C a p i t o l u l 1

C o n c e p t e g e n e r a l e

1.1 Domenii de aplicaţie ale graficii pe calculator

Sistemele grafice au devenit după anii 1980, obiect de lucru al inginerilor, oamenilor de ştiinţă, artiştilor plastici şi arhitecţilor. S-au construit o serie de aplicaţii grafice precum biblioteci de subrutine grafice şi interfeţe cu utilizatorul uşor de manipulat, cu facilităţi de acces la diverse resurse prin intermediul reţelelor.

Domeniile actuale de aplicaţie ale graficii pe calculator sunt următoarele:

1. Educaţie şi învăţământ. Reprezentările grafice au o forţă sugestivă deosebită. Rezultate semnificative se pot obţine în domeniul predării matematicii (în special în geometrie şi analiză matematică), fizicii, chimiei şi altor ştiinţe.
2. Cercetare ştiinţifică. În special în faza de prezentare a rezultatelor obţinute prin prelucrarea datelor experimentale, grafica pe calculator este un instrument preţios al cercetătorului în modelarea teoretică a unor fenomene, cu precădere în chimie, fizică şi biologie.
3. Inginerie. În cadrul pachetelor de programe CAD (Computer Aided Design – proiectare asistată), CAM (Computer Aided Machinning – fabricaţie asistată), CAE (Computer Aided Engineering – inginerie asistată), majoritatea firmelor propun pre– şi post– procesoare grafice care furnizează imagini de o înaltă calitate. Principalii beneficiari sunt ingineria mecanică, electronica şi arhitectura. Marii producători de material aeronautic, naval, auto şi feroviar au angajat mari resurse în programe proprii CAE.
4. Economie, conducere: în realizarea de grafice, histograme, diagrame care facilitează procesele de prezentare a datelor, planificarea activităţilor şi cheltuielilor, previziune, analiză.
5. Simulatoare: de exemplu, simulatoare de zbor în timp real.
6. Cartografie, meteorologie, prospectarea resurselor.

7. Artă. Calitatea imaginilor create pe calculator a permis inserarea acestora în numeroase producţii cinematografice (de exemplu, "Jurasik Park"). Alte aplicaţii vizează realizarea de imagini artistice realiste sau abstracte, studii pentru sculptură, facilităţi pentru analiza şi compoziţia muzicală.
8. Comerţ, reclamă. Graficele sunt utilizate foarte des pentru prezentarea sintetică a unor informaţii cu volum mare de tip financiar, statistic, matematic, sau economic.
9. Comanda şi urmărirea proceselor. Reprezentările grafice ale stărilor proceselor reale sunt ajutoare preţioase în procesul decizional, în special în situaţii critice şi în cazul în care numărul parametrilor de supravegheat este mare. Monitorizarea centralizată a proceselor se realizează în special în industria petrochimică, în metalurgie, în energetică.
10. Recreere: jocuri video pe caculator.

Scopurile aplicaţiilor grafice în domeniile mai sus menţionate pot fi clasificate astfel:

- afişarea informaţiilor (desen tehnic, imagini medicale, vizualizare ştiinţifică, biologie moleculară, pentru interpretarea datelor multiple);
- proiectare (o problemă principală în design este existenţa soluţiilor multiple. Proiectantul examinează un desen potenţial pe care îl modifică în ideea obţinerii soluţiei optime);
- simulare (inclusiv jocuri);
- interfaţă cu utilizatorul (de exemplu Windows).

1.2 Componentele unui sistem grafic

Un sistem grafic tipic conţine următoarele:

(a) procesor,
(b) memorie,
(c) dispozitiv de intrare,
(d) dispozitiv de ieşire.

Acest model include staţii de lucru, calculatoare personale, terminale ataşate la un calculator central cu timpi distribuiţi.

În sistemele grafice se întâlnesc două tipuri distincte de memorie. Pentru procesarea programului utilizatorului se foloseşte memoria standard a calculatorului. Imaginea este formată prin procesare aritmetică standard. Procesarea imaginii pe terminal, necesită o memorie care poate fi accesată extrem de rapid. Astfel, memoria video, numită şi memorie ecran sau fişier ecran, este, uzual, diferită faţă de memoria utilizată de procesorul ce formează imaginea, atât în caracteristicile fizice cât şi în organizare.

1.3 Procesorul de terminal

Crearea unei imagini pe dispozitivul de ieşire a unui sistem grafic presupune două procese:

1. Primul este procesul de formare al imaginii. În această etapă, sunt procesate comenzile utilizatorului. Imaginea este formată din elemente (puncte, linii, texte) care sunt disponibile în sistem cu anumite atribute (culoare, font). Interfaţa cu utilizatorul este o parte a acestui proces. Imaginea poate fi specificată printr-un număr variat de căi, de exemplu, printr-un program de desenare controlat interactiv prin meniu sau printr-un program C ce utilizează biblioteca grafică. Procesorul fizic utilizat în această etapă este de obicei procesorul calculatorului gazdă.
2. Al doilea proces este afişarea imaginii. Procesorul indicat pentru executarea acestei operaţii nu este de tipul standard al procesoarelor întâlnite în mod obişnuit în majoritatea calculatoarelor. Se utilizează procesorul de terminal, hardware specializat pentru a asista convertirea primitivelor grafice într-o "hartă" de biţi şi pentru a executa operaţii de mutare, copiere şi modificare a biţilor.

Un procesor de terminal (controlor grafic, coprocesor de display, display processor, procesorul de imagine rastru, RIP) are un set limitat de instrucţiuni. Principalul obiectiv al acestuia este reîmprospătarea ecranului cu o anumită frecvenţă. Entităţile grafice sunt plasate în memoria video care este accesată de unitatea de procesare a ecranului. Computerul gazdă defineşte primitivele grafice o singură dată. Odată ce aceste primitive sunt trimise la procesorul de terminal, gazda este liberă pentru a executa alte sarcini.

Procesoarele de terminal sunt incorporate în majoritatea sistemelor grafice performante. În sistemele grafice simple, precum calculatoarele personale, controlorul grafic este o componentă software a bibliotecii grafice.

1.4 Principiul tubului catodic

Principiul de construcţie al unui tub cu raze catodice, prescurtat CRT (majoritatea monitoarelor video intră în această categorie) constă în emiterea luminii pe o peliculă de fosfor lovită de un electron.

Un fascicul de electroni emis de un "tun de electroni" trece printr-un sistem de focalizare şi unul de deflecţie care direcţionează fasciculul spre punctul specificat de pe un ecran impregnat cu fosfor. Intensitatea iluminării este stabilită prin controlarea voltajului unei grile ce precede sistemul de focalizare. Elementul fosforescent emite o mică cuantă de lumină în fiecare punct lovit de fasciculul de electroni. Lumina emisă dispare rapid (intensitatea descreşte exponenţial). Imaginea este necesar a fi actualizată permanent (cel puţin de 60 ori pe secundă). Dacă rata de reîmprospătare descreşte, apare fenomenul de clipire.

Sunt utilizate diferite tipuri de elemente fosforescente. În afara culorii, o diferenţă majoră între acestea este persistenţa, adică cât timp elementul continuă să emită lumină după ce fasciculul de electroni a fost mutat.

Persistenţa este definită ca timpul necesar luminii pentru descreşterea cu 10% din intensitatea iniţială.

Terminalul cu memorarea imaginii pe ecran (DVST) sau tuburile catodice cu

memorie permit reţinerea imaginii pe terminal un timp de nivelul orelor. Din păcate un asemenea dispozitiv nu poate fi utilizat în aplicaţiile în timp real, deoarece părţi ale imaginii anterioare rămân până când întreaga imagine este ştearsă.

1.5 Grafica rastru. Zona tampon cadru

Există două moduri principale de memorare a imaginii:

1. punct cu punct — în grafica rastru (raster image display);
2. ca o mulţime de segmente de dreaptă pentru care se memorează coordonatele capetelor segmentelor în sistemul propriu al ecranului (vector, calligraphic sau stroke-writing display). Această variantă este compatibilă şi cu periferice de tip plotter, dar are anumite dezavantaje: este dificil de adaptat pentru sintetizarea de imagini realiste (algoritmii corespunzători fiind greoi), oferă posibilităţi reduse de utilizare a culorilor şi a nuanţelor, iar când se depăşeşte un număr critic de segmente, apare fenomenul nedorit de pâlpâire (flickering) a imaginii.

Display-urile tip rastru se caracterizează printr-un spaţiu de afişare alcătuit dintr-o matrice rectangulară de elemente grafice indivizibile. Pentru reprezentarea unei imagini pe ecran, calculatorul sau procesorul dispozitivului de afişare trebuie să genereze o aproximare discretă cât mai fidelă pentru imaginea ideală şi să selecteze în rastru punctele corespunzătoare imaginii discretizate. Definirea unei imagini se realizează printr-un set de valori de intensitate pentru fiecare punct-ecran şi aceste valori sunt vizualizate pe ecran câte o linie la un moment dat.

În grafica rastru, imaginea este stocată ca o matrice de elemente numite pixeli. Astfel, o imagine digitizată este o matrice în care fiecare element este o colecţie de numere ce descriu atributele unui pixel al imaginii (sau o funcţie de variabilă discretă: unei perechi de numere naturale i se asociază o valoare). Adesea aceste numere sunt reprezentări discrete ale unui interval de numere reale. De exemplu, întregii de la 0 la 255 pot fi utilizaţi pentru a reprezenta o diviziune echidistantă a intervalului [0,1] (numerele reprezentând intensităţi ale punctelor imaginii în scara de gri sau intensităţi ale unor componente ale culorilor respectivelor puncte).

Pentru o imagine colorată, valoarea asociată unui pixel este un ansamblu de trei numere reprezentând intensităţile componentelor de roşu, verde şi albastru ale culorii pixelului sau trei numere reprezentând indexi în tabele de intensităţi de roşu, verde şi albastru sau un singur număr care este un index într-o tabelă de triplete de culoare.

Dimensiunile matricii sunt numite lăţime, respectiv înălţime a imaginii, iar numărul de biţi asociaţi cu fiecare pixel al matricei este adâncimea imaginii. Rastrul este matricea de pixeli ce reprezintă întreaga arie a ecranului. Fiecare linie de pixeli este referită ca linie de baleiere sau scanare.

Pixelii sunt stocaţi într-o zonă de memorie specială, numită zona tampon cadru. Motivul introducerii acestei memorii speciale este următorul: dacă me-

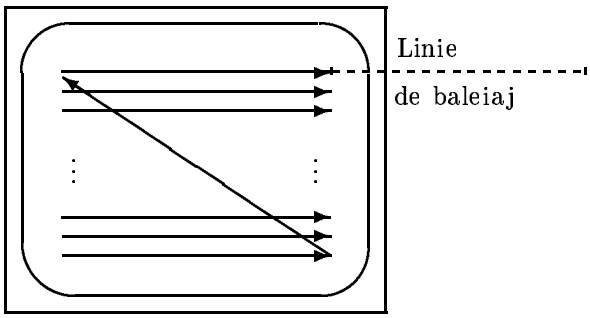


Figura 1.1: Baleierea ecranului de tip rastru

moria video poate fi citită din program, atunci imaginea poate fi construită chiar în această memorie; din păcate, nu toate terminalele oferă posibilitatea de citire prin program a stării bitului asociat cu un punct de pe ecran. Astfel, pentru a oferi posibilitatea prelucrării ulterioare a reprezentărilor simple ale imaginii, se poate crea un "ecran virtual", o zonă de memorie care este identică ca mărime cu memoria video, în care se construieşte imaginea. După definitivare, imaginea se transpune de aici în memoria video, deci pe ecran.

Zona tampon cadru se întâlneşte şi sub denumirile de zonă (buffer) de împrospătare, frame buffer, refresh buffer sau bitmap/pixmap (bitmap pentru sisteme cu 1 bit pentru reprezentarea unui pixel, pixmap sau pixel-map pentru sisteme cu mai mulţi biţi pentru un pixel), fiind folosită şi din considerente de creştere a vitezei atunci când există mai multe plane de memorie video.

Zona tampon este scanată (baleiată) secvenţial de adaptorul grafic (o linie de rastru la un moment dat) cu o rată de 50 la 70 de ori pe secundă şi imaginea este împrospătată linie cu linie, în modul în care se produc imaginile de televiziune (figura 1.1). Primitivele grafice, precum segmentele de linii sau textele, sunt afişate prin excluderea sau includerea unor pixeli în zona tampon cadru. Acest proces este numit conversie de scanare, conversie de baleiere sau rasterizare.

Rata de scanare este numărul de linii scanate pe secundă. Este aproximativ egală cu produsul dintre rata de împrospătare a imaginii şi numărul liniilor ecranului.

Adesea ciclul de împrospătare a imaginii-ecran este segmentat: la o parcurgere a ecranului se vizualizează anumite linii, la următoarea trecere, liniile rămase (interlacing). Această tehnică este utilizată în special la terminalele rastru cu rată de împrospătare redusă.

O cale simplă pentru a construi icoane grafice şi a le face să apară şi să dispară repede este aceea de a le creea o singură dată în memorie şi de a le copia în frame-buffer când sunt necesare. În pachetele de grafică rastru se realizează această tehnică prin generarea primitivelor în zone de memorie, numite tablouri (canvas) şi copierea acestora în frame-buffer. Un asemenea tablou este o structură de date care înmagazinează o imagine, precum şi informaţii referitoare

la dimensiunea şi atributele imaginii.

În Figura 1.2 sunt prezentate cele două sisteme grafice clasice. Pachetele grafice au sarcini mai puţine dacă există în sistem un procesor de terminal care tratează primitivele grafice direct (în acest caz pachetul grafic converteşte numai reprezentările interne ale primitivelor în formatele acceptate de perifericul de display).

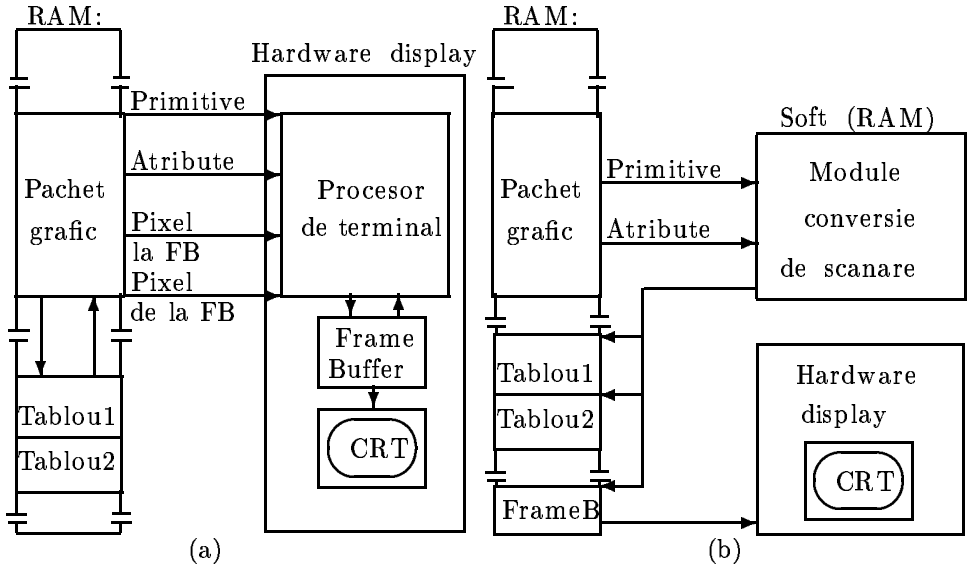


Figura 1.2: Funcţionarea unui sistem grafic (a) cu procesor de terminal şi frame-buffer (b) fără procesor de terminal şi frame-buffer incorporat în memorie.

1.6 Pixel. Rezoluţie. Raport aspectual. Spaţiu adresabil

Calitatea imaginii afişate pe un ecran depinde de mărimea unui punct al ecranului şi de numărul de puncte care pot fi create într-o unitate de lungime (număr ce desemnează adresabilitatea şi care diferă pe direcţiile orizontală şi verticală).

Un parametru foarte important al unui display este rezoluţia. Ea se defineşte prin dimensiuni relative la cel mai mic detaliu ce poate fi distins pe ecran.

Elementul cel mai fin vizibil pe ecran este spotul (punct, pixel, picture--element). Dimensiunea spotului este diametrul punctului luminos pe care îl creează pe ecran fascicolul focalizat al tunului tubului de electroni. Ca ordin

de mărime, spotul are în jur de 0.02 inch = 0.508 mm (1 inch = 2.54 cm). Distanţa dintre două puncte adiacente trebuie să fie mai mică decât diametrul unui punct pentru a asigura trasarea unei imagini continue.

Rezoluţia unui ecran este indicată prin produsul dintre numărul de pixeli pe orizontală şi pe verticală. Rezoluţia poate fi definită şi prin numărul maxim de puncte care pot fi afişate fără suprapunere sau numărul de linii distincte care pot fi create pe ecran într-un inch (dacă 40 de linii negre intercalate cu 40 de linii albe pot fi distinse într-un inch, rezoluţia este de 80 linii/inch). Rezoluţia depinde de tipul de elemente fosforescente şi de sistemele de deflecţie şi focalizare.

O proprietate importantă a monitoarelor video este raportul aspectual (aspect ratio). Acest număr reprezintă raportul dintre numărul punctele pe verticală, respectiv pe orizontală necesare pentru a produce linii de lungime egală în ambele direcţii ale ecranului. Un raport de 3/4 indică faptul că o linie verticală compusă din trei puncte-ecran are aceeaşi lungime ca o linie orizontală compusă din patru puncte ecran.

Spaţiul vizibil este delimitat de rezoluţia disponibilă. Dacă rezoluţia este 256×176, sunt vizibile toate punctele pentru care abscisa $x \in [0, 255]$, iar ordonata $y \in [0, 175]$. Spaţiul adresabil este domeniul pentru care operaţia de a pune un punct la coordonatele (x, y) nu conduce la o eroare. Spaţiul adresabil include spaţiul vizibil. Există display-uri pentru care spaţiul adresabil este limitat numai de posibilităţile de reprezentare a numerelor în sistemul folosit (sisteme cu spaţiu adresabil infinit). Atunci când spaţiul adresabil este limitat şi diferă de spaţiul vizibil, lucrurile se petrec ca şi când nu am putea vedea la un moment dat decât o parte din foaia pe care desenăm.

1.7 Cuplor grafic

În calculatoarele personale, frame-buffer-ul face parte din memoria calculatorului, iar memoria video este conţinută în spaţiul propriu zis de adresare al procesorului standard. Imaginea este formată în memoria video direct, fără a mai fi nevoie de o transmisie, obţinându-se astfel o mare viteză de execuţie. Scrierea directă în memoria ecran este numai o fază a procesorului. Pentru afişarea propriu-zisă această memorie video este citită secvenţial de un bloc logic, independent de procesor, care realizează semnalele pentru monitor, conform standardului acestuia. Această funcţie este realizată de adaptorul grafic, numit şi cuplor grafic sau cuplor video (figura 1.3).

Primele cuploare grafice existente pe calculatoarele personale au fost MDA (Monochrome Display Adapter) şi CGA (Color Graphics Adapter). Astăzi, cele mai cunoscute cuploare grafice ce echipează calculatoarele din familia IBM-PC-AT sunt EGA (Enhanced Graphics Adapter – 1984) şi VGA (Video Graphics Array – 1987).

Performanţele unui cuplor grafic sunt măsurate prin viteza de execuţie, rezoluţia (dimensiunea în pixeli, puncte) şi numărul de culori simultane. Standardele pentru cuploare grafice sunt următoarele:

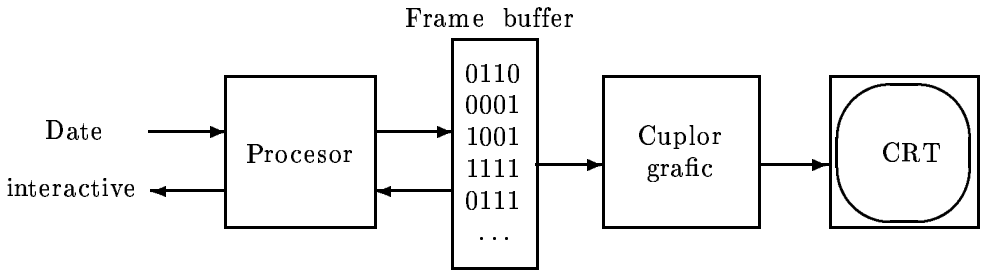


Figura 1.3: Sarcina cuplorului grafic

HGC(Hercules Graphics Card), rezoluţie standard 720 × 348, 2 culori;
CGA(Color Graphics Adapter), rezoluţie standard 640 × 200, 16 culori;
EGA(Enhanced Graphics Adapter), rezoluţie standard 640 × 350, 16-64 culori;
VGA(Video Graphics Array), rezoluţie standard 640 × 480, 16 culori;
SVGA(Super Virtual Graphics Adapter), rezoluţie standard 640 × 480, 16-256 culori.

Printre noile realizări tehnologice se remarcă AGA (Advanced Graphics Adapter) cu o rezoluţie de 1024 × 768. Cea mai bună rezoluţie realizată este de 4096 × 4096 pixeli.

1.8 Clasificarea terminalelor grafice

Terminalele (monitoarele, ecranele, display-urile) grafice pot fi clasificate astfel:

(i) după modul de memorare a imaginii:
   1. ecrane cu memorarea imaginii pe ecran (DVST – Direct View Storage Tube);
   2. ecrane cu memorarea imaginii într-o anumită zonă a RAM-ului (sisteme video-RAM cu baleiaj – Raster Scan Display);

(ii) după modul cum este obţinută imaginea pe ecranul tubului video:
   1. ecrane rastru;
   2. ecrane caligrafice (display-uri vectoriale);

(iii) după cum este întreţinută imaginea pe ecran:
   1. ecrane cu reîmprospătare periodică a imaginii (refresh display);
   2. ecrane care memorează imaginea (storage display);

(iv) după numărul de culori pe care este capabil să le afişeze:
   1. ecrane alb-negru;
   2. ecrane color.

Există mai multe metode de parcurgere (baleiere, scanare) a imaginii de către spot. Cele mai utilizate sunt:

- parcurgerea aleatoare folosită la display-urile caligrafice;
- parcurgerea rastru cu două variante:
  1. parcurgerea neîntreţesută, când tot ecranul este parcurs într-un singur cadru,
  2. parcurgerea întreţesută, când ecranul este parcurs în două semicadre: în primul, liniile impare, iar în cel de-al doilea, liniile pare.

Alte tipuri de terminale sunt cele care folosesc diode emitente de lumină (LED) sau cristale lichide (LCD), ecranele cu plasmă (Plasma Panel, matrice de bulbi de neon ce crează o imagine ce nu necesită împrospătare), şi ecranele cu laser ce produc imagini tridimensionale pe un suport bidimensional sau într-o incintă specială (display-uri holografice).

1.9 Echipamente grafice interactive

Echipamentele logice de intrare într-un sistem grafic se pot clasifica astfel:

1. tastatura: pentru introducerea unui şir de caractere;
2. locatorul: identifică o poziţie şi/sau o orientare (joystick, trackball, mouse, digitizoare: tableta sonică, cu stilet sau cursor, tableta sensibilă la atingere, creionul luminos);
3. selectorul (întrerupător): alege o variantă din mai multe alternative posibile (taste funcţionale, butoane mouse);
4. valuatorul: permite introducerea unei singure valori în spaţiul numerelor reale (potenţiometrii rotativi montaţi în mod curent în grupuri);

Tastatura produce un cod digital corespunzător secvenţei de taste apăsate de către utilizator. Aceste secvenţe sunt interpretate în mod uzual ca şi coduri de caractere. Pe de altă parte, grupuri de taste pot fi interpretate ca intrări grafice. De exemplu, tastele "săgeţi" pot fi utilizate pentru deplasarea cursorului pe ecran.

Joystick-ul este o pârghie verticală (stick) montată pe o bază, ce permite mutarea cursorului în orice direcţie. Constă din doi potenţiometri ale căror ieşiri corespund poziţiei unghiulare a tijei. Majoritatea selectează poziţia pe ecran prin actuala mişcare a pârghiei sau prin apăsare.

Operaţii similare se pot obţine prin utilizarea unui trackball — acesta realizează aceleaşi funcţii ca şi joystick-ul, dar diferă constructiv, fiind prevăzut cu o bilă ce poate fi rotită, rotaţia fiind codificată de 4 traductori optici şi transferată unităţii de control care converteşte codurile în deplasări ale cursorului pe axele de coordonate planare. În acest caz, mişcarea cursorului este obţinută prin mutarea unei sfere în loc de pârghie.

Joystick-ul şi trackball-ul au fost extinse la dispozitive ce permit mişcarea în a treia dimensiune. Mai performant, spaceball-ul, o sferă rigidă pe care utilizatorul o poate muta în orice poziţie, indică o direcţie sau o translaţie 3D.

Spre deosebire de joystick şi trackball, mouse-ul poate fi utilizat în mutări relative ale cursorului pe ecran. Mouse-ul mecanic are două role care preiau deplasările pe cele două direcţii, x şi y, iar mouse-ul optic se bazează pe

întreruperea reflexiei unei suprafeţe cu linii reflectorizante şi nereflectorizante. Butoanele sunt utilizate pentru semnalarea unor operaţii speciale. Mouse-ul vede ecranul fizic al calculatorului ca o matrice de puncte ce reprezintă ecranul virtual şi utilizează perechi de puncte virtuale pentru localizarea oricărui punct sau obiect de pe ecran. Există o corespondenţă de 1:1 între punctele din ecranul virtual şi pixelii din ecranul fizic. Mouse-ul există sub formă de:

1. cursor grafic, care este o matrice de pixeli (16 × 16) ce se deplasează peste imaginile de pe ecran;
2. cursor text software, constituind un atribut de text, cursor ce poate fi deplasat de la un caracter la altul;
3. cursor text hardware, care poate fi obţinut prin modificarea formei cursorului implicit dat de adaptorul grafic respectiv.

