquadrabile da una macchina fotografica, il viewport sia il supporto fisico su cui verra' stampata la fotografia, e che gli altri parametri definiscano la posizione e l'orientazione della macchina nella scena in esame.

5.3 Funzioni di Input

Il CORE Graphics System riconosce cinque classi di dispositivi logici di input, che sono modellati sui dispositivi fisici di input attualmente disponibili nella maggior parte delle periferiche grafiche interattive. Le funzioni svolte da tali dispositivi possono essere cosi' sommarizzate:

- Pick (penna luminosa) identifica una primitiva di output attraverso il nome del segmento cui tale primitiva appartiene, e l'attributo primitivo di identificazione associato alla primitiva stessa.
- Locator (tablet) fornisce la posizione di un punto su di una superficie di output nello spazio di coordinate di tale superficie
- Valuator (interruttore) fornisce un valore scalare
- Button (tastiera funzionale) seleziona una possibilita' tra diverse alternative
- Keyboard (tastiera) permette l'introduzione di stringhe di caratteri.

I dispositivi di tipo pick, keyboard e button generano 'eventi' che sono memorizzati in una 'coda di eventi' interrogabile da parte dell'applicazione. I dispositivi di tipo valuator e locator, viceversa, non causano eventi; e' pero' possibile stabilire un'associazione tra i due tipi di dispositivi in modo che quando si verifica un evento i valori di locators e valuators siano inclusi essi stessi nella coda di eventi.

6. CONCLUSIONI

Tornando sulla questione posta all'inizio di questo articolo sull'interpretazione della proposta CORE, si puo' qui affermare che lo scopo del GSPC e' stato quello di proporre una metodologia e di mostrare come essa possa essere applicata per lo sviluppo di un sistema grafico standard. E' infatti necessario per una chiara comprensione della proposta poter conoscere sia i principi metodologici che l'hanno ispirata sia i criteri di progetto che su questi principi sono basati. Senza la conoscenza dei principi metodologici difficilmente si possono correggere eventuali errori commessi nella progettazione, e senza la conoscenza dell'architettura del sistema e' certamente arduo comprendere come questi principi sono stati considerati; viceversa, entrambe le cose forniscono l'opportunita' di 'vedere' la metodologia in azione.

REFERENZE

[GUED76] Guedj, R.A. et. al., Report on the IFIP W.G. 5.2 Workshop on "Methodology in Computer Graphics" Luglio 1976

[GSPC77] Status Report of the Graphics Standard Planning Committee Siggraph of ACM Computer Graphics, 11(3), Agosto 1977

[BERG78] Bergeron, D. et.al., Graphics Programming Using the CORE System Computing Surveys, 10(4), 1978

[MICH78a] Michener, J. and J.D. Foley, Some Major Issues in the Design of the CORE Graphics System Computing Surveys, 10(4), 1978

[MICH78b] Michener, J. and A. Vam Dam, A Functional Overview of the CORE System with Glossary Computing Surveys, 10(4), 1978

[NEWM78] Newman, W.M. and A. Vam Dam, A Brief History of Efforts towards Graphics Standardization Computing Surveys, 10(4), 1978

[WARN78] Warner, J. and M. Polisher, DIGRAF - A FORTRAN Implementation of the Proposed GSPC Standard Computer Graphics, 12(3), Agosto 1978

[GSPC79] Status Report of the Graphics Standard Planning Committee Computer Graphics, 13(3), Agosto 1979

[FOLE81] Foley, J.D. and P. Wenner, The George Washington University CORE System Implementation Computer Graphics, 15(3), Agosto 1981

[FACO82a] Faconti, G. e P. Palamidese, Un approccio APL alla realizzazione di un sistema grafico di base CNUCE C82-3, Febbraio 1982.

[FACO82b] Faconti, G. e P. Palamidese, Practicing on the CORE Graphics System APL82 Conference, Heidelberg, Luglio 1982.