La un moment dat poate fi activ doar un singur cursor. Driver-ul de mouse (programul ce însoţeşte dispozitivul) realizează funcţii privind selectarea tipului de cursor dorit, modificarea atributelor cursorului, salvarea şi restaurarea unor porţiuni de pe ecran, controlează mişcările cursorului exprimate prin mărime, direcţie şi durată.

Digitizoarele sunt dispozitive de introducere în calculator a coordonatelor numerice ale unor puncte sau curbe de pe un desen deja realizat, coordonate raportate la un sistem rectangular de axe ataşat suprafeţei de lucru. Această acţiune are loc în scopul reproducerii desenului (în urma unor modificări, completări sau alte prelucrări) pe un ploter sau o imprimantă matriceală. Se activează un cursor manual sau un stilet la poziţiile corespunzătoare de pe o suprafaţă plană. Performanţele acestor dispozitive sunt date de numărul de funcţii predefinite oferite de către driver precum şi de rezoluţie. Cele mai performante digitizoare au un număr de 100-150 de funcţii predefinite, iar rezoluţia până la 100 de puncte distincte de digitizare per centimetru. Digitizoarele tridimensionale permit înregistrarea poziţiilor în spaţiu. Un digitizor se compune din suprafaţa de lucru, cursor şi electronica de comandă şi de interfaţare cu calculatorul.

O tabletă grafică sensibilă la atingere (touch panel) este un digitizor de dimensiuni mari care poate fi folosit pentru selecţie, o placă transparentă care poate acoperi ecranului unui CRT şi care permite selectarea poziţiilor indicate prin atingerea cu un deget. Metoda de determinare a poziţiei este optică, electrică sau acustică.

Un creion optic (light pen) permite selectarea unei poziţii pe ecran prin detectarea luminii primite de la punctele ecranului CRT. Coordonatele introduse printr-un asemenea dispozitiv sunt utilizate în aplicaţii grafice pentru selectarea sau poziţionarea obiectelor pe ecran.

Scanarea (baleierea) unei imaginii deja trasate este o soluţie (pentru construirea pe ecran şi memorarea unei imagini) mult mai rapidă decât utilizarea digitizoarelor, dar apare problema vectorizării, adică a procesului de extragere a primitivelor grafice din imaginea discretă rezultată. Scanner-ele performante actuale sunt cele foto, cu tambur şi rază laser, şi cele cu cristale lichide.

1.10 Echipamente grafice pasive

Echipamentele de trasare dintr-un sistem grafic pot fi de două tipuri: imprimante sau plotere.

Imprimantele (printer-ele) afişează informaţia prin metoda impactului sau nonimpact. Imprimantele cu impact realizează imprimarea unui caracter deja format. Asemenea caractere sunt montate, de exemplu, pe lanţuri. Imprimantele nonimpact utilizează adesea metoda matricei de puncte. Acestea oferă posibilităţi deosebite pentru grafică.

Rezoluţia grafică a imprimantelor este măsurată în dpi (densitate puncte per inch), 180 dpi fiind o rezoluţie bună pentru aceste dispozitive. Există mai multe tipuri de astfel de dispozitive cu performanţe din ce în ce mai mari, ajungându-se la imprimante cu laser cu rezoluţii între 300 dpi şi 1800 dpi (acestea putând fi comparate cu cele mai moderne echipamente de tipărit).

Imprimantele cu ace conţin capete de scriere cu 7-24 pini, fiecare putând fi acţionat independent. Imprimantele color pot conţine un singur cap de scriere cu bandă multicoloră sau mai multe capete de scriere, fiecare cu bandă de culoare distinctă.

Imprimantele cu jet de cerneală construiesc imagini prin puncte. Fiecare punct este rezultatul expedierii unei minuscule picături de cerneală sub acţiunea unor câmpuri electrostatice.

Imprimantele termice transferă pigmenţi de pe hârtia colorată pe hârtia albă specială.

Imprimantele laser baleiază o rază laser de-a lungul unui tambur rotativ încărcat pozitiv cu selenium. Ariile atinse de raza laser îşi pierd încărcarea. Toner-ul încărcat negativ este adăugat la aria tamburului şi este transferat pe hârtie. Pentru imprimare color procesul este repetat de trei ori, o dată pentru fiecare culoare primară. Un microprocesor realizează conversia de scanare, astfel încât un mare număr de imprimante laser acceptă documente PostScript (descriere procedurală a imaginii).

Ploterele sunt destinate trasării imaginilor realizate de aplicaţii grafice, desenele executându-se în tuş, cerneală sau pastă, pe un suport de hârtie (plot = trasare). Au o precizie mare de execuţie şi pot executa orice tip de desen. Comenzile tipice pentru un ploter includ cele care ridică sau coboară capul şi îl mută la o poziţie specifică. Depinzând de capabilităţile ploterului, pot fi alese de la 4 la 16 direcţii diferite pentru mişcare. Aceste dispozitive se clasifică astfel:

(i) funcţie de principiul folosit pentru scriere:

1. plotere cu peniţă: folosesc un dispozitiv mecanic la care peniţa are posibilitatea de mişcare pe două axe ortogonale; se utilizează una la opt peniţe selectabile prin program fiecare putând avea culori sau grosimi diferite ale scrisului.
2. fotoplotere: mediul de desen este un film fotografic, peniţa fiind înlocuită cu un cap optic ce impresionează pelicula;
3. plotere matriceale: descompun desenul în puncte elementare situate pe o grilă pătratică;

4. plotere electrostatice: folosesc acelaşi principiu de descompunere a desenului în puncte elementare, punctul fiind realizat printr-o depunere electrostatică a substanţei de contrast (toner);
5. plotere termice: impresionează, pe baza unui principiu termic, o hârtie termosenzitivă specială;
6. plotere cu jet de cerneală: ca şi la imprimantele cu jet de cerneală, fiecare punct este trasat prin expedierea unei minuscule picături de cerneală printr-un tub capilar sub acţiunea unor câmpuri electrostatice.

(ii) după suprafaţa utilă a desenului (A4, A3 etc);

(iii) după poziţia şi modul de fixare a hârtiei:
1. plotere plane (flatbed plotter): utilizează o bară care se poate muta de la o anumită latură a dispozitivului la cea opusă, în timp ce capul de imprimare se mişcă de-a lungul barei;
2. plotere cu tambur (drum plotter): bara pe care se plimbă capul de imprimare este fixă, mişcându-se hârtia înainte şi înapoi;
3. plotere cu pat-bombat (beltbed plotter): bara este fixă, imprimarea se realizează pe o suprafaţă netedă, după care hârtia este rulată.

Proprietatea geometrică de bază a ploterelor este rezoluţia. Aceasta se defineşte prin numărul de puncte care pot fi separat identificate şi adresate pe o direcţie elementară. Se exprimă fie ca număr de puncte pe unitatea de lungime (ex: 100 pct/mm), fie ca mărime a pasului elementar (ex: 0,01mm).

1.11 Clasificarea aplicaţiilor grafice

O aplicaţie grafică poate fi inclusă într-una din următoarele categorii:
1. editor grafic,
2. bibliotecă grafică,
3. program grafic specializat,
4. produs de birotică (desktop publishing, worksheet graphics).

Editoarele grafice (software bazat pe selecţie, turnkey software) sunt destinate prelucrării grafice în domeniul proiectării asistate de calculator şi au următoarele funcţii:
— salvarea imaginilor în fişiere,
— introducerea de noi elemente grafice,
— trasări de curbe, suprafeţe şi linii poligonale,
— selectarea culorilor şi a tipurilor de linie,
— scalări şi rotaţii ale obiectelor selectate,
— haşurări diverse ale figurilor convexe,
— facilităţi de trasare cu realizarea impresiei obiectelor tridimensionale,
— mărirea şi micşorarea desenelor,
— utilizarea textelor cu diverse tipuri de caractere, cel puţin ale alfabetului latin şi alfabetului grecesc,
— editarea de simboluri, utilizarea de simboluri matematice, astronomice, muzicale, etc.,

— trasarea în diverse sisteme de coordonate şi tipuri de proiecţii,
— selectarea sistemului de măsură şi a dimensiunii paginii de desen,
— localizarea obiectelor prin referinţă la alte obiecte,
— atribute de vizibilitate, prioritate, culoare pentru obiecte şi posibilitatea editării folosind aceste atribute.

Exemple de produse care înglobează editoare grafice sunt: AutoCAD, Freelance 2 Plus, Windows Paint, PC Paintbrush, Deluxe Paint, Mini CAD 2, Generic CAD.

În sistemele CAD, anumite părţi ale unui desen pot fi construite în mod interactiv. După specificarea dimensiunilor obiectului, proiectantul are posibilitatea de a vizualiza orice faţă a obiectului în forma finală. Arhitecţii pot porni de la planurile de desen standard. Se pot face schimbări experimentale. Tehnici similare sunt utilizate în proiectarea reţelelor de comunicaţii. Pentru proiectanţii de automobile, avioane, aeronave sau vapoare o facilitate importantă este aceea că suprafeţele ce constituie anumite secţiuni sau componente ale vehiculului pot fi proiectate separat şi potrivite apoi pentru a construi obiectul final. Simulatoarele permit testarea performanţelor vehicolelor astfel construite.

Bibliotecile grafice (terminal-based software) sunt destinate prelucrărilor grafice prin intermediul limbajelor de programare de nivel înalt (exemple: mediile Borland). Aceste biblioteci conţin rutine (primitive) grafice care realizează funcţiile grafice necesare aplicaţiilor grafice. O bibliotecă grafică trebuie să conţină:

— rutine de iniţializare a sesiunii de grafică: acestea selectează modul grafic şi stabilesc zonele de memorie pentru scrierea şi afişarea imaginilor;
— rutine pentru stabilirea zonei coordonatelor vizibile ale desenului (spaţiul utilizator) şi zona activă pe ecran (spaţiul disponibil);
— rutine pentru selectarea culorilor, rutine pentru stabilirea atributelor de culoare de trasare, stil de linie, grosime linie etc;
— rutine pentru trasarea liniilor, arcelor, elipselor, poliliniilor, trasarea haşurilor, afişarea textului;
— rutine pentru copierea imaginii grafice la imprimantă;
— rutine pentru gestiunea memoriei ecran.

Programele grafice specializate sunt destinate rezolvării problemelor din anumite domenii, oferind utilizatorului posibilitatea să enunţe probleme cât mai simplu şi în concordanţă cu limbajul utilizat în domeniul său. În general, aceste programe cuprind două părţi:

1. nucleul, ce efectuează calculele sau operaţiile corespunzătoare domeniului respectiv (exemple: calcule matematice, vezi Mathematica şi Maple, calcule tehnico-ştiinţifice, vezi Eureka şi MathCAD),
2. interfaţa cu utilizatorul, ce transferă informaţia de la utilizator la calculator şi asigură prezentarea şi procesarea grafică a rezultatelor.

Mathematica este un produs program care realizează calcule matematice în mod simbolic, adică operează în mod simbolic cu ecuaţii şi formule, face reduceri şi descompuneri în factori, derivează şi integrează funcţii (în cazul în care pentru integrală există soluţia analitică; dacă integrarea simbolică nu

este posibilă, atunci programul prezintă o soluţie numerică). Reprezentarea rezultatelor diverselor probleme se poate realiza în două sau trei dimensiuni în coordonate rectangulare sau sferice. Funcţiile grafice ale programului oferă posibilităţi pentru procesarea textelor matematice, fiind un foarte bun editor de texte matematice. În MathCAD şi Eureka, introducerea datelor se face folosind limbajul matematic uzual. Problema ce trebuie rezolvată se prezintă prin ecuaţii, sisteme, relaţii etc, care se declară prin notaţii matematice obişnuite. Calculele sunt realizate prin metode numerice.

Produsele de birotică de tipul desktop publishing sunt destinate realizării de publicaţii (ziare, reviste, reclame, etc), parcurgând toate etapele dintr-o tipografie clasică. Principalele funcţii ale acestor produse sunt:

— redactarea documentului cu ajutorul unui procesor de texte,
— editarea şi corectarea documentului,
— includerea textului în pagină,
— inserarea în text a desenelor realizate automat sau preluate prin intermediul unui scanner,
— realizarea design-ului paginii,
— machetarea paginilor şi tipărirea conform machetei.

Cele mai des utilizate produse de acest tip sunt: Xerox Ventura Publisher, WordStar, WordPerfect, MultiMate, TeX, LaTeX, PageMaker, XPress, Super Paint, Publish IT, SciWord, multe dintre acestea fiind compatibile cu editoare grafice precum AutoCAD, PaintBrush, Windows Paint.

Produsele de birotică de tipul worksheet graphics sunt destinate aplicaţiilor de evidenţă din domeniul financiar-contabil şi pot realiza: analiza vânzărilor, balanţe comerciale, salarii, date personale ale salariaţilor etc. Datele sunt introduse în celule care se află în cadrul unor tabele. Grafica este folosită la prezentarea ieşirilor sub formă de diagrame, grafice, tabele, etc. Cele mai cunoscute produse cu aceste funcţii sunt Lotus şi QuattroPro.

Un interes considerabil s-a acordat elaborării unui standard al aplicaţiilor grafice. În 1985, Organizaţia Internaţională a Standardelor (ISO) a aprobat în acest sens Graphical Kernel System (GKS). Standardul GKS defineşte un set de funcţii ce realizează sarcini grafice într-un mod independent de limbaj. Prima versiune a fost elaborată pentru grafica bidimensională. În 1988, este aprobată o extensie 3D a GKS-ului, dar sistemul grafic PHIGS (Programmer's Hierarchical Interactive Graphics System) mai sofisticat câştigă teren datorită gradului mai înalt de complexitate. GKS suportă gruparea primitivelor relaţionate logic — precum liniile, poligoanele sau şirurile de caractere — şi atributelor lor în colecţii numite segmente; aceste segmente nu pot fi însă compuse. PHIGS suportă gruparea primitivelor tridimensionale pe bază de compunere în ierarhii, numite structuri.; toate primitivele, precum şi invocarea substructurilor, sunt subiectul transformărilor geometrice (scalare, rotaţie, translaţie).

C a p i t o l u l 2

T r a n s f o r m ă r i g e o m e t r i c e

2.1 Transformări bidimensionale

Se consideră un reper cartezian, un obiect bidimensional descris în respectivul sistem şi $P(x, y)$ un punct oarecare al obiectului.

2.1.1 Translaţie

Prin translaţie se deplasează toate punctele unui obiect cu un anumit deplasament. Operaţia este definită relativ la cele două axe de coordonate carteziene prin deplasamentele specifice pe fiecare axă. Punctul $P(x, y)$ este translatat în punctul $P(x', y')$ dacă

$$\begin{cases} x' = x + \Delta x, \\ y' = y + \Delta y. \end{cases}$$

unde Δx, Δy sunt deplasamentele specifice. Vectorial, operaţia poate fi scrisă

$$P' = P + T$$

unde

$$P = \begin{pmatrix} x \\ y \end{pmatrix}, \qquad P' = \begin{pmatrix} x' \\ y' \end{pmatrix}, \qquad T = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}.$$

2.1.2 Scalare

Scalarea poate fi descrisă prin ecuaţiile

$$\begin{cases} x' = s_x x, \\ y' = s_y y, \end{cases}$$

unde s_x şi s_y sunt constantele de scalare, $P(x, y)$, punctul iniţial, iar $P'(x', y')$ punctul transformat. Un factor de scalare negativ produce ca efect secundar

reflecţia faţă de axa corespunzătoare. Ecuaţia vectorială este

$$P' = SP$$

unde

$$S = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}.$$

2.1.3 Rotaţie

Prin rotaţia cu un unghi θ faţă de origine, punctele unui obiect rămân la aceeaşi distanţă faţă de originea sistemului. Un unghi pozitiv respectă sensul de rotaţie trigonometric (contrar fusului orar). Utilizând coordonatele polare şi considerând că

$$\begin{cases} x = r\cos\phi, \\ y = r\sin\phi, \end{cases}$$

după rotaţie

$$\begin{cases} x' = r\cos(\phi + \theta), \\ y' = r\sin(\phi + \theta), \end{cases}$$

Prin dezvoltarea funcţiilor cos şi sin din expresia anterioară se obţin ecuaţiile transformării în coordonate carteziene:

$$\begin{cases} x' = x\cos\theta - y\sin\theta, \\ y' = x\sin\theta + y\cos\theta. \end{cases}$$

Vectorial, operaţia poate fi exprimată prin

$$P' = RP,$$

unde

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

2.1.4 Coordonate omogene

Pentru transformările bidimensionale amintite mai sus, ecuaţiile vectoriale sunt următoarele:

$$P' = T + P, \quad P' = SP, \quad P' = RP.$$

Se observă că translaţia este tratată în mod distinct, printr-o adunare.

Reprezentarea punctelor în coordonate omogene permite tratarea tuturor transformărilor prin multiplicări matriceale. Pe lângă cele două coordonate carteziene standard apare o a treia: un punct (x, y) în coordonate carteziene este reprezentat prin tripletul (x, y, w) în coordonate omogene. Reprezentarea unui punct nu este unică. De exemplu, (2, 4, 8) şi (1, 2, 4) reprezintă acelaşi punct din plan. Dacă $w \neq 0$, atunci (x, y, w) reprezintă acelaşi punct ca şi

$(x/w, y/w, 1)$, iar coordonatele x/w, y/w sunt coordonatele carteziene asociate punctului omogen. Punctele cu $w = 0$ se numesc puncte la infinit.

Matricile transformărilor mai sus amintite, se rescriu în coordonate omogene astfel:

$$T(\Delta x, \Delta y) = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix}, \quad S(s_x, s_y) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

iar noile relaţii vectoriale sunt:

$$P' = T(\Delta x, \Delta y)P, \text{ sau } P' = S(s_x, s_y)P, \text{ sau } P' = R(\theta)P.$$

2.1.5 Transformări afine

Majoritatea transformărilor efectuate în lumea reală păstrează liniile drepte. Asemenea transformări includ rotaţia, reflecţia, scalarea şi translaţia şi definesc clasa de transformări numite transformări afine. Deoarece transformările afine asupra unor segmente de linii conduc tot la segmente de linii, oricare două trasformări afine sunt echivalente cu o singură transformare afină. Acest fapt ne permite construirea unor transformări complexe prin combinarea unor secvenţe de transformări simple (rotaţia în jurul originii, translaţia şi scalarea). Procesul de aplicare succesivă a mai multor transformări afine este numit concatenare.

O secvenţă de transformări afine poate fi reprezentată printr-o matrice unică. De exemplu, rotaţia în jurul unui punct (x_0, y_0) poate fi descrisă prin produsul matriceal CBA, unde

$$A = T(-x_0, -y_0), \qquad B = R(\theta), \qquad C = T(x_0, y_0).$$

O clasă importantă de operaţii o constituie transformările inverse. Dacă transformarea A produce $P' = AP$, transformarea inversă produce $P = A^{-1}P'$.

Transformările afine păstrează paralelismul dintre linii dar nu şi lungimile şi unghiurile. Un exemplu concludent este înclinarea faţă de o anumită axă. Corespunzător axei x şi unghiului de înclinare ϕ, ecuaţiile transformării sunt

$$\begin{cases} x' = x + y \operatorname{ctg} \phi, \\ y' = y, \end{cases}$$

ecuaţii care conduc la matricea de transformare

$$H_x = \begin{pmatrix} 1 & \operatorname{ctg} \phi \\ 0 & 1 \end{pmatrix}.$$

Operaţia poate fi obţinută prin concatenarea a trei transformări elementare.

2.2 Sisteme carteziene de referinţă

În aplicaţiile grafice se disting următoarele trei sisteme carteziene de referinţă:

(1) sistemul de coordonate ale lumii reale (înconjurătoare) în care observatorul este centrul acestuia (WCS – World Coordonate System);

(2) sistemul de coordonate ale unui dispozitiv (DCS – Device Coordonate System);

(3) sistemul de coordonate normalizate ale unui dispozitiv (NDCS – Normalized Device Coordonate System).

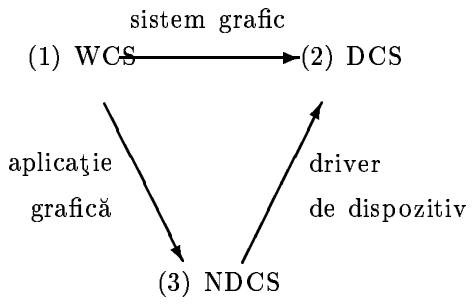


Figura 2.1: Transformarea unei imagini din lumea reală în imagine pe dispozitivul de ieşire al unui sistem grafic

Transformarea coordonatelor din sistemul (1) în sistemul (2) este sarcina sistemelor grafice (figura 2.1). În general, un sistem real are o varietate de intrări şi ieşiri, astfel încât pachetul grafic ar trebui să gestioneze mai multe sisteme (2). În majoritatea aplicaţiilor se utilizează un sistem de coordonate intermediar, desemnat prin (3). Se consideră astfel un dispozitiv virtual în termenii căruia se scrie soft-ul necesar. Dispozitivul virtual uzual este pătratul unitate din sistemul (3), care are colţul din stânga jos în originea sistemului. Valorile exprimate în unităţi de pe dispozitivul virtual sunt transformate în valori în sistemul (2) ca ultimă etapă de procesare. Această transformare este realizată de driver-urile de dispozitiv (componente software ale sistemelor grafice). Se permite astfel adăugarea unui nou dispozitiv în sistemul grafic, dacă este însoţit de un anumit driver, fără a fi necesară alterarea pachetelor grafice.

2.3 Ferestre şi zone de lucru

Anumite aplicaţii grafice permit specificarea primitivelor grafice în sistemul de coordonate ale lumii reale (WCS) utilizând unităţi de lungime. Pachetul de subrutine grafice trebuie să efectueze transformarea din coordonatele lumii reale în coordonatele ecranului (DCS).

Termenul de fereastră (window) a fost introdus spre a desemna o zonă dreptunghiulară a unui plan situat în spaţiul coordonatelor obiectului de obser-

vat, perpendiculară pe direcţia de observare, pe care se proiectează pe ecran informaţia dorită la un moment dat. Unei asemenea ferestre i se asociază o regiune dreptunghiulară în coordonatele normalizate de terminal (NDCS), numită zonă de lucru (viewport), în care imaginea din fereastră este transformată (digitizată).

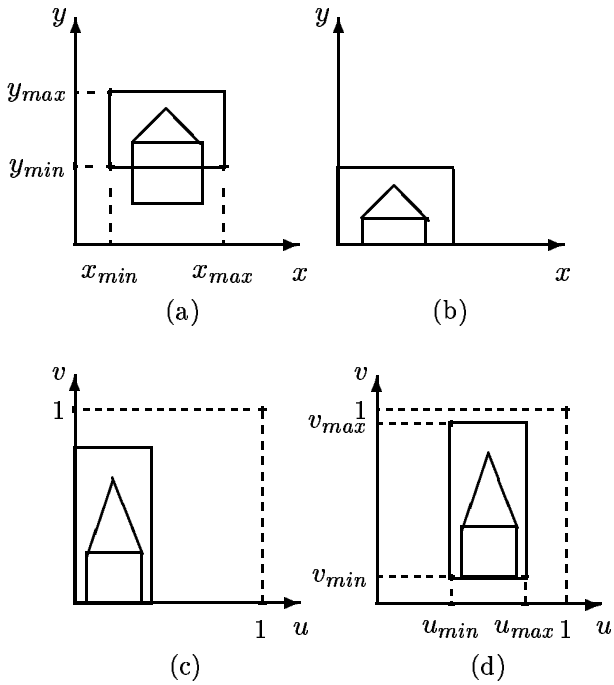


Figura 2.2: Transformarea unei ferestre într-o zonă de lucru (a) Fereastra în coordonatele lumii reale (b) Fereastra translatată în originea sistemului (c) Fereastra scalată la dimensiunea zonei de lucru (d) Translatarea imaginii în zona de lucru

Fie fereastra specificată prin (x_{min}, y_{min}), (x_{max}, y_{max}), coordonatele a două vârfuri extreme ale zonei dreptunghiulare din WCS, iar zona de lucru, prin (u_{min}, v_{min}), (u_{max}, v_{max}), unde $0 \leq u_{min} < u_{max} \leq 1$, $0 \leq v_{min} < v_{max} \leq 1$, coordonatele vârfurilor corespunzătoare ale zonei dreptunghiulare din NDCS. Matricea transformării fereastră – zonă de lucru poate fi construită prin concatenarea celor trei transformări sugerate în figura 2.2:

$$P' = MP,$$

unde

$$M = T(u_{min}, v_{min}) S \left(\frac{u_{max} - u_{min}}{x_{max} - x_{min}}, \frac{v_{max} - v_{min}}{y_{max} - y_{min}} \right) T(-x_{min}, -y_{min}).$$

Astfel, ecuaţiile de transformare a unei ferestre într-o zonă de lucru sunt următoarele:

$$\begin{cases} u = u_{min} + \dfrac{u_{max} - u_{min}}{x_{max} - x_{min}}(x - x_{min}), \\ v = v_{min} + \dfrac{v_{max} - v_{min}}{y_{max} - y_{min}}(y - y_{min}). \end{cases}$$

O pereche similară de transformări se aplică pentru trecerea de la NDCS la DCS.

Transformările efectuate asupra unui viewport se efectuează fie asupra conţinutului memoriei video (sau al refresh-buffer-ului), fie asupra coordonatelor proiecţiilor punctelor obiectului din lumea reală. În primul caz, la scalare imaginea se modifică prin înlocuirea unui pixel cu un dreptunghi având laturile (exprimate în pixeli) egale cu factorii de scară. Procedeul este utilizat pentru factori de scară întregi şi supraunitari. Rotaţia unei zone de lucru poate fi realizată ca o transformare multipas, de exemplu ca produs a trei înclinări faţă de axe:

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} 1 & -\mathrm{tg}\,(\theta/2) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin\theta & 1 \end{pmatrix} \begin{pmatrix} 1 & -\mathrm{tg}\,(\theta/2) \\ 0 & 1 \end{pmatrix}.$$

În cazul construirii unei imagini sub limitele spaţiului vizibil, pot apărea o serie de primitive care vor intersecta frontiera ferestrei (figura 2.2.a). Este necesară eliminarea din aceste primitive a porţiunilor care ies din zona observabilă (din fereastră). Această operaţie este cunoscută sub numele de clipping (decupare, retezare, tăiere).

2.4 Transformări tridimensionale

Aşa cum unei transformări bidimensionale exprimate în coordonate omogene îi corespunde o matrice 3×3, o transformare tridimensională este reprezentată printr-o matrice 4×4.

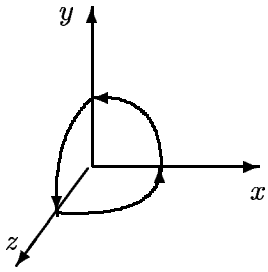
Prin transformarea de omogenizare, se asociază unui punct (x, y, z) în coordonate carteziene, un punct (x, y, z, w) în coordonate omogene, cu proprietatea că două cvadruple reprezintă acelaşi punct dacă coordonatele lor sunt proporţionale. Reprezentarea standard a unui punct omogen (x, y, z, w) cu $w \neq 0$ este $(x/w, y/w, z/w, 1)$.

Translaţia tridimensională este o simplă extensie a celei bidimensionale. Matricea transformării este

$$T(dx, dy, dz) = \begin{pmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Rotaţia oarecare a unui sistem de puncte se descompune în cel mult trei rotaţii, maxim câte una după fiecare axă a unui reper triortogonal. Pen-

tru rotaţiile în jurul axelor de coordonate s-au convenit anumite direcţii ale rotaţiilor pozitive (figura 2.3).



— pentru axa x, de la axa y înspre axa z;

— pentru axa y, de la axa z înspre axa x;

— pentru axa z, de la axa x înspre axa y.

Figura 2.3: Sensul rotaţiilor de unghi pozitiv

Astfel rotaţiile în jurul axelor sunt reprezentate prin matricele de transformare următoare:

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Rotaţia unui obiect în jurul unei drepte arbitrare, dar paralelă cu o anumită axă de coordonate presupune:

1. translatarea obiectului astfel încât axa de rotaţie să coincidă cu axa paralelă de coordonate;
2. efectuarea rotaţiei;
3. translatarea obiectului astfel încât axa de rotaţie să fie mutată în poziţia originală.

Dacă axa de rotaţie nu este paralelă cu nici una din axele de coordonate este necesară rotaţia în prealabil a acestei axe astfel încât să fie paralelă cu una din axele de coordonate. În final se aplică transformarea inversă pentru a aduce axa de rotaţie la orientarea iniţială.

Spre deosebire de rotaţie şi translaţie, înmulţirea cu factori de scală afectează distanţele dintre puncte. Cea mai simplă transformare de acest gen este asemănarea, care foloseşte un factor de scală global, s, cu care se înmulţesc coordonatele corpului. Sistemul de puncte poate fi modificat diferit după fiecare din cele trei axe ale sistemului de referinţă, caz în care se definesc factorii de scară direcţionali s_x, s_y, s_z. Astfel scalarea tridimensională este descrisă prin

matricea

$$S(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Alte cazuri particulare de scalări sunt simetriile pentru care s_x, s_y, s_y sunt -1 sau 1. Astfel, de exemplu, pentru

- simetria faţă de planul xOy: $s_x = s_y = 1$, $s_z = -1$;
- simetria faţă de axa y: $s_x = s_z = -1$, $s_y = 1$;
- simetria faţă de origine: $s_x = s_y = s_z = -1$.

Simetria faţă de un plan oarecare se poate rezolva aplicând sistemului de referinţă o translaţie şi o rotaţie astfel încât un plan de coordonate să se suprapună peste planul dat şi apoi se efectuează simetria faţă de acel plan de coordonate. După aceasta se face rototranslaţia inversă. Analog se procedează în cazul unei drepte oarecare. În cazul unui punct este suficientă o translaţie aplicată sistemului de coordonate astfel încât originea să ajungă în punctul dat, apoi se aplică relaţiile de simetrie faţă de origine; după calcularea coordonatelor punctului simetric, se aplică translaţia inversă.

Simetriile se pot exprima ca produs de simetrii faţă de planele de coordonate. Simetria faţă de o axă se obţine prin concatenarea simetriilor faţă de cele două plane a căror intersecţie este axa, iar simetria faţă de origine rezultă prin aplicarea succesivă a celor trei simetrii faţă de planele de coordonate. La rândul lor, aceste simetrii faţă de planele de coordonate se pot obţine folosind simetria faţă de un singur plan de coordonate şi rotaţiile care suprapun planele de coordonate unul peste celălalt.

Înclinarea faţă de planul xOy este descrisă prin matricea

$$H_{xy}(h_x, h_y) = \begin{pmatrix} 1 & 0 & h_x & 0 \\ 0 & 1 & h_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

2.5 Proiecţii

Complexitatea vizualizării obiectelor tridimensionale este cauzată de a treia dimensiune şi de faptul că terminalele prezintă imagini bidimensionale. Soluţia este utilizarea proiecţiilor care transformă obiectele tridimensionale în proiecţii plane bidimensionale.

2.5.1 Clasificare

Proiecţia unui obiect tridimensional este definită astfel: dreptele (razele) de proiecţie, numite şi proiectori, trec printr-un punct dat al spaţiului, numit centru de proiecţie (punct de vedere), şi prin fiecare punct al corpului, intersectând planul de proiecţie pentru a forma proiecţia.

Deoarece suprafaţa ecranului este plană, ne rezumăm la proiecţiile geometrice plane. Acestea pot fi împărţite în două clase:

1. proiecţii paralele,
2. proiecţii perspective.

Distincţia se face funcţie de distanţa dintre centrul de proiecţie şi planul de proiecţie. Dacă această distanţă este finită, proiecţia este perspectivă; dacă distanţa este infinită, proiecţia este paralelă (figura 2.4).

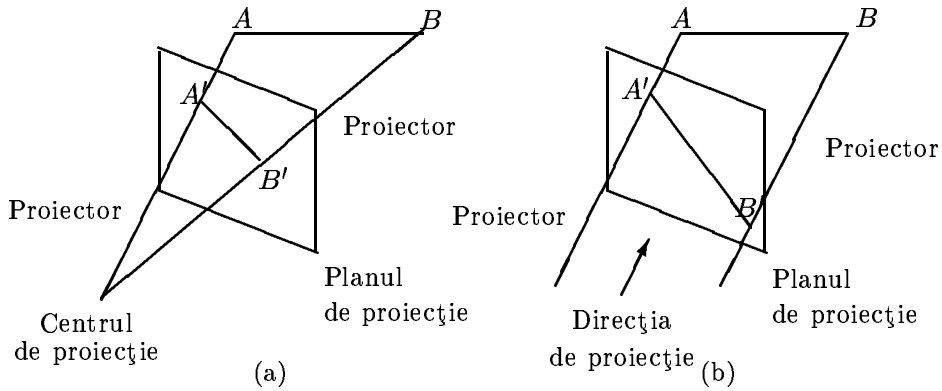


Figura 2.4: Proiecţiile unui segment de dreaptă (a) $A'B'$ este proiecţia perspectivă a lui AB (b) $A'B'$ este proiecţia paralelă a lui AB.

În cazul proiecţiei paralele, centrul de proiecţie se află la infinit, ceea ce face ca dreptele de proiecţie să fie paralele (în figura 2.4.b proiectorii AA' şi BB' sunt paraleli). Pentru a defini o proiecţie paralelă se precizează direcţia de proiecţie (un vector paralel cu proiectorii).

În proiecţia perspectivă se precizează poziţia centrului de proiecţie şi planul de proiecţie.

Există o serie de cazuri particulare atât pentru proiecţiile paralele cât şi pentru cele perspective:

- Proiecţie geometrică plană
  - perspectivă
    - cu un punct de fugă
    - cu două puncte de fugă
    - cu trei puncte de fugă
  - paralelă
    - ortografică
      - elevaţie
      - plan
      - profil
      - axonometrică
        - izometrică
        - altele
    - oblică
      - cabinet
      - cavalieră
      - altele

Observaţii:

(a) Proiecţia paralelă păstrează paralelismul liniilor, dar nu păstrează unghiurile (mai puţin cele aflate în planuri paralele cu planul de proiecţie).

(b) Efectul vizual al proiecţiei perspective este asemănător cu cel realizat de tehnica fotografică şi de sistemul vizual uman. Principala caracteristică este aceea că dimensiunea proiecţiei perspective a unui obiect variază invers proporţional cu distanţa de la obiect la centrul de proiecţie. Distanţele nu sunt cele reale, unghiurile se păstrează numai dacă aparţin unor feţe ale obiectului paralele cu planul de proiecţie, iar liniile paralele nu sunt proiectate, în general, în linii paralele. Lungimile unor segmente egale în spaţiu pot apărea în imagine diferite, depinzând de apropierea de centrul de proiecţie.

2.5.2 Proiecţia perspectivă

În proiecţia perspectivă, liniile paralele între ele şi neparalele cu planul de proiecţie converg spre un anumit punct numit punct de fugă. Dacă liniile sunt paralele cu una dintre axele principale, punctul de fugă este numit punct de fugă axial (al axei). Există cel mult trei asemenea puncte într-o imagine, câte unul pentru fiecare axă. De exemplu, dacă planul de proiecţie taie doar axa z, numai axa z are punct de fugă deoarece liniile paralele cu axele y şi x sunt paralele şi în planul de proiecţie şi nu au puncte de fugă (nu se intersectează în proiecţie).

Proiecţiile perspective sunt clasificate funcţie de numărul punctelor de fugă axiale. În figura 2.5 se prezintă trei proiecţii perspective, cu un punct de fugă, ale unui cub.

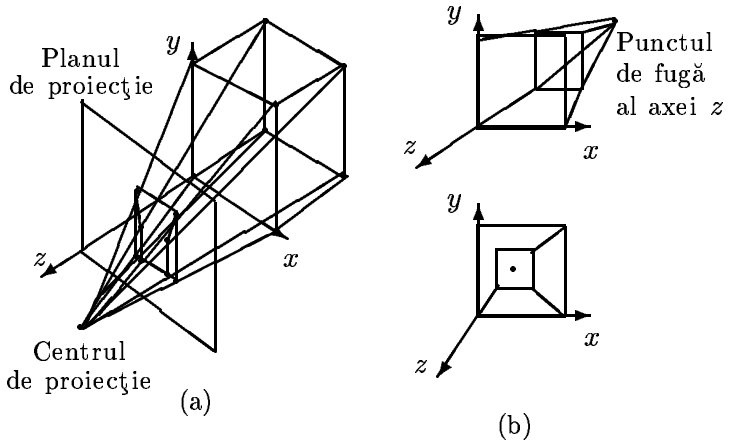


Figura 2.5: Proiecţia perspectivă a unui cub pe un plan care taie numai axa z (a) Construcţia proiecţiei (b) Două perspective cu un punct de fugă

Proiecţiile perspective cu două puncte de fugă sunt utilizate des în arhitectură şi design industrial. În figura 2.6 se prezintă modul de construcţie al

proiecţiei perspective a unui cub cu două puncte de fugă.

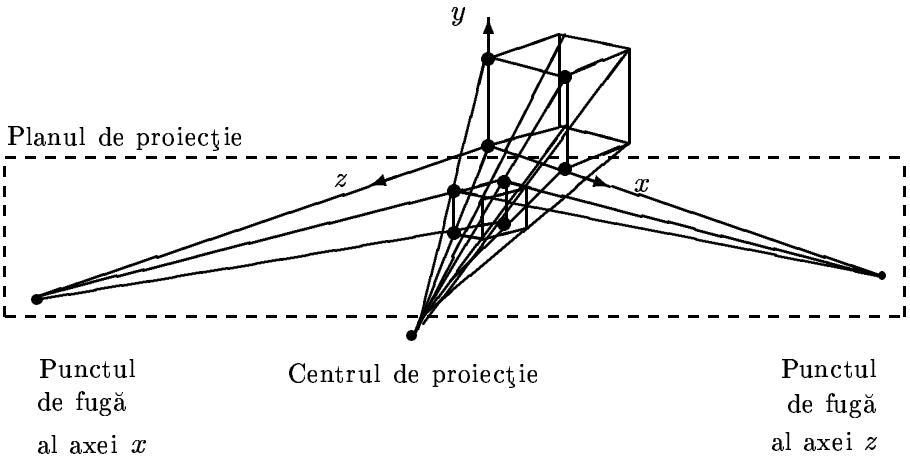


Figura 2.6: Proiecţia perspectivă a unui cub pe un plan paralel cu axa y

Funcţie de înclinarea planului de proiecţie faţă de centrul de proiecţie şi obiect se deosebesc:

1. perspectiva ascendentă (cu punctul de fugă al verticalelor situat deasupra obiectului)
2. perspectiva descendentă (cu punctul de fugă al verticalelor situat sub obiect).

Perspectiva ascendentă a unui obiect apare în situaţia în care observatorul priveşte obiectul de jos şi destul de aproape. Perspectiva descendentă a unui obiect apare în situaţia în care observatorul priveşte obiectul de sus şi de aproape.

Observaţie. În general, pentru a găsi coordonatele în proiecţia perspectivă a unui punct material se rezolvă sistemul format din ecuaţia planului de proiecţie şi ecuaţia razei vizuale. Reprezentarea perspectivei pe planul de proiecţie prin această modalitate este greoaie. De aceea este util să fie exprimate coordonatele perspective relativ la triedrul tridreptunghic de referinţă asociat planului de proiecţie şi normalei pe acest plan dusă din centrul de proiecţie. Se determină, în primul rând, transformarea tridimensională care transpune reperul cartezian ale lumii reale (în care este descris obiectul) în reperul cartezian asociat planului de proiecţie şi normalei la acesta, dusă prin centrul de proiecţie. Se aplică această transformare obiectului, apoi se proiecteză.

2.5.3 Proiecţia paralelă

Funcţie de unghiul dintre direcţia de proiecţie şi normala la planul de proiecţie, proiecţia paralelă poate fi:

1. proiecţie ortografică (ortogonală), în cazul în care direcţia de proiecţie coincide cu normala la planul de proiecţie, adică direcţia de proiecţie este perpendiculară pe planul de proiecţie;
2. proiecţie oblică, în cazul în care direcţia de proiecţie diferă de normala la planul de proiecţie.

Proiecţia ortografică

Cele mai comune proiecţii ortogonale sunt proiecţiile care utilizează, ca plane de proiecţie, anumite plane perpendiculare pe axele de coordonate. Denumirea dată în desenul tehnic unor asemenea proiecţii ale obiectelor sunt plan (vedere de sus), profil (vedere laterală) şi elevaţie (vedere frontală). În figura 2.7 se prezintă cele trei asemenea proiecţii ale unei case (originea sistemului de coordonate se află la intersecţia celor trei plane de proiecţie). Aceste proiecţii au proprietatea de a păstra distanţele şi unghiurile, astfel încât sunt des utilizate în inginerie şi construcţii. Natura tridimensională a obiectului este însă greu de înţeles, chiar dacă se studiază simultan cele trei proiecţii ale respectivului obiect.

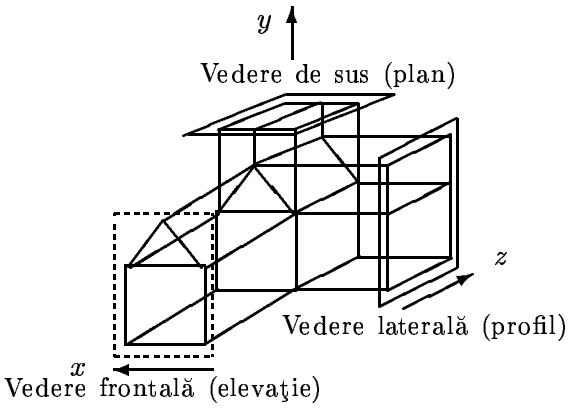


Figura 2.7: Planul, profilul şi elevaţia unei case

Fie punctul de coordonate omogene $(x, y, z, 1)$. Vederea frontală (proiecţia pe planul xOy) are matricea caracteristică

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

astfel încât ecuaţiile transformării sunt $x' = x$, $y' = y$, $z' = 0$. Pentru vederea de sus (proiecţia pe planul xOz) se efectuează rotaţia de unghi $-90°$ în jurul

axei x şi proiecţia pe planul xOy:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

astfel încât $x' = x$, $y' = z$, $z' = 0$. Pentru vederea laterală (proiecţia pe planul yOz) se efectuează rotaţia de unghi 90° în jurul axei y şi proiecţia pe planul xOy:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

astfel încât $x' = z$, $y' = y$, $z' = 0$.

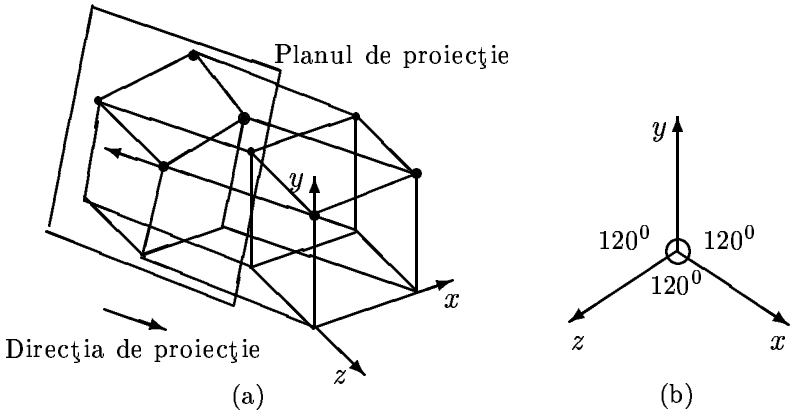


Figura 2.8: Proiecţii izometrice (a) Construirea proiecţiei unui cub în direcţia $(1, -1, -1)$ (b) Proiecţia izometrică a versorilor în direcţia $(1, 1, 1)$

Alte tipuri de proiecţii ortogonale particulare sunt proiecţiile axonometrice, pentru care planul de proiecţie nu mai este perpendicular pe nici o axă a sistemului de referinţă. Păstrează paralelismul liniilor, dar nu şi unghiurile. Distanţele se măsoară de-a lungul fiecărei axe de coordonate, în general, cu factori de scală diferiţi.

Cea mai utilizată proiecţie ortografică axonometrică este proiecţia izometrică, pentru care direcţia de proiecţie (care coincide cu normala la planul de proiecţie) face unghiuri egale cu cele trei axe ale sistemului de referinţă. În cazul acestei proiecţii cei trei factori de scară pentru măsurarea lungimilor sunt egali, iar axele principale se proiectează pe plan în trei drepte care fac unghiuri egale (figura 2.8.b). Există doar opt direcţii de proiecţie care permit proiecţii izometrice. În figura 2.8 (a) este prezentată o astfel de proiecţie a unui cub.

Proiecţia oblică

Este obţinută prin proiectarea unui obiect de-a lungul unor linii paralele care nu sunt perpendiculare pe planul de proiecţie. Proiecţia unei feţe a obiectului, paralelă cu planul de proiecţie, permite măsurarea corectă a unghiurilor şi distanţelor, celelalte proiecţii ale feţelor permiţând doar măsurarea distanţelor.

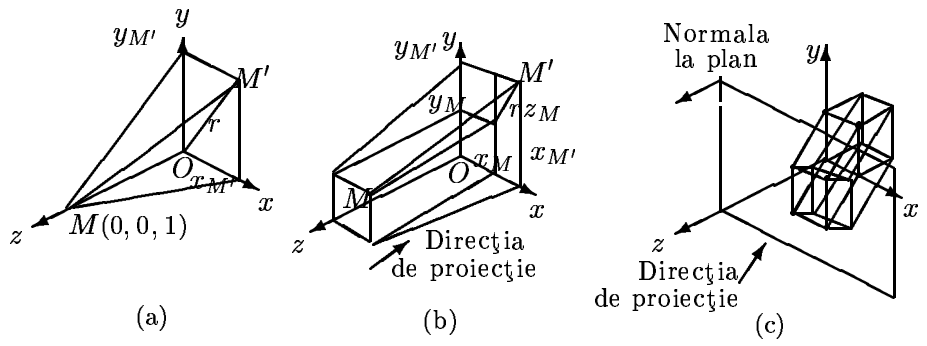


Figura 2.9: Proiecţia oblică (a) Construirea proiecţiei punctului $M(0,0,1)$ pe planul xOy (b) Proiecţia unui punct oarecare pe planul xOy (c) Proiecţia unui cub pe un plan paralel cu axa y

Se convine ca o proiecţie oblică pe planul xOy să fie caracterizată prin punctul în care este proiectat $M(0,0,1)$ pe planul xOy, distanţa r de la origine a noului punct şi unghiul α dintre raza r şi Ox (figura 2.9.a). Funcţie de aceste valori se pot exprima noile coordonate (x', y', z') ale proiecţiei oblice pe planul xOy a unui punct oarecare (x, y, z) (vezi figura 2.9.b):

$$\begin{cases} x' = x + zr\cos\alpha, \\ y' = y + zr\sin\alpha, \\ z' = 0, \end{cases}$$

adică în coordonate omogene,

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & r\cos\alpha & 0 \\ 0 & 1 & r\sin\alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

Proiecţia ortogonală se obţine când $r = 0$.

Cele mai frecvente proiecţii oblice sunt:

(a) proiecţia cavalieră, pentru care $r = 1$;

(b) proiecţia cabinet, pentru care $r = 1/2$.

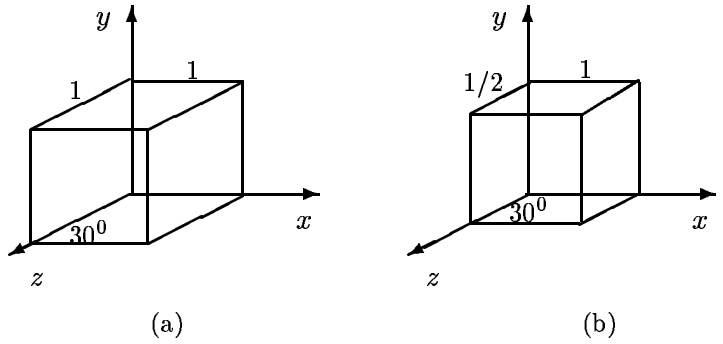


Figura 2.10: Proiecţii oblice ale cubului unitate (a) Proiecţia cavalieră cu direcţia de proiecţie $(\sqrt{3}/2, 1/2, -1)$ (b) Proiecţia cabinet cu direcţia de proiecţie $(\sqrt{3}/4, 1/4, -1)$

În proiecţia cavalieră direcţia de proiecţie face un unghi de 45^0 cu planul de proiecţie, astfel încât proiecţia unui segment de dreaptă perpendicular pe planul de proiecţie are aceeaşi lungime ca şi segmentul însuşi (figura 2.10.a).

În proiecţia cabinet direcţia de proiecţie face un unghi de $\text{arctg}(2) \approx 63.4^0$ cu planul de proiecţie, astfel încât segmentele perpendiculare pe planul de proiecţie se proiectează la 1/2 din lungimea lor reală (figura 2.10.b).

Dintre proiecţiile oblice, proiecţia cabinet oferă imaginile cele mai realiste.

2.5.4 Transformări proiective

Se determină în cele ce urmează matricea 4×4 caracteristică unei transformări proiective oarecare. Presupunem că în proiecţia perspectivă planul de proiecţie este perpendicular pe axa z, iar în proiecţia paralelă, planul de proiecţie este $z = 0$ (planul xOy). Fie $P(x, y, z)$ un punct oarecare al spaţiului şi $P'(x', y', z')$ proiecţia acestuia.

Se consideră următoarele trei cazuri:

1. Centrul de proiecţie perspectivă se află în originea sistemului, iar planul de proiecţie este $z = d$.

Din asemănarea triunghiurilor din figura 2.11 (b) şi (c) rezultă

$$\frac{x'}{d} = \frac{x}{z}, \quad \frac{y'}{d} = \frac{y}{z},$$

adică $x' = xd/z$, $y' = yd/z$, $z' = d$. Astfel, divizarea cu z a coordonatelor în proiecţia perspectivă explică faptul că obiectele mai îndepărtate apar mai mici decât obiectele mai apropiate de centrul de proiecţie. Matricea transformării

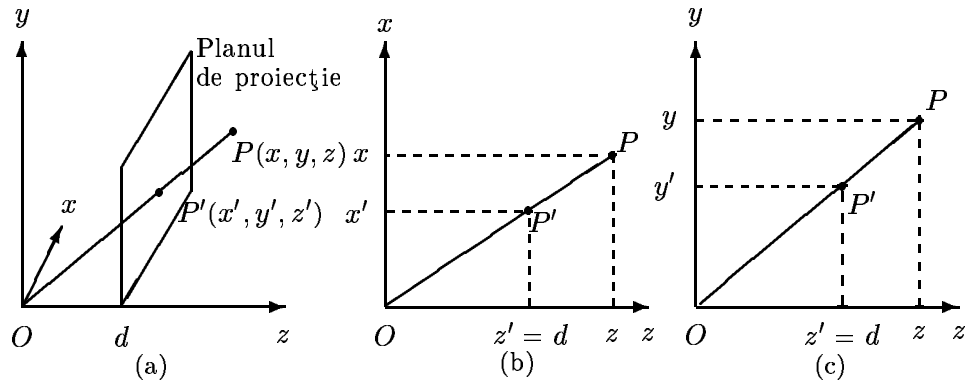


Figura 2.11: Cazul centrului de proiecţie în originea sistemului şi al planului de proiecţie $z = d$ (a) Construcţia proiecţiei (b) Proiecţie pe planul xOz (c) Proiecţie pe planul yOz

este

$$M_{pers} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix}.$$

Multiplicând $(x, y, z, 1)^T$ cu M_{pers} se obţine punctul omogen $(X, Y, Z, W)^T = (x, y, z, z/d)^T$. Coordonatele carteziene asociate punctului omogen se obţin prin diviziune cu W, adică $(x', y', z') = (X/W, Y/W, Z/W) = (xd/z, yd/z, d)$.

2. Centrul de proiecţie se află la $z = -d$, iar planul de proiecţie este $z = 0$. Din asemănarea triunghiurilor din figura 2.12 se obţine

$$\frac{x'}{d} = \frac{x}{z+d}, \quad \frac{y'}{d} = \frac{y}{z+d},$$

adică $x' = xd/(z+d)$, $y' = yd/(z+d)$, $z' = 0$. Matricea transformării este

$$M_{pers}{}' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix}.$$

Spre deosebire de cazul anterior, această formulare permite includerea proiecţiilor paralele: dacă $1/d \to 0$, se obţine proiecţia ortografică pe planul $z = 0$, de ecuaţii $x' = x$, $y' = y$, $z' = 0$.

3. Planul de proiecţie este $z = z_p$, iar centrul de proiecţie se află la distanţa Q de punctul $(0, 0, z_p)$ în direcţia vectorului normalizat (dx, dy, dz) (figura 2.13).

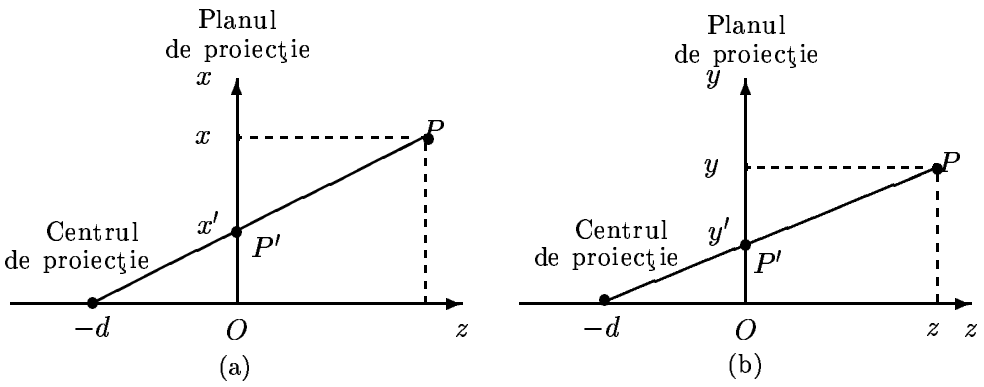


Figura 2.12: Cazul centrului de proiecţie la $z = -d$ şi al planului de proiecţie $z = 0$ (a) Proiecţie pe planul xOz (c) Proiecţie pe planul yOz

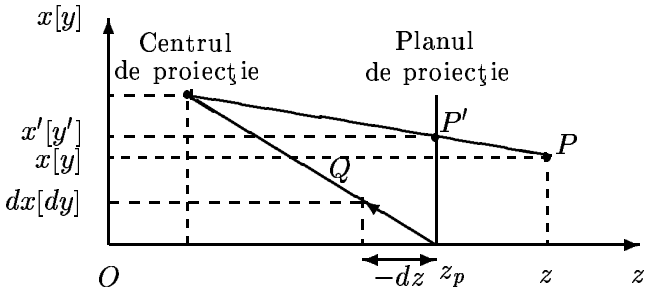


Figura 2.13: Cazul centrului de proiecţie la distanţa Q de $(0, 0, z_p)$ în direcţia (dx, dy, dz) şi planul de proiecţie $z = z_p$

Dreapta care uneşte centrul de proiecţie C_p şi punctul P poate fi descrisă parametric: $C_p + t(P - C_p)$, $t \in R$. Punctul P' se află pe această dreaptă. Cum $C_p = (0, 0, z_p) + Q(dx, dy, dz)$,

$$\begin{cases} x' = Qdx + t(x - Qdx), \\ y' = Qdy + t(y - Qdy), \\ z' = (z_p + Qdz) + t[z - (z_p + Qdz)]. \end{cases}$$

Cum P' se află în planul $z' = z_p$, se obţine t:

$$t = \frac{z_p - (z_p + Qdz)}{z - (z_p + Qdz)}.$$

Astfel

$$x' = \frac{x - z\dfrac{dx}{dz} + z_p\dfrac{dx}{dz}}{\dfrac{z_p - z}{Qz_p} + 1}, \quad y' = \frac{y - z\dfrac{dy}{dz} + z_p\dfrac{dy}{dz}}{\dfrac{z_p - z}{Qz_p} + 1},$$

$$z' = z_p\frac{\dfrac{z_p - z}{Qdz} + 1}{\dfrac{z_p - z}{Qdz} + 1} = \frac{-z\dfrac{z_p}{Qdz} + \dfrac{z_p^2 + z_pQdz}{Qdz}}{\dfrac{z_p - z}{Qdz} + 1}.$$

Matricea transformării este constituită astfel încât ultima linie înmulţită cu $(x, y, z, 1)^T$ să producă coordonata omogenă W egală cu numitorul comun al fracţiilor de mai sus:

$$M_{general} = \begin{pmatrix} 1 & 0 & -\frac{dx}{dz} & z_p\frac{dx}{dz} \\ 0 & 1 & -\frac{dy}{dz} & z_p\frac{dy}{dz} \\ 0 & 0 & -\frac{z_p}{Qdz} & \frac{z_p^2}{Qdz} + z_p \\ 0 & 0 & -\frac{1}{Qdz} & \frac{z_p}{Qdz} + 1 \end{pmatrix}.$$

Matricele cazurilor anterioare se pot regăsi în această reprezentare. Astfel,

- pentru M_{pers}, $z_p = d$, $Q = d$, $(dx, dy, dz) = (0, 0, -1)$;
- pentru M'_{pers}, $z_p = 0$, $Q = d$, $(dx, dy, dz) = (0, 0, -1)$;
- pentru proiecţia ortogonală pe $z = 0$, $z_p = 0$, $Q = \infty$, $(dx, dy, dz) = (0, 0, -1)$;
- pentru proiecţia cavalieră pe $z = 0$, $z_p = 0$, $Q = \infty$, $(dx, dy, dz) = (\cos\alpha, \sin\alpha, -1)$;
- pentru proiecţia cabinet pe $z = 0$, $z_p = 0$, $Q = \infty$, $(dx, dy, dz) = (1/2\ \cos\alpha, 1/2\ \sin\alpha, -1)$.

Când $Q \neq \infty$, $M_{general}$ defineşte o proiecţie perspectivă cu un punct de fugă. Punctul de fugă se obţine multiplicând punctul de la infinit de pe axa z, reprezentat în coordonate omogene prin $(0, 0, 1, 0)^T$, cu $M_{general}$. Se obţine $x' = Qdx$, $y = Qdy$, $z' = z_p$.

2.5.5 Exemple

Pentru definirea unei proiecţii s-au convenit o serie de notaţii, specificate în cele ce urmează.

Planul de proiecţie este definit printr-un punct de referinţă al planului P_r şi normala la plan, vectorul n.

Se construieşte sistemul de vedere de referinţă (asociat planului de proiecţie şi normalei la acesta) astfel:

(a) originea sistemului este P_r;
(b) una din axe este n;
(c) o altă axă este vectorul v, care este proiecţia pe planul de proiecţie a verticalei imaginii (proiecţia axei y din sistemul lumii reale);

(d) a treia axă, u, este determinată astfel încât $u \times v = n$.
Planul de proiecţie este astfel uP_rv.

Fereastra dreptunghiulară din planul de proiecţie este definită relativ la axele de coordonate u şi v, nu neapărat simetric faţă de P_r. Se notează centrul ferestrei cu C_f.

Centrul de proiecţie C_p şi direcţia de proiecţie D_p sunt definite printr-un punct de referinţă al proiecţiei P_d şi un indicator al tipului de proiecţie. Dacă proiecţia este perspectivă, $C_p = P_d$. Dacă proiecţia este paralelă, D_p este definită de vectorul cu originea în P_d şi capătul în C_f.

Specificările clasice ale proiecţiilor în notaţiile de mai sus sunt următoarele:

1. P_r= originea sistemului de coordonate ale lumii reale, xyz;
2. n= axa z;
3. v= axa y;
4. fereastra: $[0,1] \times [0,1]$ în planul uP_rv;
5. P_d=$(0.5, 0.5, 1)$ în sistemul uvn.

De exemplu, parametrii proiecţiei paralele corespunzătoare (proiecţie pe xOy) sunt

$$P_r = (0,0,0)\ (xyz), \quad n = (0,0,1)\ (xyz), \quad v = (0,1,0)\ (xyz),$$

$$P_d = (0.5, 0.5, 1)\ (uvn), \quad \text{Fereastra} = (0,1,0,1)\ (uvn), \quad \text{Proiecţie: paralelă}$$

Se consideră exemplul casei din figura 2.14.

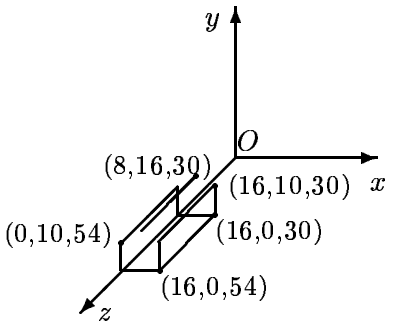


Figura 2.14: Coordonatele definitorii ale unui schelet de casă ce urmează a fi proiectat

O proiecţie perspectivă pe un plan ce taie axa z este prezentată în figura 2.15.

Proiecţia paralelă pe planul $z = 0$ produce imaginea din figura 2.16.

În figura 2.17 sunt trasate alte trei proiecţii ale casei.

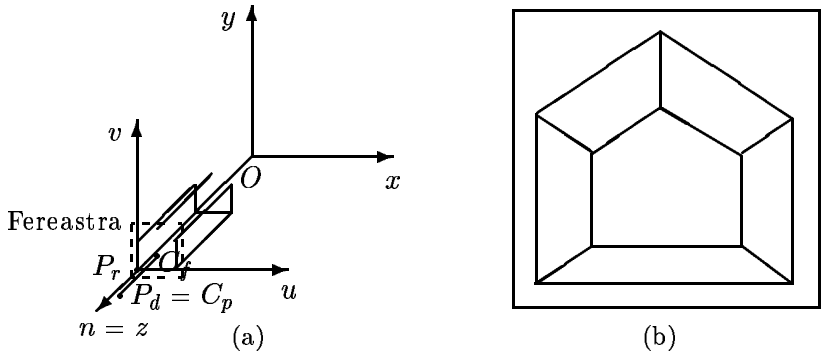


Figura 2.15: Proiecţia perspectivă a casei cu parametrii: $P_r = (0, 0, 54)$, $n = (0, 0, 1)$, $v = (0, 1, 0), P_d = (8, 6, 30)$, Fereastra=$(-1, 17, -1, 17)$, Proiecţie: perspectivă (a) Elementele caracteristice proiecţiei (b) Imaginea rezultată

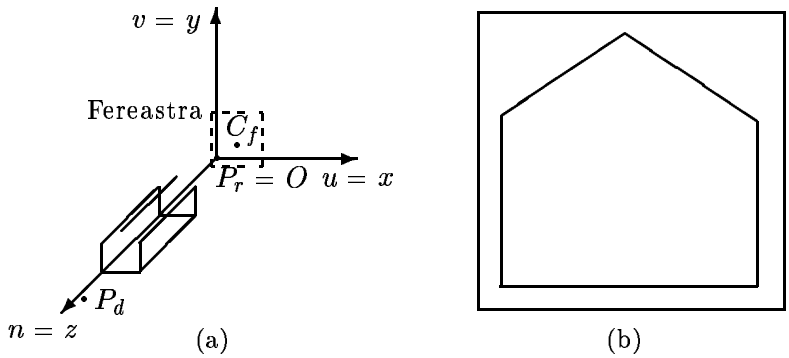
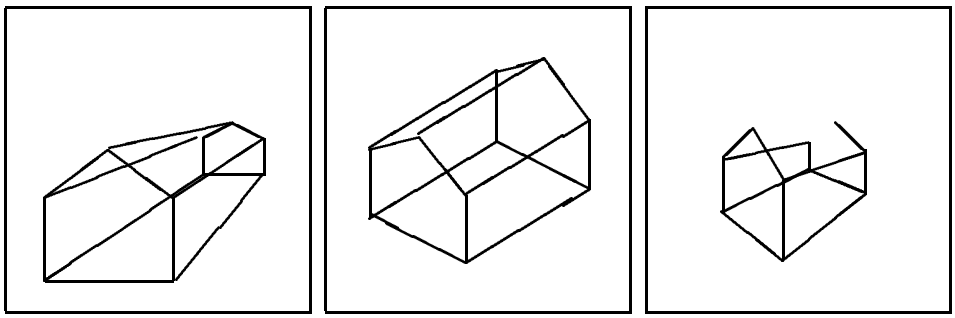


Figura 2.16: Proiecţia paralelă a casei cu parametrii: $P_r = (0, 0, 0)$, $n = (0, 0, 1)$, $v = (0, 1, 0), P_d = (8, 8, 100)$, Fereastra=$(-1, 17, -1, 17)$, Proiecţie: paralelă (a) Elementele caracteristice proiecţiei (b) Imaginea rezultată



(a) (b) (c)

Figura 2.17: Proiecţii cu parametrii (a) $P_r = (16, 0, 54)$, $n = (0, 0, 1)$, $v = (0, 1, 0), P_d = (20, 25, 20)$, Fereastra$=(-20, 20, -5, 35)$, Proiecţie: perspectivă (cu un punct de fugă) (b) $P_r = (8, 8, 42)$, $n = (1, 1, 1)$, $v = (0, 1, 0), P_d = (0, 0, 10)$, Fereastra$=(-20, 20, -20, 20)$, Proiecţie: paralelă (izometrică) (c) $P_r = (16, 0, 54)$, $n = (1, 0, 1)$, $v = (0, 1, 0), P_d =(0,25,20\sqrt{2})$, Fereastra$=(-20, 20, -5, 35)$, Proiecţie: perspectivă (cu două puncte de fugă)

2.6 Volumul de vedere

Ochiul uman poate observa toate obiectele situate în interiorul unui con de vedere. Directoarea acestui con situată într-un plan normal pe direcţia de observare este eliptică. În aplicaţiile grafice se înlocuieşte conul de vedere cu o piramidă de vedere. Astfel, se poate imagina că observatorul priveşte lumea printr-o fereastră dreptunghiulară decupată într-un plan opac, situată la o anumită distanţă de observator.

Dacă pentru construirea unei imagini în două dimensiuni sunt necesare specificarea unei ferestre şi a unei zone de lucru, în trei dimensiuni sunt indicate un volum de vedere, un tip de proiecţie pe un plan, o fereastră în acel plan şi o zonă de lucru pe suprafaţa de vizualizare. Conţinutul volumului de vedere este proiectat în fereastra din planul de proiecţie şi apoi este transferat în zona de lucru.

Volumul de vedere mărgineşte acea porţiune din spaţiul lumii reale care va fi proiectat pe planul de proiecţie. Este definit astfel:

1. în proiecţia perspectivă, ca o piramidă semi-infinită cu vârful în centrul de proiecţie şi cu laturile trecând prin colţurile ferestrei din planul de proiecţie;
2. în proiecţia paralelă, ca un paralelipiped infinit cu laturile paralele cu direcţia de proiecţie.

Pentru a limita numărul de primitive grafice proiectate în planul de proiecţie, aceste volume sunt transformate în corpuri "finite" prin introducerea a două

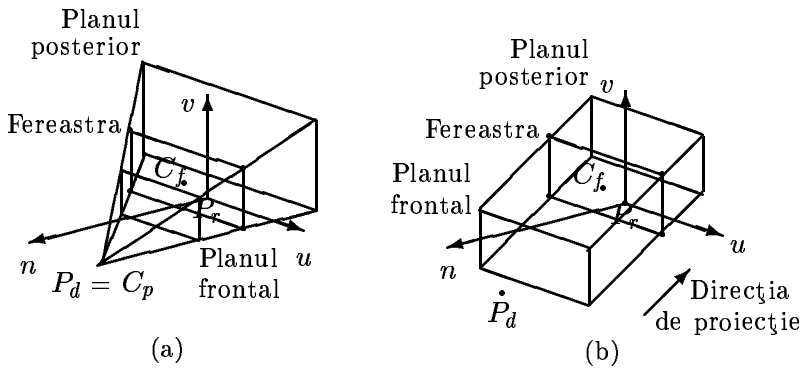


Figura 2.18: Volumul de vedere (a) Cazul proiecţiei perspective (b) Cazul proiecţiei paralele

plane de limitare, paralele cu planul de proiecţie, în faţa şi în spatele acestuia, aflate la distanţe bine definite de-a lungul normalei la planul de proiecţie (figura 2.18). Se obţine un trunchi de piramidă, respectiv un paralelipiped.

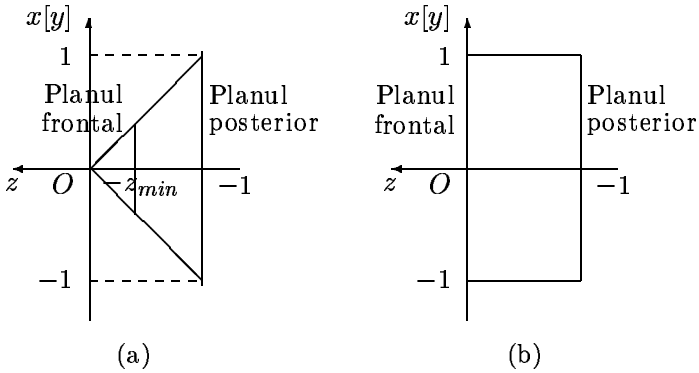


Figura 2.19: Proiecţii ale volumului de vedere canonic (a) Cazul proiecţiei perspective (b) Cazul proiecţiei paralele

În urma transformării de normalizare a volumului de vedere, planele ce definesc frontiera noului volum, numit volum de vedere canonic, sunt:

1. pentru proiecţia paralelă, $x = -1$, $x = 1$, $y = -1$, $y = 1$, $z = 0$, $z = -1$ (figura 2.19.a);
2. pentru proiecţia perspectivă, $x = z$, $x = -z$, $y = z$, $y = -z$, $z = -z_{min}$, $z = -1$ (figura 2.19.b).

Volumul de vedere canonic din proiecţia perspectivă poate fi transformat în volumul de vedere canonic din proiecţia paralelă astfel: unui punct (x, y, z) din interiorul trunchiului de piramidă îi corespunde un punct (x', y', z') din

interiorul paralelipipedului prin relaţiile

$$x' = -\frac{x}{z}, \quad y' = -\frac{y}{z}, \quad z' = \frac{z_{min} - z}{z_{min} + 1}.$$

Matricea corespunzătoare indică o transformare perspectivă. Astfel, pentru construirea imaginii prin proiecţie perspectivă a unui corp 3D se poate proceda în două moduri:

(a) se proiectează fiecare punct component, noile componente fiind stocate separat;

(b) se aplică corpului transformarea perspectivă care îl deformează astfel încât proiecţia paralelă a corpului obţinut să coincidă cu proiecţia perspectivă a obiectului iniţial. În acest fel imaginea va fi stocată în memorie prin chiar coordonatele x şi y ale punctelor implicate. Se proiectează ortogonal corpul obţinut.

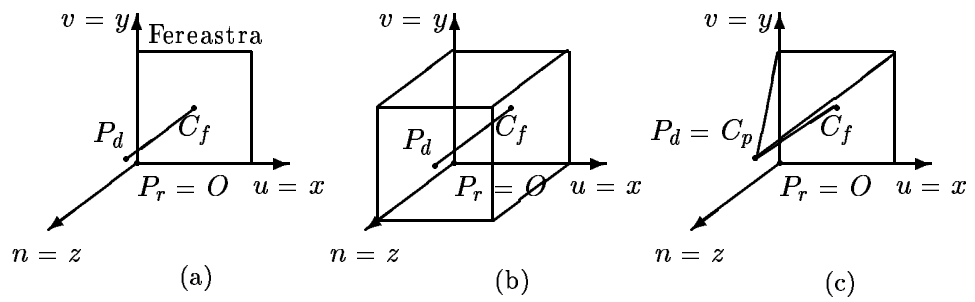


Figura 2.20: (a) Elementele caracteristice ale proiecţiei în specificarea clasică (b) Volumul de vedere în proiecţia paralelă (c) Volumul de vedere în proiecţia perspectivă

Volumele de vedere asociate specificării clasice sunt prezentate în figura 2.20.

Capitolul 3

Reprezentări simple ale imaginii

3.1 Primitive grafice

Se numesc primitive grafice acele elemente de bază pe care programatorul le poate folosi pentru a realiza desenele necesare unei anumite aplicaţii. Primitivele grafice pot fi puncte, segmente de dreaptă, caractere, dreptunghiuri, conice, curbe de tip spline, simboluri speciale, etc.

Se consideră un dispozitiv de afişare grafică cu tub catodic cu rastru dreptunghiular, cu originea (0,0) în punctul din stânga jos al ecranului, cu valorile absciselor crescând la dreapta şi valorile ordonatelor crescând în sus. Unităţile axelor u, v sunt egale cu distanţele dintre doi pixeli alăturaţi pe orizontală, respectiv verticală. Se presupune de asemenea că fiecare pixel de pe ecran are două stări posibile: aprins sau stins (imagine monocromă). Se neglijează, în acest capitol, celelalte atribute posibile pentru reprezentarea unei imagini pe ecran, cum sunt culoarea, nuanţa, intensitatea, etc.

3.2 Punctul

Pentru generarea efectivă a unui punct pe ecran se consideră diponibilă operaţia

$$\text{plot } (u, v),$$

unde plot este o subrutină care depinde de configuraţia hardware a dispozitivului de afişare, iar u, v sunt coordonatele întregi în spaţiul de afişare grafică ale punctului care trebuie generat. Pentru aceste coordonate se utilizează denumirea de variabile de adresare fizică.

Dacă nu se dispune de o funcţie de tip plot, se poate obţine efectul dorit folosind adresarea directă a memoriei. Se consideră un display monocrom de rezoluţie $m \times n$ (coloane $\times$ linii) a cărui origine are asociat primul bit al octetului de la adresa a (m este în general divizibil cu 8). Se presupune că memoria video este liniară, deci biţii corespunzători punctelor se află în ordinea citirii acestora din urmă pe linii. Pentru aprinderea pixelului $P(u, v)$ se realizează următoarele:

1. test iniţial care ne asigură încadrarea în spaţiul adresabil (de forma $m_a \times n_a$): $0 \leq u \leq m_a - 1$, $0 \leq v \leq n_a - 1$.
2. se calculează adresa relativă a_r a octetului în care se află bitul corespunzător pixelului P, apoi adresa absolută a acestuia, a_a: $a_r = v \cdot m/8 + [u/8]$, $a_a = a + a_r$.
3. se calculează poziţia bitului în octet: $b_r = u - 8[u/8] + 1$ (biţii unui octet sunt numerotaţi de la 1 la 8);
4. se setează bitul pe 1.

Dacă se doreşte afişarea simultană a unor puncte dintr-o gamă variată de culori, atunci sunt necesari mai mulţi biţi pentru reprezentarea unui punct al ecranului.

3.3 Trasarea incrementală

O atenţie deosebită s-a acordat optimizării algoritmilor de trasare a primitivelor în ceea ce priveşte eficienţa de execuţie (viteza de generare a unei primitive). În acest scop se urmăreşte reducerea numărului de operaţii în virgulă mobilă, în favoarea celor cu numere întregi şi minimizarea numărului de înmulţiri şi împărţiri (microprocesoarele actuale nu conţin instrucţiuni pentru aritmetică în virgulă mobilă, astfel încât instrucţiunile de înmulţire şi împărţire sunt mai lente decât operaţiile de adunare şi scădere).

Metodele de trasare incrementală se bazează pe scheme cu diferenţe între mărimi asociate punctelor succesive, care conduc la relaţii liniare de transformare iterativă a unor variabile. Aceste relaţii liniare pot fi implementate în general prin operaţii simple de adunare şi scădere de numere întregi. Metodele incrementale îşi dovedesc eficienţa în special la trasarea curbelelor de grad unu (linii drepte) şi doi (cercuri, elipse, hiperbole, parabole).

Algoritmii de trasare incrementală a curbelor de grad cel mult doi pornesc de la anumiţi parametri care caracterizează analitic curba şi construiesc cea mai bună reprezentare discretizată a curbei respective. Se efectuează deplasări elementare între două puncte vecine pe orizontală, verticală sau diagonală, care se traduc prin operaţii simple de incrementare sau decrementare a variabilelor de adresare fizică. Algoritmul de trasare incrementală păstrează în permanenţă informaţii cu privire la starea curentă a procesului iterativ, prin intermediul uneia sau mai multor variabile de stare. Acestea permit realizarea cât mai simplă a funcţiilor algoritmului şi sunt uşor actualizabile în vederea trecerii la următorul pas iterativ.

Ţinând seama de faptul că modelul reprezentării este o curbă continuă pentru care tangenta variază continuu, pe când spaţiul de afişare este discret, în dezvoltarea algoritmilor s-au adoptat următoarele convenţii metodologice de reprezentare:

1. pentru fiecare punct generat la un capăt al curbei, sau la o margine a ferestrei de vizualizare, există exact un punct vecin pe orizontală, verticală sau diagonală, generat pe curbă.
2. orice alt punct generat pe curbă are exact două puncte vecine pe orizontală, verticală sau diagonală, generate pe curbă, cu care formează un

unghi de 180^0 sau 135^0 (se evită unghiurile ascuţite în reprezentările discrete ale curbelor) pentru un raport aspectual unitar.

Aceste reguli sunt referite sub numele de reguli de conexiune discretă.

3.4 Trasarea liniilor

Pentru a trasa un segment de dreaptă trebuie cunoscute coordonatele capetelor acestuia. Deoarece un segment se trasează în timp, cele două puncte pot fi denumite extremitatea de început şi extremitatea de sfârşit a segmentului.

Coordonatele punctului de început, fie $A(x_A, y_A)$, pot fi transmise în două moduri:

(a) specificarea coordonatelor;

(b) identificare cu coordonatele poziţiei cursorului grafic curent (ultimul punct desenat).

Coordonatele punctului de sfârşit $B(x_B, y_B)$ pot fi specificate în mod:

(a) absolut: se specifică x_B, y_B,

(b) relativ: se specifică valorile $x_B - x_A$, $y_B - y_A$.

Pentru trasarea segmentului pe ecran, numită conversie prin baleiaj (baleiere, scanare) sau conversie scan, se determină care sunt pixelii rastrului cei mai apropiaţi de imaginea ideală a segmentului (care vor fi aprinşi). Segmentul obţinut va arăta ca o scăriţă (figura 3.1). Un pixel este reprezentat printr-un disc centrat în punctul de coordonate întregi (u, v) (în sistemele actuale diametrul unui pixel circular este mai mare decât spaţiul interpixeli, deci reprezentarea din figură este exagerată). Segmentul de linie trasat are "lăţimea" de un pixel.

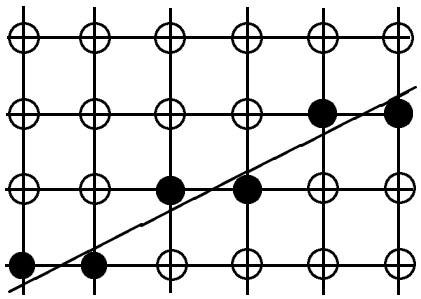


Figura 3.1: Un segment de dreaptă convertit prin baleiaj, pentru care pixelii aprinşi sunt reprezentaţi prin cercuri negre

3.4.1 Algoritmul incremental de bază

Algoritmul incremental de bază sau algoritmul DDA (Digital Differential Analyzer) determină poziţia pixelilor cei mai apropiaţi de un segment de linie,

parcurgând paşi unitari pe o axă de coordonate şi calculând valorile corespunzătoare pe cealaltă axă. Fie xOy sistemul de coordonate ale dispozitivului de afişare.

În planul ecranului, ecuaţia dreptei care trece prin A şi B este

$$y = y_A + m(x - x_A), \quad m = \frac{y_B - y_A}{x_B - x_A},$$

unde m reprezintă panta dreptei. Se presupune că coordonatele punctelor extreme, x_i, y_i, $i = A, B$, sunt numere întregi. Altfel, se rotunjesc la cel mai apropiat întreg.

Dacă $x_B > x_A$, atunci pornind din x_A şi crescând pe x cu câte o unitate (incrementând cu un pixel pe orizontală), se calculează y pentru fiecare x, se rotunjeşte la cel mai apropiat întreg (Round(a)=$[a + 0.5]$) şi se apelează de fiecare dată funcţia plot$(x, \text{Round}(y))$ (figura 3.2.a). Se trece la următoarea valoare a lui x, până când x depăşeşte pe x_B.

Dacă la un moment dat intersecţia cu o verticală a grilei este (x_i, y_i) şi se aprinde pixelul $(x_i, \text{Round}(y_i))$, atunci la pasul următor intersecţia cu următoarea verticală va fi (x_{i+1}, y_{i+1}), unde $x_{i+1} = x_i + 1$, iar

$$y_{i+1} = \text{Round}(y_A + m(x_{i+1} - x_A)) = \text{Round}(y_A + m(x_i + 1 - x_A)) = \text{Round}(y_i + m).$$

Se aprinde pixelul (x_{i+1}, y_{i+1}).

Pentru a proceda uniform indiferent de relaţia de ordine dintre x_A şi x_B, se calculează $s = \text{sgn}(x_B - x_A)$ şi formula de calcul a lui y rămâne neschimbată, doar că x se incrementează cu s în loc de 1.

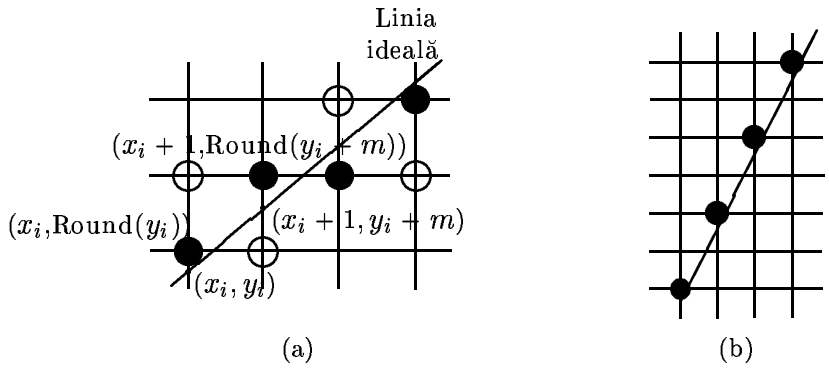


Figura 3.2: Algoritmul DDA (a) Calculul incremental pentru o pantă subunitară (b) Cazul incrementării unitare pe axa x cu $|m| > 1$

Folosind acest raţionament, segmentele mai apropiate de verticală decât de orizontală, ar apărea punctate (figura 3.2.b), deoarece panta $|m| > 1$, iar segmentele verticale nu ar putea fi reprezentate deoarece $x_B - x_A = 0$. Astfel se face distincţie în algoritm între segmentele apropiate de orizontală şi cele apropiate de verticală, pentru acestea din urmă calculându-se x în funcţie de y şi realizând incrementarea după y cu $s = \text{sgn}(y_B - y_A)$ între y_A şi y_B.

3.4.2 Algoritmul punctului de mijloc

În algoritmul DDA ponderea cea mai mare o ocupă în timp calculele în virgulă flotantă (funcţia Round). Din punct de vedere al timpului, mai eficienţi sunt algoritmii care elimină operaţiile în virgulă mobilă. Un exemplu tipic de algoritm în aritmetica numerelor întregi pentru trasarea segmentelor de dreaptă este algoritmul Bresenham (algoritmul punctului mijlociu).

Pentru exemplificare, se consideră cazul unui segment al cărui înclinare este cuprinsă între 0^0 şi 45^0 (panta $m \in (0,1)$). Conform regulilor de conexiune discretă, cunoscând poziţia curentă a unui punct oarecare de pe segment, există două puncte posibile pentru următoarea poziţie (figura 3.3).

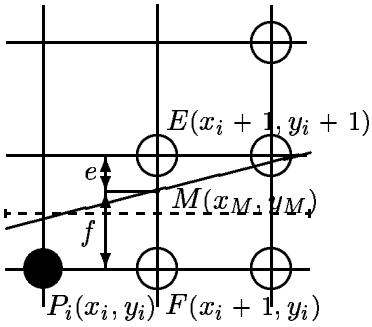


Figura 3.3: Algoritmul Bresenham

Se consideră că la pasul i s-a determinat pixelul care trebuie aprins ca fiind $P(x_i, y_i)$. La pasul $i+1$ este necesară alegerea unui punct dintre $E(x_i+1, y_i+1)$ şi $F(x_i+1, y_i)$. Criteriul de decizie pentru algoritm este apropierea punctului faţă de segmentul real. Practic se compară distanţele e şi f din figura 3.3. Dacă $f < e$, se alege F, altfel se alege E.

Pentru stabilirea criteriului decizional se parcurg următoarele etape:

Pas 0: Se compară x_A cu x_B şi eventual se face o schimbare între capetele segmentului astfel încât $x_A < x_B$. Se realizează o translaţie cu deplasamentul $(-x_A, -y_A)$ astfel încât ecuaţia dreptei modificate să fie $y = (dy/dx)x$, unde $dx := x_B - x_A$, $dy := y_B - y_A$. Se consideră $x_0 = 0$, $y_0 = 0$.

Pasul i: Punctul M de intersecţie al dreptei $x = x_i+1$ cu dreapta ce se trasează are coordonatele $x_M = x_i + 1, y_M = (dy/dx)x_M$. Se determină distanţele la punctele din rastru cele mai apropiate pe verticala $x = x_i + 1$: $f = y_M - y_i$, $e = y_i + 1 - y_M$. Se determină cantitatea $dx(f - e)$:

$$dx(f - e) = dx(2y_M - 2y_i - 1) =$$

$$= dx\left[2\frac{dy}{dx}(x_i + 1) - 2y_i - 1\right] = 2dy(x_i + 1) - 2y_i dx - dx.$$

Deoarece $dx > 0$ (din pasul 0) cantitatea $f - e$ are acelaşi semn ca $dx(f - e)$. Se notează $d_{i+1} = dx(f - e)$ şi se caută o relaţie recursivă pentru acest factor de decizie. Cum

$$d_{i+1} = 2(x_i dy - y_i dx) + 2dy - dx, \qquad d_i = 2(x_{i-1} dy - y_{i-1} dx) + 2dy - dx,$$

rezultă

$$d_{i+1} - d_i = 2[dy(x_i - x_{i-1}) - dx(y_i - y_{i-1})]$$

şi fiindcă $x_i - x_{i-1} = 1$ se obţine relaţia recursivă

$$d_{i+1} = d_i + 2dy - 2dx(y_i - y_{i-1}),$$

cu valoarea de start

$$d_1 = 2(x_0 dy - y_0 dx) + 2dy - dx = 2dy - dx.$$

În concluzie, criteriul decizional este următorul: dacă $d_{i+1} \geq 0$ se alege $E(x_i + 1, y_i + 1)$ şi $d_{i+2} = d_{i+1} + 2(dy - dx)$, iar dacă $d_{i+1} < 0$, atunci se alege $F(x_i + 1, y_i)$ şi $d_{i+2} = d_{i+1} + 2dy$.

Algoritmul general presupune mai multe cazuri, funcţie de înclinarea dreptei care trebuie trasată.

Observaţie. Algoritmul poartă numele de algoritmul punctului mijlociu deoarece variabila decizională poate fi exprimată liniar funcţie de ecuaţia dreptei evaluată în punctul de mijloc al segmentului EF. Fie $D(x, y) = ax + by + c$, $a > 0$ ecuaţia dreptei. Variabila de decizie este $d_{i+1} = 2D(x_i + 1, y_i + \frac{1}{2})$. Cum $D(x, y)$ este negativ pentru un punct de deasupra liniei şi pozitiv pentru un punct sub linie, se alege E dacă $d_{i+1} \geq 0$ şi F dacă $d_{i+1} < 0$. Dacă este ales F, atunci

$$d_{i+2} = 2D(x_i + 2, y_i + 1/2) = 2[a(x_i + 2) + b(y_i + 1/2) + c] = d_{i+1} + 2a = d_{i+1} + 2dy,$$

iar dacă se alege E, atunci

$$d_{i+2} = 2D(x_i + 2, y_i + 3/2) = d_{i+1} + 2(a + b) = d_{i+1} + 2(dy - dx).$$

Valoarea de start este

$$d_1 = 2D(x_A + 1, y_A + 1/2) = 2D(x_A, y_A) + 2a + b = 2a + b = 2dy - dx,$$

deoarece se presupune că (x_A, y_A) se află pe dreaptă.

3.4.3 Intersecţia segmentelor de linii în rastru

Intersecţia a două drepte neconfundate şi neparalele în plan este un punct. Pe un display rastru, acest principiu matematic nu este valabil. Datorită aproximaţiilor introduse la trasarea segmentelor este perfect posibil ca intersecţia a două segmente să fie ea însăşi un segment, adică o mulţime de pixeli care pe ecran aparţin ambelor segmente.

Sunt posibile următoarele situaţii:

(a) dacă ambele segmente sunt mai apropiate de orizontală decât de verticală, atunci segmentul de intersecţie are direcţia orizontală;
(b) dacă ambele segmente sunt mai apropiate de verticală decât de orizontală, atunci segmentul de intersecţie are direcţia verticală;
(c) dacă un segment este mai apropiat de verticală, celălalt de orizontală, sau cel puţin un segment la 45^0 sau 135^0 faţă de orizontală, atunci intersecţia constă dintr-un singur punct.

O tehnică des utilizată în prelucrarea reprezentărilor simple este transpunerea segmentelor pe ecran prin baleiere cu linii orizontale. Aceasta presupune reconstituirea liniilor orizontale de pixeli de pe ecran, pe rând, redând elementele din desen conţinute în fiecare linie. Pentru o rezoluţie $m \times n$, o linie de baleiaj este privită ca un segment orizontal situat la cota y_{bal} între $(0, y_{bal})$ şi $(m-1, y_{bal})$. Un segment oarecare se transpune pe linia de baleiaj în pixelii care alcătuiesc intersecţia dintre acesta şi segmentul de baleiaj. Transpunerea segmentelor se face astfel: se determină mulţimea pixelilor care formează intersecţia dintre segmentul de baleiaj şi segmentul de transpus, apoi, pe linia respectivă, se aprind pixelii atunci când baleiajul ajunge "în" intersecţie. Intersecţia segmentului cu linia de baleiere se poate determina (pentru segmente scurte) şi prin reconstituirea algoritmului de desenare a segmentului şi marcarea momentelor de "intrare" şi "ieşire" din intersecţie, determinate pe baza verificării cotei.

3.4.4 Atenuarea efectelor datorate discretizării imaginii

Se consideră disponibil un display care are cel puţin două nivele de intensitate. Liniile apar în rastru ca nişte scăriţe. Acest fenomen este rezultatul conversiei scan de tipul tot sau nimic (un pixel este aprins sau stins). Aplicarea tehnicilor care reduc sau elimină fenomenul sunt referite ca antialiasing.

O primă metodă este creşterea rezoluţiei, respectiv utilizarea unui display cu o rezoluţie superioară. O altă tehnică este cea a pixelilor în diferite stadii. Salturile în liniile de pe ecran sunt ajustate prin mutarea cu o micropoziţie a unor pixeli. Sistemele care permit o asemenea tehnică sunt construite astfel încât poziţii individuale de pixeli pot fi deplasate cu o fracţiune din latura pixelului, de obicei 1/4, 1/2 sau 3/4. Aceste două metode diminuează efectul de scăriţă, dar nu îl elimină.

O metodă des utilizată pe display-urile care permit mai multe nivele de intensitate este metoda ariilor. Ideea de bază este aceea că punctele şi liniile de pe un ecran au dimensiuni finite. Se consideră că un pixel este aproximativ un pătrat, iar o linie are grosimea cel puţin egală cu latura pătratului-pixel (figura 3.4a). În loc de a trasa linia cu un singur pixel corespunzător fiecărei poziţii pe axa Ox, toţi pixelii peste care se suprapune o parte din "aria" liniei, sunt aprinşi cu o intensitate proporţională cu aria de suprapunere (metoda ariilor fără greutate). Această tehnică are trei proprietăţi:

1. intensitatea unui pixel care intersectează o latură descreşte proporţional cu creşterea distanţei dintre centrul pixelului şi segmentul de linie;

2. primitiva de trasare a segmentului nu influenţează intensitatea unui pixel dacă nu intersectează aria acestuia;
3. arii egale contribuie egal la intensitate.

Metoda ariilor cu greutate păstrează primele două proprietăţi, dar nu şi a treia: arii egale pot contribui inegal la intensitate, deoarece o arie mică apropiată de centrul pixelului are o mai mare influenţă decât o arie egală aflată la o distanţă mai mare. Aria de influenţă a unui pixel corespunde cercului determinat de centrele celor patru pixeli vecini mai apropiaţi.

Corectarea intensităţii pixelilor se utilizează pentru anularea mai multor efecte a trasării liniilor pe rastru. Astfel, o linie orizontală de aceeaşi lungime cu una înclinată va apărea mai luminată decât cea din urmă, chiar dacă numărul de pixeli este identic. Explicaţia constă în faptul că intensitatea per unitate de lungime (geometrică) este mai redusă în cazul segmentului înclinat. În figura 3.4 (b) segmentul CB are panta 1 şi este de $\sqrt{2}$ ori mai lung decât segmentul orizontal CA, deşi au acelaşi număr de pixeli în reprezentarea discretă. Dacă intensitatea per pixel este I, intensitatea per unitatea de lungime a segmentului CA este I, iar a segmentului CB este $I/\sqrt{2}$. Pentru compensarea acestui efect nedorit, se ajustează intensitatea de trasare a fiecărei linii în funcţie de panta acesteia. Liniile verticale şi orizontale sunt trasate cu intensitate redusă, în timp ce liniile înclinate la 45^0, cu intensitate maximă.

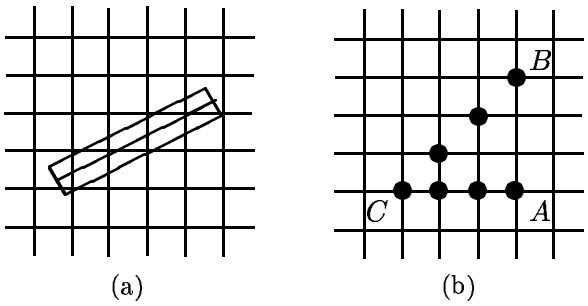


Figura 3.4: (a) Metoda ariilor (b) Variaţia intensităţii funcţie de pantă

Tehnica antialiasing este utilizată pentru obţinerea unor imagini de calitate. Datorită efortului de calcul implicat, tehnica este utilizată de obicei doar în ultimul pas al creării unei imagini.

3.5 Trasarea poligoanelor

Fiind dat un poligon prin coordonatele extremităţilor segmentelor care îl alcătuiesc (deci coordonatele vârfurilor), la convertirea scan a acestuia trebuie determinate, pentru fiecare linie de baleiaj care taie poligonul, intersecţiile liniei de baleiaj cu laturile poligonului.

Algoritmul de conversie scan a unui poligon constă în următoarele etape:

— se determină valoarea maximă a ordonatelor y_{max} şi valoarea minimă a ordonatelor y_{min};
— se baleiază poligonul cu y_{bal} de la y_{min} la y_{max}. Pentru fiecare linie de baleiaj se parcurg toate laturile poligonului. Dacă linia de baleiaj taie latura curentă:
  (a) se calculează intersecţia dintre latură şi linia de baleiaj, reţinând începutul x_{min} şi sfârşitul acesteia x_{max};
  (b) pentru x de la x_{min} la x_{max} se aprinde pixelul de coordonate (x, y_{bal});
  Se trece la următoarea latură.

O serie de alţi algoritmi au fost construiţi pentru trasări mai rapide. Unii dintre aceştia sunt menţionaţi în secţiunea referitoare la umplerea poligoanelor din capitolul următor.

3.6 Trasarea cercurilor şi a elipselor

Se consideră un cerc centrat în origine, de ecuaţie

$$x^2 + y^2 = R^2,$$

unde raza R este un număr întreg de unităţi de rastru. Cercurile care nu sunt centrate în origine se translatează înspre origine, iar la convertirea scan se ţine seama de deplasament.

Există o serie de variante pentru conversia scan a cercurilor. Rezolvând ecuaţia implicită se obţine $y = \pm\sqrt{R^2 - x^2}$. Pentru a trasa un sfert de cerc (celelalte trasându-se prin simetrie, vezi figura 3.5.a) se poate incrementa x de la 0 la R cu paşi unitari, rezolvând ecuaţia $y = \sqrt{R^2 - x^2}$ pentru fiecare pas. Viteza de conversie a cercului în acest caz este extrem de scăzută datorită numeroaselor operaţii de multiplicare şi extragere a rădăcinii pătrate. O altă variantă este afişarea punctelor apropriate de $(R\cos\theta, R\sin\theta)$, cu θ parcurgând pas cu pas intervalul $[0^0, 45^0]$ (calculele sunt de asemenea numeroase).

Ca şi în algoritmul de generare a liniilor în rastru, determinarea poziţiilor de coordonate întregi ce formează o imagine circulară pot fi obţinute printr-un algoritm incremental ce determină, la fiecare pas, pixelul cel mai apropiat de cerc.

Algoritmul de generare incrementală se bazează în mod esenţial pe ipoteza de echidistanţă orizontală şi verticală a punctelor rastrului. Dacă această condiţie nu este îndeplinită, algoritmul următor va trasa elipse în loc de cercuri.

Problema este examinată separat pentru fiecare octant. În interiorul unui octant sunt posibile numai două deplasări elementare între două puncte generate consecutiv: o deplasare perpendiculară pe axa mai apropiată, sau o deplasare diagonală. De exemplu, în octantul 1, deplasările elementare sunt în sus sau spre stânga-sus, iar în octantul 2, la stânga sau spre stânga-sus.

Se consideră, de exemplu, cazul primului octant şi a unui cerc de ecuaţie $C(x, y) = 0$ cu $C(x, y) = x^2 + y^2 - R^2$. Se consideră generate la un moment dat (la un pas anterior) coordonatele (x_i, y_i) relative la centrul cercului. Se pot

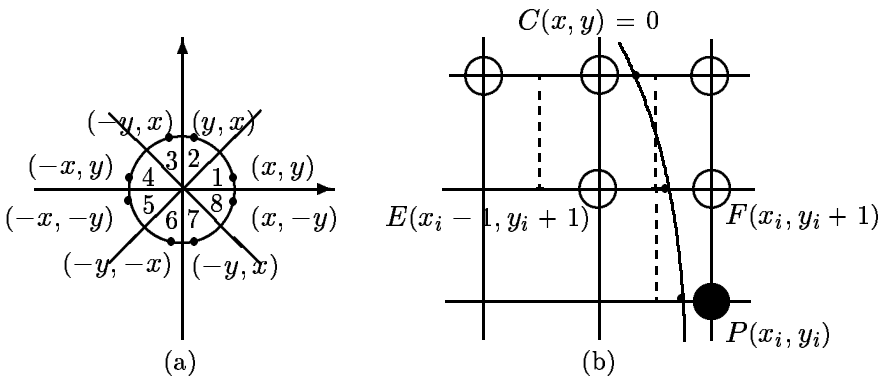


Figura 3.5: (a) Construirea arcelor de cerc prin simetrie (b) Algoritmul incremental în octantul unu

alege două posibilităţi la pasul următor: (x_i, y_i+1) sau (x_i-1, y_i+1) (figura 3.5.b). Va fi generat punctul care este cel mai apropiat de cercul ideal. Primul se află în exteriorul cercului, deci $C(x_i, y_i+1) > 0$, iar celălat în interior, deci $C(x_i-1, y_i+1) < 0$. Punctul corespunzător valorii celei mai mici în modul este cel selectat. Astfel se consideră factorul de decizie

$$\Delta_i = C(x_i, y_i+1) + C(x_i-1, y_i+1) = x_i^2 + (x_i-1)^2 + 2(y_i+1)^2 - 2R^2.$$

Dacă $\Delta_i < 0$, adică $|C(x_i, y_i+1)| < |C(x_i-1, y_i+1)|$, atunci se aprinde primul punct, altfel al doilea. Pentru evitarea multiplelor operaţii de adunare, scădere şi înmulţire, factorul de decizie se poate obţine printr-o formulă recursivă. Astfel, dacă $\Delta_i < 0$, atunci

$$\Delta_{i+1} = \Delta_i + 4y_{i+1} + 2,$$

altfel

$$\Delta_{i+1} = \Delta_i - 4x_{i+1} + 4y_{i+1} + 2.$$

Cele trei puncte (x_i, y_i, Δ_i) definesc starea curentă a algoritmului pentru fiecare punct generat. Pentru aceste numere vor fi alocate variabile de stare. Valorile iniţiale pentru cele trei variabile de stare sunt

$$x_0 = R, \qquad y_0 = 0, \qquad \Delta_0 = 3 - 2R^2.$$

Odată rezolvată problema generării cercului în primul octant, pentru trasarea în ceilalţi octanţi se poate proceda prin simetrie. Există însă situaţii când ordinea generării punctelor cercului are importanţă, de exemplu, dacă trasarea trebuie să se efectueze nu cu linie continuă, ci cu linie întreruptă. Atunci se continuă generarea incrementală şi în ceilalţi octanţi.

În octantul 2, după generarea punctului (x_i, y_i), următorul punct generat trebuie ales între (x_i-1, y_i+1) şi (x_i-1, y_i). Factorul de decizie este

$$\Delta_i = y_i^2 + (y_i+1)^2 + 2(x_i-1)^2 - 2R^2,$$

cu relaţia recursivă

$$\Delta_{i+1} = \begin{cases} \Delta_i - 4x_{i+1} + 2, & \Delta_i < 0 \\ \Delta_i - 4x_{i+1} - 4y_{i+1} + 2, & \text{altfel} \end{cases}.$$

Prin calcule simple se pot stabili formulele de trecere de la octantul 1 la octantul 2 pentru factorul decizional, funcţie de ultimele valori alese şi ultima direcţie de deplasare, respectând regulile de conexiune discretă.

Observaţii:

(a) Pentru simplificarea calculelor factorului decizional, deplasamentul faţă de valoarea anterioară poate fi de asemenea calculat recursiv.

(b) Algoritmul punctului mijlociu pentru trasarea dreptelor poate fi extins la cazul trasării cercurilor. Variabila de decizie se exprimă funcţie de valoarea lui C în punctul situat la mijlocul segmentului format de cele două puncte în discuţie. În cazul primului octant valoarea în discuţie este $C(x_i - \frac{1}{2}, y_i + 1)$. Factorul de decizie este astfel ales încât să se elimine în evaluarea lui C împărţirile, iar valoarea de start a procesului iterativ al factorului de decizie să fie întreagă.

(c) Algoritmul punctului mijlociu a fost extins şi la cazul conversiei scan a elipselor (algoritmul Da Silva). Pentru simplificarea calculelor se trasează arcul de elipsă care se află în primul cuadrant, iar celelalte trei, prin simetrie. Se divide primul cuadrant în două regiuni (figura 3.6.a), în urma determinării punctului în care tangenta la elipsă are panta -1 (punctul satisface ecuaţia elipsei $E(x, y) = b^2x^2 + a^2y^2 - a^2b^2 = 0$ şi componentele vectorului gradient trebuie să fie egale, $2b^2x = 2a^2y$). Ca şi în algoritmul punctului mijlociu pentru drepte, se evaluează funcţia E în punctul de mijloc dintre cei doi pixeli în discuţie în regiunea curentă pentru a determina poziţia punctului de mijloc faţă de elipsă şi, deci, cel mai apropiat pixel. Variabila de decizie este aleasă proporţional cu această valoare, multiplicată cu un factor ce elimină împărţirile prin constante, şi care poate fi evaluată prin diferenţe.

Octant	1	2	3	4	5	6	7	8
Δx	1	0,1	0,-1	-1	-1	0,-1	0,1	1
Δy	0,1	1	1	0,1	0,-1	-1	-1	0,-1

Tabelul 3.1: Mutări în octanţi

Algoritmul descris în observaţia anterioară poate fi extins pentru cazul unei elipse oarecare descrisă prin ecuaţia $G(x, y) := ax^2 + bxy + cy^2 + dx + ey + f = 0$ (se include astfel şi cazul cercului). Planul se divide în opt octanţi, funcţie de direcţiile de trasare incrementală (figura 3.6.b şi tabelul 3.1). Astfel, în octantul 1 se poate alege între mutarea la dreapta sau pe diagonală sus-dreapta. Se consideră aprins pixelul (x_i, y_i) din octantul 1. Variabila de decizie este $d_i = G(x_i + 1, y_i + \frac{1}{2})$. Următoarea valoare poate fi $d_{i+1} = G(x_i + 2, y_i + \frac{1}{2})$

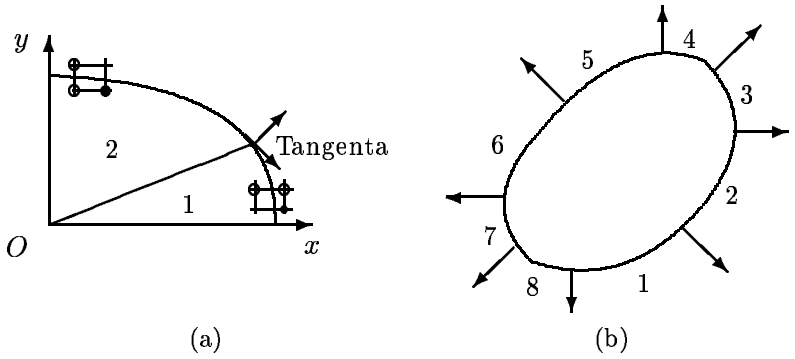


Figura 3.6: (a) Primul cuadrant în trasarea incrementală a elipsei centrate în origine şi cu semiaxele paralele cu axele de coordonate (b) Octanţii în cazul unei elipse oarecare

sau $d_{i+1} = G(x_i + 2, y_i + \frac{3}{2})$. În primul caz, $d_{i+1} = d_i + u_{i+1}$, iar în al doilea, $d_{i+1} = d_i + v_{i+1}$, unde

$$u_i = a(2x_i + 1) + b(y_i + 1/2) + d, \quad v_i = (2a + b)x_i + (b + 2c)y_i + a + b/2 + d + e.$$

Pentru u_i şi v_i se pot stabili relaţii recursive ce depind de direcţia de mişcare. Pentru mişcarea orizontală,

$$u_{i+1} = u_i + 2a, \quad v_{i+1} = v_i + 2a + b,$$

iar pentru mişcarea diagonală,

$$u_{i+1} = u_i + 2a + b, \; v_{i+1} = v_i + 2a + 2b + 2c.$$

Trecerea din octantul 1 în octantul 2 se realizează când expresia următoare îşi schimbă semnul:

$$\left(\frac{\partial G}{\partial x} + \frac{\partial G}{\partial y}\right)(x_i, y_i) = (2ax_i + by_i + d) + (bx_i + 3cy_i + e) = v_i - \left(a + \frac{b}{2}\right)$$

iar trecerea din octantul 2 în octantul 3, când expresia următoare îşi schimbă semnul:

$$\frac{\partial G}{\partial x}(x_i, y_i) = 2ax_i + by_i + d = u_i - \left(a + \frac{b}{2}\right).$$

Algoritmul constă din mai mulţi paşi (un pas pentru un pixel), la un pas efectuându-se următoarele etape:

1. trasare pixel;
2. alegere pixel nou pe baza valorii d_i;
3. calculare u_{i+1} şi v_{i+1} funcţie de alegerea efectuată în etapa anterioară;
4. calculare d_{i+1} prin adăugare u_{i+1} sau v_{i+1} funcţie de alegerea efectuată în etapa a doua;
5. verificarea schimbării octantului.

3.7 Trasarea curbelor de grad doi

Fie o curbă de grad doi în forma generală

$$f(x,y) = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6 = 0,$$

unde coeficienţii $a_1, a_2, \ldots, a_6$ sunt în general numere reale, cu $a_1 > 0$. Curba reprezentată de această ecuaţie este o conică. Se consideră cantităţile

$$\delta = a_2^2 - 4a_1a_3, \qquad \Delta = a_1a_5^2 + a_3a_4^2 - a_2a_4a_5 + (a_2^2 - 4a_1a_3)a_6$$

(dacă determinăm pe x funcţie de y din ecuaţia $f(x,y) = 0$, discriminantul ecuaţiei de grad doi, $\Delta_1(y)$, polinom de grad doi în y, are la rândul său ca discriminant pe $4a_1\Delta$).

Ecuaţia conicei reprezintă o curbă degenerată dacă funcţia de grad doi $f(x,y)$ se poate scrie sub forma

$$f(x,y) = (b_1x + b_2y + b_3)(c_1x + c_2y + c_3),$$

unde coeficienţii b_i, c_i sunt în general numere complexe. Dacă aceştia sunt numere reale, conica degenerează în două drepte. Condiţia de degenerare este

$$\Delta = 0.$$

Dacă curba este nedegenerată, ecuaţia conicei reprezintă o curbă propriu-zisă, reală sau imaginară. Tipul conicei depinde de semnul discriminantului δ:

$$\delta \begin{cases} < 0: & \text{elipsă (sau cerc)}, \\ = 0: & \text{parabolă}, \\ > 0: & \text{hiperbolă}. \end{cases}$$

Dacă curba este reală, atunci funcţia $f(x,y)$ ia valori negative în interiorul concavităţilor curbei $f(x,y) = 0$ şi valori pozitive în exteriorul acestora.

Se presupune că a_i sunt numere întregi fără divizori comuni. Se utilizează ipoteza de echidistanţă orizontală şi verticală a punctelor rastrului. Dacă dimensiunile celulei elementare de rastru sunt în raportul $p/q \neq 1$, cu p, q întregi, pentru o trasare corectă se poate proceda în modul următor: se înlocuieşte ecuaţia curbei $f(x,y) = 0$ cu

$$g(x,y) = p^2 f\left(x, \frac{q}{p}y\right) = 0,$$

care are toţi coeficienţii întregi şi permite redarea corectă a proporţiilor.

Rezolvarea exactă a problemei plasării punctelor în toate cazurile implică, în general, rezolvarea unor ecuaţii de grad doi pentru fiecare punct generat. Aceste operaţii sunt prea complicate pentru un algoritm eficient. De aceea se caută o procedură incrementală de trasare a curbei, care să funcţioneze prin

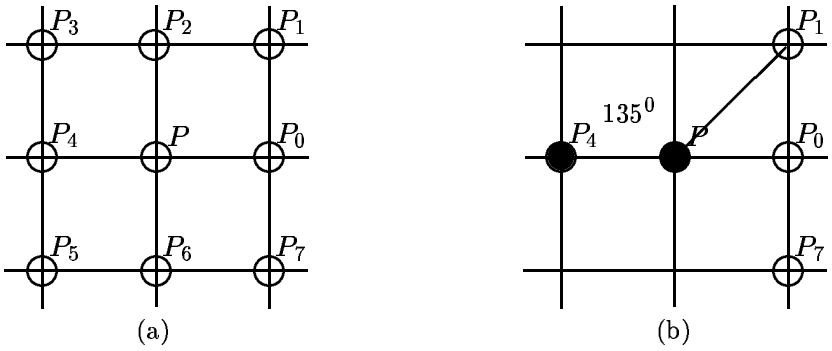


Figura 3.7: (a) Direcţiile posibile de deplasare în algoritmul incremental (b) Aplicarea regulii conexiunii discrete

deplasări elementare pe orizontală, verticală sau în diagonală între punctele succesiv generate (8 direcţii de deplasare).

Se numerotează de la 0 la 7 direcţiile posibile de deplasare de la un punct la cele 8 puncte vecine din rastru, numerotarea făcându-se în sens trigonometric direct. Punctele vecine lui P se notează cu P_i, $i = 0, \ldots, 7$ (figura 3.7.a).

Observaţii:

(a) pentru două puncte (x_1, y_1) şi (x_2, y_2) relativ apropiate de curba ideală, se poate decide care dintre ele este cel mai apropiat prin compararea cantităţilor $|f(x_1, y_1)|$ şi $|f(x_2, y_2)|$. Punctul cel mai apropiat de curbă este acela în care funcţia $f(x, y)$ este minimă în valoare absolută.

(b) dacă algoritmul de generare a efectuat o deplasare în direcţia i între două puncte succesive, atunci pentru următorul punct generat trebuie să se efectueze o deplasare elementară în una din direcţiile $i-1$, i sau $i+1$ (considerate modulo 8). Această regulă este o altă exprimare pentru regula de conexiune discretă (figura 3.7.b). Cercurile minime care îndeplinesc această condiţie au diametrul de cel puţin 3 unităţi de rastru.

Ideea algoritmului este următoarea. Se alege sensul de parcurs al curbei astfel încât $f(x, y) < 0$ (> 0) la stânga (respectiv dreapta) direcţiei de înaintare. Dacă punctul P a fost generat în urma deplasării elementare în direcţia i, atunci punctele candidat sunt P_{i-1}, P_i, P_{i+1} (considerate modulo 8). Conform presupunerii anterioare P_{i+1} se află în interior $(f(P_{i+1}) < 0)$, iar P_{i-1} în exterior $(f(P_{i-1}) > 0)$. Despre punctul P_i nu se poate afirma dacă este în exterior sau interior. Pentru alegerea unuia dintre cele trei puncte sunt necesare următoarele comparaţii:

$$\begin{cases} f(P_{i-1}) + f(P_i) < 0: & \text{se alege punctul } P_{i-1} \\ f(P_{i-1}) + f(P_i) \geq 0 \text{ şi } f(P_i) + f(P_{i+1}) \leq 0: & \text{se alege punctul } P_i \\ f(P_i) + f(P_{i+1}) > 0: & \text{se alege punctul } P_{i+1} \end{cases}$$

Observaţie. Aplicând acest criteriu decizional pentru ecuaţia unei drepte, se obţin relaţiile ce stau la baza algoritmului Bresenham. De asemenea, acest criteriu general este particularizat de relaţiile care guvernează trasarea incrementală a cercurilor.

Dacă se exprimă toate valorile funcţiei f în punctele din imediata apropiere a lui P, de coordonate (x, y), rezultă:

$$f(P_0) = f(x+1, y) = f(P) + (2a_1x + a_2y + a_4) + a_1;$$
$$f(P_1) = f(P) + (2a_1x + a_2y + a_4) + (a_2x + 2a_3y + a_5) + a_1 + a_2 + a_3;$$
$$\ldots$$

şi se poate dezvolta algoritmul de trasare incrementală pe baza criteriului de decizie şi a acestor relaţii de recurenţă. Sunt efectuate doar operaţii de adunare, scădere şi înmulţire între numere întregi pentru fiecare punct generat. Operaţiile de înmulţire pot fi evitate în cazul unei alegeri adecvate a variabilelor de stare. Relaţiile de recurenţă pot fi exprimate simplu în funcţie de derivatele parţiale f'_x şi f'_y ale lui f în punctul P, notate cu fx şi fy:

$$fx(x, y) = f'_x(x, y) = (2a_1x+a_2y+a_4),\ fy(x, y) = f'_y(x, y) = (a_2x+2a_3y+a_5).$$

Funcţiile fx, fy fiind liniare, actualizările lor incrementale se exprimă numai prin operaţii de adunare şi scădere. În cazul unei deplasări elementare între două puncte vecine:

$$fx(P_0) = fx(P) + 2a_1, \qquad fy(P_0) = fy(P) + a_2,$$
$$fx(P_1) = fx(P) + 2a_1 + a_2, \qquad fy(P_1) = fy(P) + a_2 + 2a_3,$$
$$\ldots$$

Variabilele de stare sunt valorile funcţiilor f, fx, fy în punctul curent generat.

3.8 Trasarea curbelor plane

În raport cu un reper cartezian ortogonal, curbele plane pot fi descrise prin:

1. ecuaţii explicite: un punct oarecare (x, y) de pe curbă este descris în forma explicită $y = f(x)$;
2. ecuaţii polare: un punct oarecare (x, y) de pe curbă este descris în coordonate polare prin cuplul (r, t), unde t parcurge un interval $[a, b]$ şi $r = f(t)$, corespondenţa dintre cele două sisteme de coodonate, carteziene şi polare, presupunând ecuaţii de tipul $x = r\cos t$, $y = r\sin t$.
3. ecuaţii parametrice: un punct oarecare (x, y) de pe curbă este descris prin $x = f(t)$, $y = g(t)$, unde t parcurge un interval $[a, b]$ din I R, adică $F(t) = (f(t), g(t))$ reprezintă un drum în planul I R×I R;
4. ecuaţii implicite: un punct oarecare (x, y) de pe curbă este descris de ecuaţia $F(x, y) = 0$. Folosind teorema funcţiilor implicite din calculul diferenţial, se ajunge la studiul curbelor descrise prin ecuaţii explicite. În secţiunea anterioară a fost descris un algoritm incremental pentru cazul în care $F(x, y)$ este funcţie polinomială de grad maxim doi în x şi y.

Pentru trasarea curbelor definite explicit trebuie să se ţină seama de dispozitivul de afişare grafică de care se dispune. Este necesară scrierea unui proceduri care operează cu coordonate (în mod grafic) ce reprezintă numere întregi şi care ţine seama de scara la care se execută desenul. Scalarea care trebuie efectuată se obţine în urma analizei valorilor extreme ale funcţiei pe domeniul pe care se reprezintă.

Pentru reprezentarea grafică a unei curbe descrisă prin ecuaţii polare este necesară o transformare în coordonate carteziene. Această transformare este $x = f(t)\cos t$, $y = f(t)\sin t$ (curbă plană descrisă prin ecuaţii parametrice).

Se abordează problema trasării curbei cu reprezentarea parametrică

$$x = x(t), \quad y = y(t), \qquad t \in [0, 1],$$

unde $x(\cdot)$ şi $y(\cdot)$ sunt funcţii continue.

O primă idee pentru dezvoltarea unui algoritm de trasare constă în alegerea unei rate constante Δ de creştere a variabilei independente t şi selectarea, în mediul de afişare, a punctelor care corespund cel mai bine adreselor calculate $(x(k\Delta), y(k\Delta))$, unde $k = 0, 1, \ldots, [1/\Delta]$. Principala dificultate constă în alegerea parametrului Δ, deoarece nu se ştie de la început câte puncte vor fi efectiv generate. Pe de altă parte, însăşi reprezentarea parametrică dată, care nu este unică, poate fi defavorabilă, astfel încât orice alegere a unui constante de creştere Δ este total nepotrivită: pe unele porţiuni ale curbei, parametrul Δ poate fi prea mic, ceea ce are ca rezultat selectarea de mai multe ori a aceluiaşi punct; pe alte porţiuni ale curbei, acelaşi parametru Δ poate fi prea mare, generând puncte consecutive neadiacente în mediu de afişare, care, dacă sunt unite prin linii drepte, nu mai aproximează suficient de exact curba dată. Astfel, orice metodă bazată pe creşterea constantă a variabilei independente nu poate produce algoritmi eficienţi în toate cazurile. Este necesar să se găsească o metodă bazată pe variaţia neconstantă a variabilei t, care să urmărească pe cât posibil variaţia curbei.

Principiul metodei trasării prin înjumătăţirea intervalului este următorul: se calculează mai întâi adresele punctelor de la capetele curbei, corespunzând extremităţilor intervalului de variaţie a parametrului t. Dacă aceste două puncte sunt suficient de apropiate în raport cu rezoluţia mediului de afişare, problema este rezolvată prin aprinderea acestor puncte. În caz contrar, intervalul curent de variaţie a lui t se împarte în două subintervale de lungimi egale, subintervalul din dreapta se memorează într-o structură de date de tip stivă pentru necesităţi ulterioare şi se reia problema cu subintervalul din stânga. Înjumătăţirea intervalului continuă până când lungimea acestuia este suficient de mică pentru ca punctele de pe curbă corespunzătoare extremităţilor intervalului să fie adiacente în mediu de afişare. În acest caz se descarcă stiva şi se reia procedura cu cel mai recent subinterval memorat în stivă. Trasarea curbei se termină când stiva se goleşte complet.

Metoda de trasare prin înjumătăţirea intervalului poate fi concretizată şi într-o formă mai concisă cu ajutorul unei proceduri recursive, care să nu facă apel în mod explicit la o structură de date de tip stivă.

Corectitudinea procedurii de principiu se bazează implicit pe faptul că funcţiile de parametrizare sunt continue. În caz contrar, nu este neapărat necesar ca, între două puncte corespunzătoare valorilor $t_1 < t_2$ ale parametrului, toate punctele intermediare (în sensul valorilor absciselor şi ordonatelor) să corespundă la valori ale lui t din intervalul $[t_1, t_2]$.

Există situaţii când ordinea generării punctelor curbei are importanţă. De exemplu, dacă trasarea curbei trebuie să se efectueze nu cu linie continuă, ci cu linie întreruptă corespunzând unui anumit şablon (configuraţie liniară de biţi, pattern) predefinit, atunci algoritmul de trasare trebuie să permită calcularea punctelor succesive ale curbei în paralel cu parcurgerea configuraţiei binare a şablonului, care se repetă în mod circular. Punctul curent de pe curbă este efectiv selectat în mediu de afişare numai dacă bitul corespunzător din şablon este 1. Trasarea cu şablon nu este pe deplin reuşită dacă algoritmul de trasare generează de două ori acelaşi punct pe curbă. Este necesară inhibarea generării efective a unui punct dacă coordonatele acestuia coincid cu cele ale unui punct generat anterior.

3.9 Trasarea curbelor spaţiale

3.9.1 Descrierea curbelor spaţiale

Curbele spaţiale pot fi descrise:

1. explicit prin ecuaţii de tipul $y = f(x),\ z = g(x)$;
2. implicit printr-o ecuaţie de tipul $F(x, y, z) = 0$;
3. parametric prin ecuaţii de tipul $x = x(t),\ y = y(t),\ z = z(t)$.

Prima descriere nu este valabilă pentru toate curbele. De exemplu, un cerc într-un plan cu z constant are două valori pentru un x. A doua descriere oferă mai multe soluţii, dar, de exemplu, semicercul $x^2 + y^2 = 1,\ x \geq 0$ nu poate fi descris printr-o ecuaţie de tipul cerut. De aceea, reprezentarea parametrică este des utilizată.

Curbele parametrice polinomiale definesc un punct de pe o curbă spaţială utilizând trei polinoame de un parametru t. Curbele cubice folosesc polinoame cubice în t. Curbele polinomiale de grad mai mare ca patru sunt rar utilizate deoarece prezintă o multitudine de ondulaţii.

3.9.2 Curbe parametrice cubice

Cubica parametrică spaţială poate fi exprimată parametric prin trei funcţii cubice şi este definită prin 12 coeficienţi:

$$\begin{cases} x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) = a_y t^3 + b_y t^2 + c_y t + d_y, \\ z(t) = a_z t^3 + b_z t^2 + c_z t + d_z, \end{cases}$$

unde $t \in [0,1]$. Dacă se notează

$$Q(t) = (x(t), y(t), z(t)), \quad T = (t^3, t^2, t, 1), \quad C = \begin{pmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{pmatrix}$$

atunci $Q(t) = TC$. Vectorul tangent la curbă în punctul corespunzător unui anumit t este

$$Q'(t) = (3t^2, 2t, 1, 0)C$$

Fiecare polinom cubic are 4 coeficienţi şi deci sunt necesare 12 condiţii pentru determinarea necunoscutelor.

Un segment de curbă este definit prin constrângeri privind:

(a) punctele de capăt;
(b) vectorii tangenţi;
(c) continuitatea între mai multe segmente de curbă.

În cazul general, matricea C este rescrisă sub forma $C = MG$, unde M este o matrice 4×4, numită matrice de bază, iar G o matrice 4×3, numită vectorul geometric (care defineşte constrângerile geometrice). Se notează G_x prima coloană a lui G. Atunci $x(t) = TMG_x$, adică

$$x(t) = b_1(t)g_{1x} + b_2(t)g_{2x} + b_3(t)g_{3x} + b_4(t)g_{4x},$$

unde polinoamele blending (amestec, combinare) b_i sunt polinoamele cubice obţinute din ecuaţia matriceală $B = TM$ (de exemplu, $b_1(t) = t^3m_{11} + t^2m_{21} + tm_{31} + m_{41}$).

Curbe Hermite

Curbele Hermite sunt descrise prin funcţiile parametrice cubice ce au gradul minim necesar pentru a îndeplini patru condiţii: (figura 3.8.a):

(a) să treacă prin două puncte date (P_1, P_4);
(b) să aibă tangente date în aceste puncte (R_1, R_4).

Se defineşte vectorul de geometrie

$$G_H = \begin{pmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{pmatrix}$$

unde cele trei elemente ale unei linii reprezintă coordonatele punctelor, respectiv lungimile tangentelor descompuse pe axele de coordonate. Fie G_{Hx} prima coloană a acestei matrici şi M_H matricea de bază. Atunci $x(t) = TM_HG_{Hx}$. Pentru determinarea matricei de bază M_H se utilizează condiţiile din ipoteză. Se ştie că

$$\begin{aligned} x(0) &= x_{P_1} = (0,0,0,1)M_HG_{Hx}, \\ x(1) &= x_{P_4} = (1,1,1,1)M_HG_{Hx}, \\ x'(0) &= x_{R_1} = (0,0,1,0)M_HG_{Hx}, \\ x'(1) &= x_{R_4} = (3,2,1,0)M_HG_{Hx} \end{aligned}$$

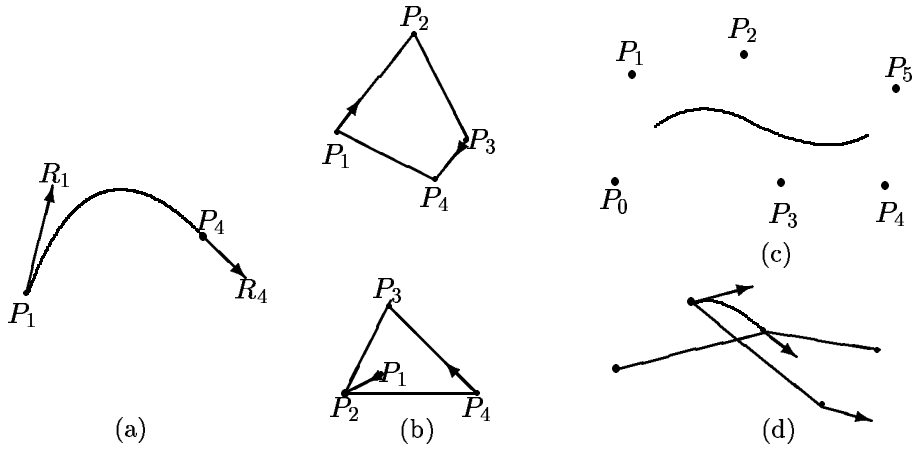


Figura 3.8: Definirea curbelor cubice (a) Curbă Hermite (b) Curbe Bézier (c) Curba B-spline uniformă (d) Curba spline Catmull-Rom

adică

$$G_{Hx} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} M_H G_{H_x}.$$

Astfel

$$M_H = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Aceeaşi matrice se obţine utilizând condiţiile în coordonata y sau z. Generic,

$$Q(t) = TM_H G_H.$$

Polinoamele Hermite B_H se obţin din $B_H = TM_H$. Funcţie de acestea, curba Hermite este descrisă astfel:

$$Q(t) = B_H G_H = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4.$$

Pentru continuitatea geometrică a două segmente de curbă Hermite este necesar ca vectorii de geometrie să fie de tipul $(P_1, P_4, R_1, R_4)^T$ şi $(P_4, P_7, kR_4, R_7)^T$, unde $k \geq 0$. Pentru continuitatea de clasă C^1 a segmentului rezultat este necesar ca $k = 1$.

Observaţie. Curbele Hermite pot fi uşor transformate modificând doar vectorul de geometrie (matricea de bază este invariantă la rotaţii, scalări sau translaţii).

Curbe Bézier

Se obţin cu ajutorul a patru puncte de control. Primul şi ultimul servesc la precizarea capetelor intervalului (în figura 3.8.b, P_1 şi P_4), iar punctele suplimentare determină, împreună cu capetele, direcţia tangentelor (P_1P_2, P_3P_4). Dacă R_1 şi R_4 sunt tangentele în punctele P_1, respectiv P_4, atunci

$$R_1 = Q'(0) = m(P_2 - P_1), \quad R_4 = Q'(1) = m(P_4 - P_3).$$

Parametrul m se numeşte factor de formă. Din considerentul includerii curbei în închiderea convexă a punctelor P_1, P_2, P_3, P_4 se consideră $m = 3$. Vectorul de geometrie Bézier este

$$G_B = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix}.$$

Atunci

$$G_H = \begin{pmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -m & m & 0 & 0 \\ 0 & 0 & -m & m \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} = M_{HB}G_B.$$

Cum $Q(t) = TM_HG_H = TM_BG_B$ se obţine matricea de bază Bézier pentru $m = 3$

$$M_B = M_HM_{HB} = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Polinoamele $B_B = TM_B$ sunt polinoamele Bernstein. Funcţie de acestea curba Bézier este descrisă prin

$$Q(t) = (1-t)^3P_1 + 3t(t-1)^2P_2 + 3t^2(1-t)P_3 + t^3P_4.$$

Pentru continuitatea geometrică a două segmente de curbă Bézier este necesar ca vectorii de geometrie să fie de tipul $(P_1, P_2, P_3, P_4)^T$ şi (P_4, P_5, P_6, P_7), cu $P_3 - P_4 = k(P_4 - P_5)$ unde $k \geq 0$. Pentru continuitatea de clasă C^1 a segmentului rezultat este necesar ca $k = 1$.

Observaţie. Deoarece suma ponderilor polinoamelor Bernstein este unu, segmentul de curbă se află în interiorul închiderii convexe a celor patru puncte de control.

Curbe spline

Sunt constituite din mai multe segmente ce prezintă continuitate de clasă C^1 sau C^2 în punctele de cuplare. Există mai multe tipuri de curbe de acest tip,

cele mai utilizate fiind curbele B-spline uniforme sau neuniforme şi curbele β-spline.

În cazul concatenării unor segmente de curbe de tip Hermite sau Bézier, schimbarea unui punct de control afectează întreaga curbă.

Curbele B-spline consistă din mai multe segmente de curbă a căror coeficienţi polinomiali depind de un număr mic de puncte de control (control local), astfel încât modificarea unui punct afectează numai o mică parte a curbei.

O curbă cubică B-spline presupune precizarea a $m+1$ puncte de control notate P_0, $P_1, \ldots,$ P_m, unde $m \geq 3$, şi constă din $m-2$ segmente de curbă cubică, notate Q_3, $Q_4, \ldots,$ Q_m. Fiecare segment de curbă este definit doar pe propriul interval $0 \leq t < 1$. Parametrul t este ajustat astfel încât Q_i să fie definit de $t_i \leq t \leq t_{i+1}$, pentru $3 \leq i \leq m$. Dacă $m = 3$, există un singur segment de curbă, Q_3, definit pe intervalul $t_3 \leq t < t_4$ prin punctele de control $P_0, \ldots, P_3$. Pentru $m \geq i \geq 4$, există un punct comun (nod) între segmentele Q_{i-1} şi Q_i.

O curbă B-spline uniformă presupune ca nodurile să fie spaţiate la intervale egale ale parametrului t. Se presupune astfel că $t_3 = 0$, $t_{i+1} - t_i = 1$, $i \geq 3$. Termenul "B" provine de la "bază", deoarece acest tip de curbe pot fi reprezentate ca sume ponderate ale unor funcţii polinomiale de bază (funcţii blending).

Segmentul de curbă Q_i este definit de punctele P_{i-3}, P_{i-2}, P_{i-1}, P_i astfel încât vectorul de geometrie B-spline pentru segmentul Q_i este $G_{Bs_i} = (P_{i-3}, P_{i-2}, P_{i-1}, P_i)$, $3 \leq i \leq m$. Fiecare punct de control influenţează patru segmente de curbă. Dacă $T_i = ((t-t_i)^3, (t-t_i)^2, (t-t_i), 1)$, atunci $Q_i(t) = T_i M_{Bs} G_{Bs\,i}$, $t_i \leq t < t_{i+1}$, $3 \leq i \leq m$. Pentru determinarea matricei de bază se exprimă segmentele de curbă funcţie de polinoamele blending:

$$Q_i(t-t_i) = b_1(t)P_{i-3} + b_2(t)P_{i-2} + b_3(t)P_{i-1} + b_4(t)P_i, \quad 0 \leq t < 1.$$

Dacă se impune condiţia de continuitate de clasă C^2 a segmentelor de dreaptă (Q_i şi Q_{i+1} să se cupleze în t_{i+1} prin continuitate de clasă C^2, pentru oricare puncte de control), şi condiţia ca ponderile polinomiale să aibă suma 1 (curba se află în închiderea convexă a mulţimii punctelor de control, vezi figura 3.8.c),

$$\begin{cases} Q_i(t_{i+1}-t_i) = Q_{i+1}(t_{i+1}-t_{i+1}), & i = 3, \ldots, m \\ Q_i'(t_{i+1}-t_i) = Q_{i+1}'(t_{i+1}-t_{i+1}), & i = 3, \ldots, m \\ Q"_i(t_{i+1}-t_i) = Q"_{i+1}(t_{i+1}-t_{i+1}), & i = 3, \ldots, m \\ b_1(t) + b_2(t) + b_3(t) + b_4(t) = 1, & \forall t \in [0,1], \end{cases}$$

se obţin polinoamele blending astfel încât

$$Q_i(t-t_i) = \frac{(1-t)^3}{6}P_{i-3} + \frac{3t^3-6t^2+4}{6}P_{i-2} + \frac{-3t^3+3t^2+3t+1}{6}P_{i-1} + \frac{t^3}{6}P_i,$$

iar matricea de bază B-spline este

$$M_{Bs} = \frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}.$$

Un alt exemplu de curbă spline este cea din figura 3.8 (d), numită curba Catmull-Rom. Considerând patru puncte succesive din set, P_0, P_1, P_2, P_3, se determină $x(t), y(t), z(t)$ care interpolează P_1 şi P_2 cu derivatele aproximative la capete $(P_2 - P_0)/2$ şi $(P_3 - P_1)/2$. Se obţine continuitatea C^1 a segmentelor şi o altă matrice de bază decât M_{Bs}. Curba nu posedă proprietatea închiderii convexe.

3.9.3 Trasarea curbelor parametrice cubice

Există două metode principale de trasare a curbelor parametrice:

- evaluarea polinoamelor $x(t)$, $y(t)$, $z(t)$ pentru o diviziune a intervalului [0,1] parcurs de t, în variantele:
  (a) evaluare directă (eventual utilizând schema Horner pentru determinarea valorilor polinoamelor);
  (b) evaluare iterativă;
- subdivizarea recursivă.

Metoda evaluării directe presupune următoarele etape:

1. se citesc coordonatele punctelor de control;
2. pentru fiecare pereche de puncte de control succesive:
   - se parcurge intervalul [0,1] cu un pas constant $\delta = 1/n$, $n > 1$;
   - pentru fiecare $t_i = i\delta$, $i = 1, \ldots, n$ se calculează
     (a) coordonatele punctului de pe curbă $x(t_i)$, $y(t_i)$, $z(t_i)$;
     (b) coordonatele proiecţiei acestui punct intermediar;
3. se trasează pe ecran o linie poligonală care trece prin proiecţiile punctelor intermediare.

Se observă că pentru evaluarea polinoamelor cubice sunt necesare 11 multiplicări şi 10 adunări pentru un punct. Utilizând schema Horner pentru evaluarea polinoamelor sunt necesare doar 9 multiplicări şi 10 adunări.

Multiplicările pot fi evitate prin evaluarea iterativă. Fie $f(t) := at^3 + bt^2 + ct + d$, $t_n := n\delta$, $f_n := f(t_n)$, $\Delta f(t) := f(t+\delta) - f(t)$, $\Delta^2 f(t) := \Delta(\Delta f(t))$ (diferenţe finite). Se observă că

$$\begin{gathered} \Delta f(t) = f(t+\delta) - f(t) = 3at^2\delta + t(3a\delta^2 + 2b\delta) + a\delta^3 + b\delta^2 + c\delta, \\ \Delta^2 f(t) = \Delta f(t+\delta) - \Delta f(t) = 6a\delta^2 t + 6a\delta^3 + 2b\delta^2, \\ \Delta^3 f(t) = \Delta^2 f(t+\delta) - \Delta^2 f(t) = 6a\delta^3 \end{gathered}$$

şi

$$f_{n+1} = f_n + \Delta f_n, \quad \Delta f_n = \Delta f_{n-1} + \Delta^2 f_{n-1}, \quad \Delta^2 f_{n-1} = \Delta^2 f_{n-2} + \Delta^3 f_{n-2}.$$

Astfel, pornind de la valorile de start

$$f_0 = d, \quad \Delta f_0 = a\delta^3 + b\delta^2 + c\delta, \quad \Delta^2 f_0 = 6a\delta^3 + 2b\delta^2, \quad \Delta^3 f_0 = 6a\delta^3,$$

se pot calcula recursiv valorile f_n. În această variantă, algoritmul pentru trasarea unei curbe parametrice cubice presupune efectuarea unor multiplicări doar în faza de iniţializare a valorilor recursive. Pasul n al algoritmului presupune următoarele calcule:

$$\begin{cases} x_{n+1} = x_n + \Delta x_n, & \Delta x_{n+1} = \Delta x_n + \Delta^2 x_n, & \Delta^2 x_{n+1} = \Delta^2 x_n + \Delta^3 x_n; \\ y_{n+1} = y_n + \Delta y_n, & \Delta y_{n+1} = \Delta y_n + \Delta^2 y_n, & \Delta^2 y_{n+1} = \Delta^2 y_n + \Delta^3 y_n; \\ z_{n+1} = z_n + \Delta z_n, & \Delta z_{n+1} = \Delta z_n + \Delta^2 z_n, & \Delta^2 z_{n+1} = \Delta^2 z_n + \Delta^3 z_n; \end{cases}$$

precum şi trasarea dreptei ce uneşte proiecţia punctului (x_n, y_n, z_n) cu proiecţia punctului $(x_{n+1}, y_{n+1}, z_{n+1})$. Algoritmul presupune 9 adunări per pas şi nici o multiplicare. Dezavantajul utilizării acestui algoritm constă în faptul că pentru valori mari ale numărului de paşi n rezultatele pot fi afectate de cumularea erorilor de trunchiere.

Trasarea unei curbe 3D cu pas constant nu conduce întotdeauna la imaginile dorite (vezi trasarea curbelor plane). Există două modalităţi de eliminare a acestui efect nedorit:

1. creşterea gradului funcţiilor parametrice (se produc o serie de inflexiuni adiţionale);
2. creşterea numărului punctelor de control cu divizarea segmentelor de curbă în mai multe segmente.

Se consideră exemplul unui segment de curbă Bézier determinat prin patru puncte de control. Acesta poate fi subdivizat în două segmente având în total şapte puncte de control (cele două segmente noi au un punct comun şi se suprapun pe segmentul original). Dată curba $Q(t)$ definită de punctele P_1, P_2, P_3, P_4, se caută punctele L_1, L_2, L_3, L_4 care definesc un segment de curbă ce coincide cu $Q(t)$ pentru $0 \leq t \leq 1/2$ (analog R_1, R_2, R_3, R_4 pentru $1/2 \leq t \leq 1$). Punctul de pe curba Bézier care corespunde unui parametru t se poate obţine astfel (figura 3.9.a): se determină punctele $L_2 \in P_1P_2$, $H \in P_2P_3$, $R_3 \in P_3P_4$ care împart segmentele P_1P_2, P_2P_3 şi P_3P_4 în raportul $t/(1-t)$; analog se determină $L_3 \in L_2H$ şi $R_2 \in HR_3$ care împart segmentele menţionate în acelaşi raport şi $L_4 \in L_3R_2$ tot în acelaşi raport. Punctul L_4 este punctul căutat. În cazul $t = 1/2$ noile puncte de control pot fi uşor calculate: $L_2 = (P_1 + P_2)/2$, $H = (P_2 + P_3)/2$, $L_3 = (L_2 + H)/2$, $R_3 = (P_3 + P_4)/2$, $R_2 = (H + R_3)/2$, $L_4 = R_1 = (L_3 + R_2)/2$.

Curbele B-spline neuniforme permit de asemenea subdivizarea segmentelor componente.

Algoritmul bazat pe subdivizarea recursivă este indicat, în special, în cazul curbelor Bézier, deoarece subdivizarea este uşor de realizat şi testul de oprire a recursivităţii este simplu: dacă P_1 şi P_4 sunt punctele prin care trece segmentul de curbă, iar P_2, P_3 punctele de control pentru determinarea derivatei, iar distanţa de la P_2 la P_1P_4 şi distanţa de la P_3 la P_1P_4 sunt sub un anumit nivel ε (figura 3.9b), atunci segmentul de curbă P_1P_4 este suficient de plat

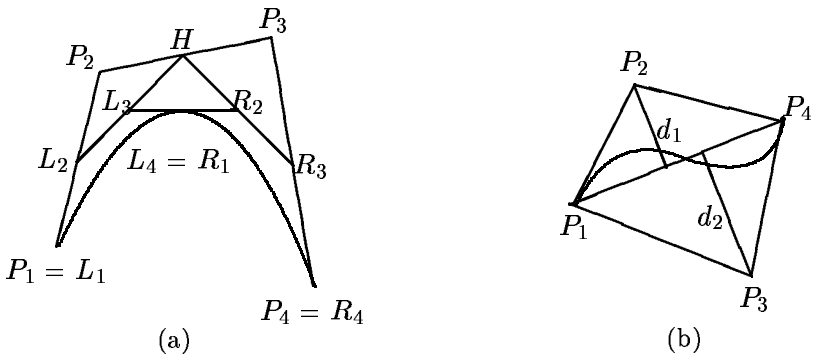


Figura 3.9: Subdivizarea recursivă (a) Subdivizarea unui segment de curbă Bézier (b) Testul de liniaritate a segmentului de curbă Bézier

(din proprietatea de incluziune în închiderea convexă a mulţimii punctelor de control) pentru a fi aproximat cu o dreaptă (altfel se continuă subdivizarea).

3.10 Trasarea suprafeţelor curbe

3.10.1 Descrierea suprafeţelor curbe

Suprafeţele curbe pot fi descrise:

1. explicit, prin ecuaţii de tipul $z = f(x, y)$;
2. implicit, printr-o ecuaţie de tipul $F(x, y, z) = 0$;
3. parametric, prin ecuaţii de tipul $x = x(s, t)$, $y = y(s, t)$, $z = z(s, t)$.

În grafica pe calculator sunt mult utilizate suprafeţele determinate polinomial prin funcţia de două variabile x şi y:

$$z = \sum_{j=0}^{n} \sum_{i=0}^{m} a_{ij} x^i y^i.$$

Suprafaţa se notează $P_{m,n}$. Matricea A poartă numele de amprentă a suprafeţei. Se disting următoarele cazuri speciale:

- $P_{0,0}$, $P_{0,1}$, $P_{1,0}$ sunt plane;
- $P_{1,1}$ este paraboloidul hiperbolic;
- $P_{0,n}$, $P_{m,0}$ pentru $m, n \geq 2$ sunt suprafeţe cilindrice cu generatoarele paralele cu axa Ox respectiv Oy, iar curbele directoare ale suprafeţelor cilindrice sunt conţinute în planele yOz, respectiv xOz;
- $P_{1,n}$, $P_{m,1}$ pentru $m, n \geq 1$ sunt suprafeţele riglate, ale căror generatoare sunt conţinute în planele $y = const$, respectiv $x = const$.

- $P_{2,2}$ sunt suprafeţele cuadrice (hiperboloidul, paraboloidul, elipsoidul, cilindrul, conul);
- $P_{3,3}$ sunt suprafeţele bicubice.

Un exemplu concludent de suprafeţe determinate prin polinoame sunt suprafeţele de interpolare, definite prin puncte şi prin curbe marginale. Suprafaţa de interpolare definită de patru puncte de colţ este un patrulater strâmb, numit cuadrică riglată.

Suprafeţele cuadrice sunt definite implicit printr-o ecuaţie $F(x, y, z) = 0$ cu F polinom cuadric în x, y şi z:

$$F(x, y, z) = a_{11}x^2 + a_{22}y^2 + a_{33}z^2 +$$

$$+2a_{21}xy + 2a_{23}yz + 2a_{13}xz + 2a_{14}x + 2a_{24}y + 2a_{34}z + a_{44}.$$

Această reprezentare este familiară pentru sferă, elipsoid sau cilindru. De exemplu, dacă $a_{11} = a_{22} = a_{33} = -a_{44} = 1$ şi ceilalţi coeficienţi sunt nuli, se defineşte sfera unitate centrată în origine. Reprezentarea matriceală este

$$P^T AP = 0$$

unde A este matricea simetrică a coeficienţilor, iar $P = (x, y, z, 1)^T$.

Suprafeţele parametrice polinomiale bivariate definesc coordonatele unui punct de pe o suprafaţă utilizând trei polinoame bivariate, câte unul pentru fiecare dimensiune.

O problemă importantă este cuplarea suprafeţelor. Aceasta poate fi:

(a) de tip alipire, când se cere identitatea anumitor curbe marginale;

(b) cuplare netedă, ce impune suplimentar identitatea tangentelor (derivate parţiale de ordinul întâi) şi a curburilor (derivate mixte) de-a lungul curbelor marginale.

3.10.2 Suprafeţe parametrice bicubice

Folosind două familii de curbe cubice 3D se poate defini o suprafaţă curbă în spaţiu. O familie de curbe cubice 3D se poate obţine introducând, în ecuaţiile parametrice ale unei curbe 3D, a unui parametru nou ce variază de asemenea în intervalul $[0, 1]$. Fie curba $Q(s) = SMG$. Dacă G variază funcţie de un parametru t se obţine o familie de curbe cubice:

$$Q(s, t) = SMG(t) = SM \begin{pmatrix} G_1(t) \\ G_2(t) \\ G_3(t) \\ G_4(t) \end{pmatrix}.$$

Dacă $G_i(t)$ sunt polinoame cubice de acelaţi tip ca şi cele în s, adică $G_i(t) = TMG_i$, unde G_i este vectorul de geometrie caracteristic, $G_i =$

$(g_{i1}, g_{i2}, g_{i3}, g_{i4})^T$, atunci

$$Q(s,t) = SM \begin{pmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{pmatrix} M^T T^T = SMGM^T T^T, \quad 0 \le s,t \le 1$$

unde M este matricea de bază, iar G este matricea geometrică. Notând G_x, G_y, G_z matricile 4×4 ce conţin coordonatele carteziene ale elementelor de geometrie g_{ij}, se obţin expresiile parametrice

$$\begin{cases} x(s,t) = SMG_x M^T T^T, \\ y(s,t) = SMG_y M^T T^T, \\ z(s,t) = SMG_z M^T T^T. \end{cases}$$

Suprafeţe Hermite

O suprafaţă bicubică Hermite se defineşte pe baza a patru puncte în spaţiu corespunzătoare valorilor extreme 0 şi 1 pentru s şi t, precum şi prin câte trei tangente la suprafaţă în fiecare dintre aceste puncte. Fiecare vector tangent are componentele determinate de derivatele de ordin 1 şi 2 ale funcţiilor $x(t,s)$, $y(s,t)$, $z(t,s)$.

Matricea de geometrie poate fi construită pornind de la familia de curbe Hermite:

$$x(s,t) = SM_H G_{Hx}(t) = SM_H \begin{pmatrix} P_1(t) \\ P_4(t) \\ R_1(t) \\ R_4(t) \end{pmatrix}_x$$

În figura 3.10 (a) sunt trasate cubicele în s definite la $t = 0,\ 0.2,\ 0.4,\ 0.6,\ 0.8, 1$.

Fiecare element al vectorului de geometrie poate fi reprezentat drept o curbă Hermite:

$$(P_1(t), P_4(t), R_1(t), R_4(t)) = TM_H \begin{pmatrix} g_{11} & g_{21} & g_{31} & g_{41} \\ g_{12} & g_{22} & g_{32} & g_{42} \\ g_{13} & g_{23} & g_{33} & g_{43} \\ g_{14} & g_{24} & g_{34} & g_{44} \end{pmatrix},$$

unde g_{ij} rezultă din forma Hermite pentru curbe. De exemplu, $g_{33_x} = (\partial^2 x)/(\partial s \partial t)(0,0)$, deoarece este vectorul tangent la curba $R_{1x}(t)$ în $t = 0$, care este la rândul său vectorul tangent la curba $x(s,0)$ în $s = 0$. Astfel

$$G_{Hx} = \begin{pmatrix} x(0,0) & x(0,1) & \frac{\partial}{\partial t}x(0,0) & \frac{\partial}{\partial t}x(0,1) \\ x(1,0) & x(1,1) & \frac{\partial}{\partial t}x(1,0) & \frac{\partial}{\partial t}x(1,1) \\ \frac{\partial}{\partial s}x(0,0) & \frac{\partial}{\partial s}x(0,1) & \frac{\partial^2}{\partial s \partial t}x(0,0) & \frac{\partial^2}{\partial s \partial t}x(0,1) \\ \frac{\partial}{\partial s}x(1,0) & \frac{\partial}{\partial s}x(1,1) & \frac{\partial^2}{\partial s \partial t}x(1,0) & \frac{\partial^2}{\partial s \partial t}x(1,1) \end{pmatrix}$$

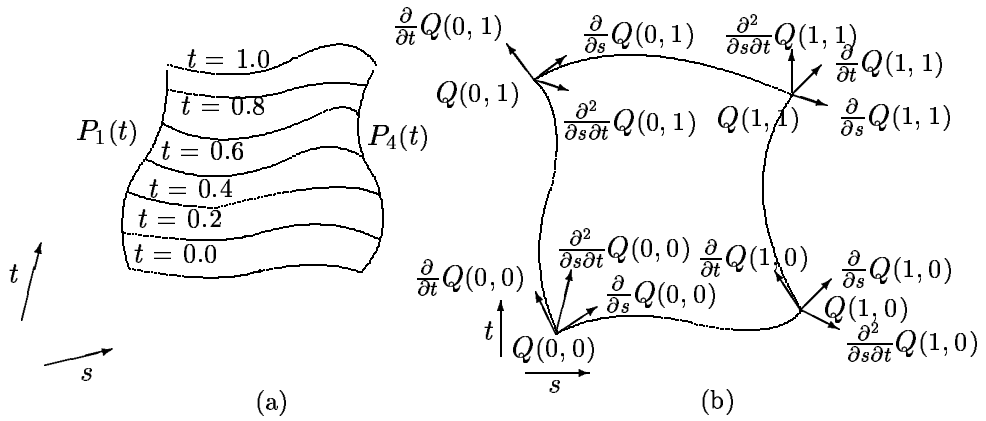


Figura 3.10: Suprafeţe Hermite (a) Interpolarea cubică între $P_1(t)$ şi $P_4(t)$ (b) Tangentele la suprafaţă

Elementele matricei geometrice sunt reprezentate în figura 3.10 (b).

Condiţiile de joncţiune a două suprafeţe Hermite se pun relativ simplu: capetele curbei de margine trebuie să coincidă, iar tangentele la suprafeţe în aceste capete trebuie să fie proporţionale. Astfel matricele de geometrie ale suprafeţelor trebuie să concorde: linia a doua a primei matrice să coincidă cu prima linia a celei de a doua, iar linia a patra a primei matrice să fie proporţională, cu un factor unitar pozitiv, faţă de linia a treia a celei de a doua matrice.

Suprafeţe Bézier

Pentru a defini o suprafaţă Bézier se folosesc cele 4 puncte de control corespunzătoare valorilor $(0,0),(0,1),(1,0),(1,1)$ ale parametrilor s şi t şi în plus alte 12 puncte de control prin intermediul cărora se precizează tangentele la suprafaţă. Astfel geometria unei suprafeţe Bézier este caracterizată prin coordonatele a 16 puncte de control (figura 3.11). Proprietăţile cele mai importante ale acestei suprafeţe sunt includerea în închiderea convexă a punctelor de control şi posibilitatea simplă de subdivizare (se divizează suprafaţa de-a lungul unui parametru, apoi cele două noi suprafeţe de-a lungul celuilalt parametru).

3.10.3 Trasarea suprafeţelor parametrice bicubice

Pentru generarea imaginii unei suprafeţe bicubice parametrice prin evaluare directă, se procedează în modul următor:

1. se parcurg valorile s între 0 şi 1 cu anumit pas $\delta_s = 1/m$;
2. pentru fiecare valoare $s_i = i\delta_s,\ i = 1,\ldots,m$
   - se parcurg valorile lui t între 0 şi 1 cu anumit pas $\delta_t = 1/n$;
   - pentru fiecare valoare $t_j = j\delta_t,\ j = 1,\ldots,n$ se calculează

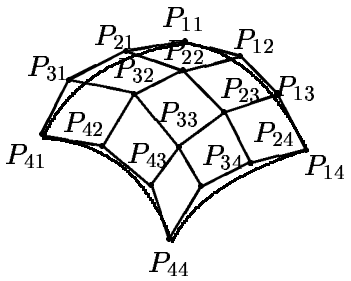


Figura 3.11: Suprafaţă Bézier

(a) coordonatele punctului $x(s_i, t_j)$, $y(s_i, t_j)$, $z(s_i, t_j)$;
(b) proiecţia punctului calculat anterior;

3. se trasează curbele din cele două familii (pentru fiecare s, respectiv pentru fiecare t) prin punctele calculate, ca linii poligonale.

Acurateţea redării suprafeţei depinde de alegerea paşilor incrementali.

Valorile funcţiilor polinomiale se pot calcula iterativ, folosind diferenţe finite şi urmărind algoritmul propus la trasarea curbelor spaţiale.

Pentru algoritmul subdivizării recursive testul de suprafaţă plată pentru o suprafaţă Bézier este realizat prin determinarea planurilor, care trec prin câte trei puncte de control ce aparţin suprafeţei, şi determinarea distanţelor de la celelalte 13 puncte de control la aceste plane: distanţa maximă trebuie să fie sub anumit nivel ε pentru ca subdivizarea în patru noi suprafeţe să nu fie necesară. O suprafaţă aproape plată, specificată, de exemplu, prin colţurile A, B, C, D, poate fi trasată fie prin muchiile cuadrilaterului ABCD, fie prin cele patru triunghiuri determinate de muchii şi punctul interior P, definit ca medie a punctelor de colţ.

3.11 Atributele primitivelor

Imaginea unui primitive poate fi controlată prin atributele sale. Fiecărei primitive îi sunt specifice anumite atribute.

Atributul punctului este culoarea.

Atributele unei linii (drepte sau poligonale) sunt:

(a) stilul liniei (continuă sau solidă, întreruptă, punctată sau corespunzătoare unui şablon)
(b) grosimea liniei;
(c) culoarea;
(d) stilul peniţei (pentru liniile groase, umplerea cu texturi a zonelor dreptunghiulare asociate)

Un şablon (model, pattern) este desemnat printr-o matrice. Inhibarea aprinderii sau aprinderea, conform unei anumite culori, a unui pixel (x, y) de pe o primitivă se realizează în urma testării valorii înscrise în matricea şablon

$m \times n$ la poziţia (x mod m, y mod n).

Aplicarea unui şablon nu este recomandată în cazul liniile întrerupte, deoarece lungimea segmentelor de linii din linia punctată sunt mai mici pentru o linie înclinată decât pentru o linie verticală sau orizontală. Pentru rezolvarea acestei probleme, liniile întrerupte sunt privite ca secvenţe de dreptunghiuri ale căror vârfuri sunt calculate exact, ca funcţie de stilul liniei selectate.

Grosimea unei primitive este distanţa dintre marginile primitivei perpendiculare pe tangenta sa. Îngroşarea unei linii se poate efectua prin următoarele procedee:

1. duplicarea pixelilor pe coloane pentru pante subunitare şi pe linii, în caz contrar. Această tehnică are mai multe dezavantaje:
   (a) liniile se termină întotdeauna vertical sau orizontal;
   (b) grosimea liniilor depinde de înclinare: dacă pentru o linie orizontală grosimea este t, la acelaşi număr de pixeli, o linie cu panta unu are grosimea $t/\sqrt{2}$, obţinându-se un efect negativ asupra strălucirii;
2. alegerea unei peniţe dreptunghiulare al cărui centru străbate primitiva trasată la un pixel. Grosimea este mai mare la segmentele cu panta 1;
3. umplerea ariei dintre margini: se trasează primitiva la distanţele $t/2$ de ambele părţi ale primitivei ideale şi se utilizează o tehnică de umplere. Datorită rotunjirilor este posibil ca linia ideală să nu fie centrată.

Când un sistem permite trasarea în culori, parametrii care furnizează culoarea actuală corespund cu cei din lista de valori-atribute ale sistemului. Indexul de culoare este stocat în frame buffer în locaţia corespunzătoare pixelului. Într-un sistem cu rastru în mai multe culori, numărul de biţi necesari pentru reprezentarea unui pixel în frame buffer depinde de numărul de culori disponibile şi de metoda de stocare a valorilor-culoare.

Atributele poligoanelor, cercurilor şi elipselor sunt cele ale liniei de frontieră plus stilul de umplere (textura interiorului).

Atributele textului sunt multiple:

(a) stil sau font (Roman, Helvica etc);
(b) mod de tipărire (drept, îngroşat, înclinat);
(c) dimensiune (măsurată în puncte tipografice - un punct aproximativ 1/72 inch);
(d) lăţime;
(e) spaţiu între caractere;
(f) spaţiu între linii consecutive;
(g) direcţia de scriere (orizontal, vertical sau sub un anumit unghi).

3.12 Decuparea

În cazul construirii unei imagini ce depăşeşte limitele zonei de lucru de pe ecran sau al spaţiului vizibil, vor apărea o serie de primitive care vor intersecta frontiera (ecranului sau spaţiului vizibil). Este necesară eliminarea din aceste primitive a porţiunilor care ies din zona observabilă. Această operaţie este cunoscută sub numele de decupare (clipping, retezare, tăiere).

Procesul de decupare 2D decide care parte, dacă există, a unei primitive se află în interiorul unei ferestre sau a unei zone de lucru.

Sarcina procesului clasic este eliminarea acelor părţi ale primitivelor care sunt în afara ferestrei din WCS (sistemul coordonatelor lumii reale) şi nu din NDCS (sistemul coordonatelor terminalului normalizate). Se evită, în acest caz, costul transformării de normalizare pentru orice primitivă care nu apare pe ecran. Primitivele care se află într-o fereastră sunt transformate şi transferate într-o zonă de lucru, iar celor care sunt parţial sau complet în afara ferestrei li se aplică un algoritm de decupare.

Algoritmii de decupare sunt aplicaţi şi în cazul selectării unor părţi a primitivelor dintr-o zonă de lucru în vederea copierii, mutării sau ştergerii.

Dificultatea procesului depinde de tipul primitivei căreia i se aplică (deoarece o curbă generală satisface o ecuaţie neliniară, calcularea punctelor de intersecţie cu marginea ferestrei necesită utilizarea unor metode numerice).

Decuparea 3D este efectuată asupra corpurilor 3D pentru încadrarea în volumul de vedere. Se elimină prin acest procedeu proiectarea unor primitive care nu se încadrează în fereastră (şi ulterior în zona de lucru).

Se consideră în cele ce urmează problema decupării bidimensionale a unei primitive plane faţă de o fereastră convexă.

3.12.1 Decuparea unui punct

Se presupune că fereastra este dreptunghiulară şi are limitele: superioară $y = y_{max}$, inferioară $y = y_{min}$, laterală stângă $x = x_{min}$, laterală dreaptă $x = x_{max}$. Dacă fereastra corespunde 1:1 cu spaţiul vizibil al ecranului ($m \times n$ coloane-linii), atunci $y_{min} = 0$, $y_{max} = n - 1$, $x_{min} = 0$, $x_{max} = m - 1$.

Pentru ca un punct $P(x, y)$ să fie vizibil în fereastră trebuie satisfăcute relaţiile

$$x_{min} \leq x \leq x_{max}, \qquad y_{min} \leq y \leq y_{max}.$$

3.12.2 Decuparea segmentelor de linii

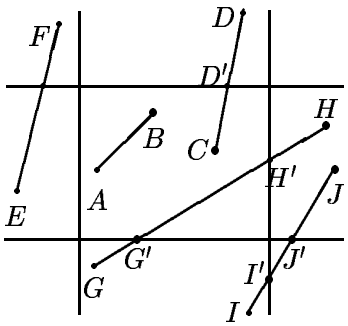
În cazul segmentelor, problema se complică. O soluţie pentru cazul când desenăm pe întreg ecranul şi spaţiul adresabil coincide cu cel vizibil o reprezintă folosirea instrucţiunii eroare imediat după apelarea instrucţiunii de trasare a segmentului în afara spaţiului vizibil. Această soluţie nu rezolvă problema în cazul desenării într-o fereastră mai mică decât ecranul.

O altă variantă este utilizarea unei subrutine de trasare a segmentelor de dreaptă care testează înainte de aprinderea unui pixel, vizibilitatea acestuia faţă de fereastră.

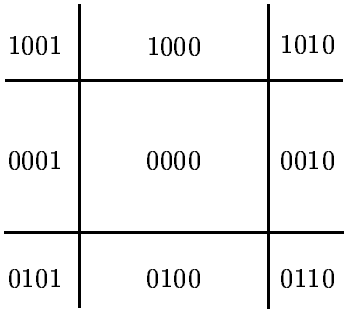
Un segment de dreaptă se află faţă de o fereastră într-una din următoarele patru situaţii (figura 3.12.a):

1. complet în interior (segmentul AB),
2. complet în exterior (segmentele EF şi IJ),
3. un capăt în interior, celălalt în exterior (segmentul CD),

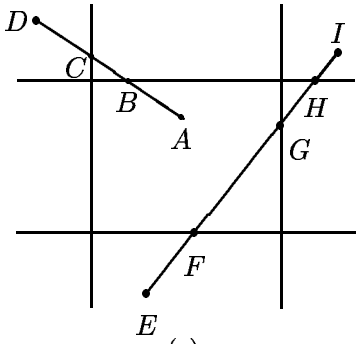
4. ambele capete în exterior şi o parte a segmentului, în interior (segmentul GH).



(a)



(b)



(c)

Figura 3.12: (a) Diferite segmente de dreptă poziţionate distinct faţă de fereastră (b) Cele nouă regiuni din algoritmul Cohen-Sutherland (c) Exemplu pentru algoritmul Cohen-Sutherland

Un algoritmul simplu este următorul:

- dacă capetele segmentului sunt în interior, trasează segmentul;
- dacă numai un capăt este în interior, determină punctul de intersecţie cu marginea ferestrei şi trasează subsegmentul de interior;
- altfel, calculează intersecţiile cu liniile ce definesc marginea ferestrei şi dacă punctele de intersecţie sunt pe segmentele de dreaptă ale marginii ferestrei, trasează subsegmentul dintre acestea.

Teste de acceptare/respingere

Se consideră cazul unei ferestre dreptunghiulare. Algoritmul de mai sus poate fi perfecţionat prin efectuarea unor teste simple asupra capetelor segmentului. Segmentul care are ambele capete vizibile, este vizibil în întregime (fereastra

este o mulţime convexă). Testul care verifică dacă ambele capete ale segmentului sunt în fereastră, poartă numele de test de acceptare. Orice segment care are ambele capete de aceeaşi parte a ferestrei (sus, jos, stânga, dreapta), este complet invizibil. Testul care verifică această condiţie se numeşte test de respingere. Prin adăugarea testelor de acceptare şi respingere la algoritmul de trasare a unui segment de dreaptă se obţine un algoritm a cărui eficienţă se manifestă cu precădere în 2 situaţii:

(a) dacă majoritatea segmentelor de reprezentat se află în interiorul ferestrei;
(b) dacă majoritatea segmentelor de reprezentat se află în afara ferestrei.

Algoritmul Cohen-Sutherland

Dacă o linie nu este nici respinsă, nici acceptată, atunci se calculează într-o ordine prestabilită intersecţiile cu dreptele: $y = y_{max}$, $x = x_{max}$, $y = y_{min}$, $x = x_{min}$ îndepărtând porţiunile de segment care se situează în afara ferestrei (dreptunghiulare) . Rezultă astfel un nou segment, iar procedura se buclează asupra ei însăşi până când un segment rezultat este acceptat sau respins.

Se divide WCS în nouă regiuni determinate de marginile ferestrei (figura 3.12.b). În prezentarea clasică, algoritmul Cohen-Sutherland asociază fiecărui capăt al unui segment un cod de 4 cifre:

(a) prima cifră este 1 dacă punctul este deasupra ferestrei, 0 altfel;
(b) a doua cifră este 1 dacă punctul este sub fereastră, 0 altfel;
(c) a treia cifră este 1 dacă punctul este în dreapta ferestrei, 0 altfel;
(d) a patra cifră este 1 dacă punctul este în stânga ferestrei, 0 altfel.

Puterea algoritmului constă în faptul că odată determinate codurile asociate capetelor segmentului, testele de acceptare sau respingere se realizează prin operaţii logice rapide (de exemplu, în limbajul C: testul de acceptare are succes dacă $cod_1|cod_2$ este zero, iar testul de respingere, dacă $cod_1\&cod_2$ nu este zero). Astfel, segmentul AB din figura 3.12 (a) este acceptat, iar EF respins. Rămân de analizat cazurile în care capetele segmentului sunt în regiuni diferite.

Se consideră că primul capăt al segmentului (x_1, y_1) este în afara ferestrei (eventual printr-o inversare). Atunci codul asociat este nenul. Se determină primul bit nenul, aflând astfel de care parte a ferestrei se află primul capăt. Presupunem că primul bit este nenul, deci $y_1 > y_{max}$. Se elimină porţiunea din segment ce se află deasupra liniei $y = y_{max}$. Fie $(x_1{}', y_1{}')$ punctul de intersecţie cu $y = y_{max}$. Se determină codul punctului nou, necesar următorului pas. Analog se procedează pentru partea de jos, stânga şi dreapta ferestrei.

Se consideră cele două segmente din figura 3.12 (c). Punctul A are codul 0000, iar punctul D, 1001. Segmentul nu poate fi acceptat sau respins. Codul lui D indică faptul că segmentul trece peste latura de sus şi peste latura stângă a ferestrei. Urmărind ordinea biţilor din codul lui D se determină, prima dată, intersecţia segmentului cu latura de sus. Codul punctului B de intersecţie este 0000, astfel încât segmentul AB este acceptat. Segmentul EI necesită iteraţii multiple. Cum E are codul 0100<1010, codul lui I, se consideră primul punct exterior. Testând codul lui E, se determină intersecţia F a lui EI cu dreapta suport a laturii de jos. Deoarece codul lui F este 0000, iar al lui I este 1010,

testul de acceptare/respingere nu are succes. Punctul exterior este I. Testând codul lui I de la stânga la dreapta se determină laturile ferestrei pe care FI le poate intersecta. Prima este latura de sus. Se determină H, punctul de intersecţie. Codul lui H este 0010. Se face o diviziune a segmentului obţinând FG care este acceptat.

Algoritmul, în această variantă comună, nu este eficient: deoarece testele şi decuparea se efectuează într-o anumită ordine, se fac adesea operaţii suplimentare (de exemplu, determinarea punctului H de pe EI). Algoritmul Nicholl-Lee-Nicholl ocoleşte calculul intersecţiilor externe ferestrei.

Algoritmul bisecţiei

Algoritmul elimină calculele necesare pentru determinarea intersecţiilor. La început se folosesc testele de respingere şi acceptare trivilă. Dacă un segment nu verifică nici unul din aceste teste, el este împărţit în două jumătăţi egale. Cele două jumătăţi sunt analizate din nou şi procesul continuă până la acceptare sau respingere. De obicei, la fiecare iteraţie se poate accepta sau respinge jumătate din segmentul analizat. Acest algoritm ajunge la un rezultat după maximum $\log_2 N$ paşi, unde N este dimensiunea maximă a segmentului (pe orizontală sau verticală), măsurată în pixeli.

Algoritmul Cyrus-Beck de decupare a liniilor parametrice

Metoda este utilizată pentru a tăia o linie bidimensională printr-un dreptunghi sau un poligon convex, sau o linie tridimensională printr-un poliedru. Algoritmul Cohen-Sutherland determină intersecţiile segmentului de linie cu liniile suport ale laturilor poligonului-fereastră substituind valorile coordonatelor astfel obţinute în ecuaţia segmentului de dreaptă. Algoritmul liniei parametrice determină valoarea parametrului t din reprezentarea parametrică a segmentului de linie pentru punctele în care segmentul intersectează liniile infinite pe care se află laturile ferestrei. Dacă poligonul este un dreptunghi, se calculează patru valori ale parametrului t, care apoi sunt comparate pentru a determina acelea care reprezintă puncte interioare segmentului. Se reduce timpul faţă de algoritmul Cohen-Sutherland deoarece se evită saltul repetitiv necesar pentru a tăia segmentul prin laturile multiple ale dreptunghiului.

Se defineşte un sens de parcurgere a laturilor poligonului. Semiplanul interior asociat unei laturi este semiplanul determinat de aceasta şi care conţine fereastra. Normala la o latură se consideră, în acest caz, orientată spre exteriorul ferestrei.

În figura 3.13 (a) s-au notat cu P_0, P_1 capetele segmentului şi E_i o latură a ferestrei. Linia suport a segmentului este reprezentată parametric

$$P(t) = P_0 + (P_1 - P_0)t.$$

Se presupune că $x_0 < x_1$. Dacă P_{E_i} este un punct de pe latura E_i, iar N_i este vectorul normal la latură, atunci, valoarea t, corespunzătoare punctului de

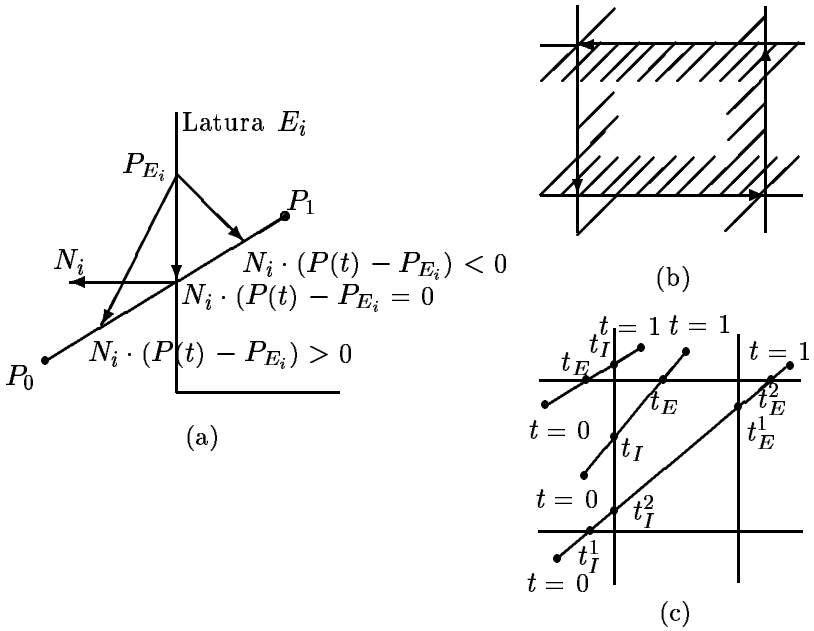


Figura 3.13: Algoritmul Cyrus-Beck (a) O latură şi segmentul ce se decupează (b) Semiplanele interioare definite de laturi (c) Exemplu

intersecţie notată t_i, satisface

$$N_i \cdot [P(t_i) - P_{E_i}] = 0$$

unde $\cdot$ reprezintă produsul scalar. Atunci

$$N_i \cdot [P_0 + (P_1 - P_0)t_i - P_{E_i}] = N_i \cdot (P_0 - P_{E_i}) + N_i \cdot (P_1 - P_0)t_i = 0,$$

deci

$$t_i = \frac{N_i \cdot (P_0 - P_{E_i})}{-N_i \cdot (P_1 - P_0)}.$$

Pentru fiecare segment se determină cele patru valori t_i corespunzătoare liniilor suport ale laturilor ferestrei, E_i. Pasul următor constă în determinarea valorilor ce corespund intersecţiilor cu laturile. Se adaugă la listă $t = 0$ şi $t = 1$. Valorile $t \notin [0, 1]$ sunt eliminate deoarece nu conduc la puncte din interiorul segmentului P_0P_1. Valorile t rămase sunt sortate (maxim şase puncte). Se determină punctele, t_I, în care segmentul "intră" în interiorul unui semiplan, prin $N_i \cdot (P_1 - P_0) < 0$ (unghi mai mare decât $\pi/2$) şi punctele t_E unde segmentul iese în exterior, prin $N_i \cdot (P_1 - P_0) > 0$ (unghi mai mic decât $\pi/2$). Se consideră $t = 0$ cel mai mic t_I, iar $t = 1$ cel mai mare t_E. Segmentul interior ferestrei este definit de cel mai mare t_I şi cel mai mic t_E. În cazul primei linii din figura 3.13 (c), cel mai mic t_I este mai mare decât cel mai mare t_E, astfel încât nici o porţiune din segment nu se află în interiorul dreptunghiului.

Algoritmul Nicholl-Lee-Nicholl

Se dă un segment de dreaptă PQ şi un dreptunghi de decupare, ca în figura 3.14. Determinând poziţia relativă a lui Q faţă de liniile ce trec prin P şi colţurile ferestrei, se află care laturi ale ferestrei intersectează PQ.

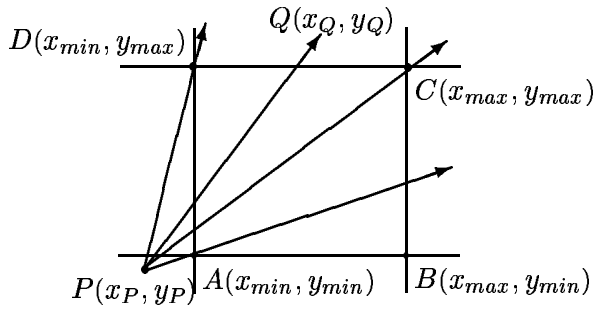


Figura 3.14: Algoritmul Nicholl-Lee-Nicholl

Se studiază poziţia lui P, determinând sectorul (din cele nouă posibile) în care se află. Funcţie de această poziţie se efectuează diferite teste asupra lui Q. Presupunem că P se află în regiunea de stânga-jos (figura 3.14). Dacă Q se află sub y_{min} sau la stânga lui x_{min}, atunci PQ nu poate intersecta fereastra. Acelaşi lucru se poate afirma şi dacă Q se află la stânga liniei de la P la colţul de stânga sus sau dacă Q se află la dreapta liniei de la P la colţul din dreapta jos. Se verifică de asemenea poziţia lui Q relativ la vectorul de la P la colţul din stânga-jos. Fie cazul în care Q este deasupra. Dacă Q se află sub y_{max}, Q este în interiorul dreptunghiului sau la stânga: PQ intersectează latura $x = x_{min}$ sau/şi latura $x = x_{max}$. Dacă Q se află peste y_{max}, Q este în regiunea de sus sau stânga-sus. Dacă Q este deasupra liniei dintre P şi colţul dreapta-sus, PQ intersectează $x = x_{min}$ şi $y = y_{max}$, altfel $x = x_{min}$ şi $x = x_{max}$.

De exemplu, pentru a testa dacă Q este în stânga segmentului de la P la colţul de dreapta-sus, (x_{max}, y_{max}), se verifică dacă

$$(y_Q - y_P)(x_{max} - x_P) > (y_{max} - y_P)(x_Q - x_P).$$

Dacă rezultatul este pozitiv, se calculează intersecţia cu latura de sus a ferestrei. În caz contrar, se determină intersecţia cu latura din dreapta: ordonata y a intersecţiei este

$$y_P + (y_Q - y_P)\frac{x_{max} - x_P}{x_Q - x_P}$$

Calcule similare se efectuează pentru celelalte colţuri. Valorile $y_Q - y_P$, $x_Q - x_P$ se repetă şi pot fi calculate o singură dată.

Algoritmul presupune mai puţine împărţiri şi doar o treime din numărul de comparaţii necesare algoritmului Cohen-Sutherland.

3.12.3 Decuparea poligoanelor

Rezultatul decupării faţă de o fereastră poligonală convexă este: niciunul, unul (cazul poligon decupat convex) sau mai multe poligoane (cazul poligon decupat concav).

Teste de acceptare/respingere

Considerăm o linie poligonală cu un număr mare de segmente. Abordarea clasică a clippingului presupune aplicarea algoritmului la fiecare segment de dreaptă în parte. Efortul de calcul este mare, mai ales dacă linia poligonală se află în exteriorul ferestrei. Metoda ariei mărginite (bounding boxes) determină în prealabil cel mai mic dreptunghi care încadrează linia poligonală – se determină colţurile acestuia ca valori minime şi maxime ale vârfurilor liniei poligonale. Dacă dreptunghiul este în afara ferestrei, linia poligonală nu este trasată. Dacă este complet în interior, linia poligonală este trasată în întregime. Dacă dreptunghiul şi fereastra au doar o porţiune comună se trece la decupare pe fiecare segment de dreaptă în parte.

Algoritmul Sutherland-Hodgman

Algoritmul utilizează o strategie divide-et-impera (divide-and-conquer) pentru a decupa un poligon convex sau concav relativ la un poligon convex care defineşte fereastra. Intrarea în algoritm este o listă a vârfurilor, iar ieşirea are o structură similară. Algoritmul comportă un număr de etape egal cu numărul laturilor poligonului-fereastră (patru în cazul unui dreptunghi). Într-o etapă, poligonului curent i se aplică tehnica de tăiere relativă la una din laturile poligonului-fereastră, rezultând o listă modificată de vârfuri (figura 3.15.a). La fiecare etapă se parcurg un număr de paşi egali cu numărul laturilor poligonului care se taie. În fiecare pas se adaugă la lista de vârfuri 0, 1 sau 2 vârfuri noi ce aparţin laturii curente a poligonului-fereastră.

Se consideră latura poligonului de decupat ce porneşte de la vârful S la vârful P (figura 3.15.b). În primul caz, când latura poligonului este complet în interiorul ferestrei, se adaugă P la lista de ieşire. În al doilea caz, punctul de intersecţie I este introdus ca nou vârf. În cazul al treilea, ambele vârfuri sunt în afara ferestrei, astfel încât nu se produc modificări în lista de ieşire. În cazul al patrulea, atât punctul de intersecţie I, cât şi P sunt adăugate la lista de ieşire.

Algoritmul Liang-Barsky

Algoritmul presupune decuparea faţă de o fereastră dreptunghiulară. Se consideră poligonul care va fi tăiat reprezentat prin secvenţa vârfurilor P_1, $P_2, \ldots,$ P_n, parcurse într-un anumit sens (trigonometric sau conform fusului orar). Fiecare latură este considerată ca un vector (pornind, de exemplu, de

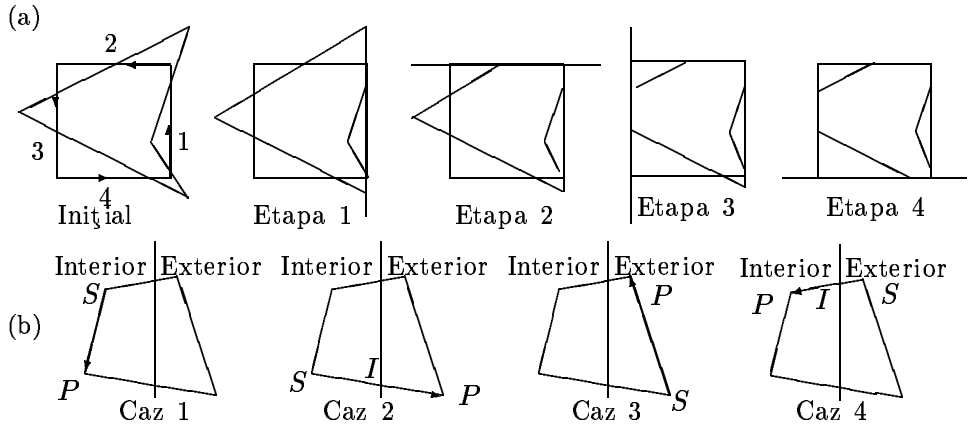


Figura 3.15: Algoritmul Sutherland-Hodgman (a) Etapele algoritmului (b) Modificarea listei vârfurilor

la un P_i la P_{i+1}). Planul este divizat în nouă regiuni determinate de laturile ferestrei. Fiecare latură a ferestrei împarte planul în două semiplane. Semiplanul care conţine fereastra este semiplan de interior. Cele nouă regiuni se identifică funcţie de numărul de semiplane interioare care sunt incluse în interiorul lor (figura 3.16.a). Regiunile notate "interior 2" se numesc regiuni de colţ, iar cele notate "interior 3", regiuni de latură.

Dacă ultima latură ce intersectează fereastra generează un vârf la marginea de sus a ferestrei şi următoarea latură ce intersectează fereastra va crea un vârf la dreapta ferestrei, poligonul de ieşire va trebuie să conţină colţul din dreapta sus a ferestrei (figura 3.16.b). În general, o latură care intră într-o regiune de colţ adaugă un colţ de fereastră ca vârf de ieşire. Un asemenea vârf este numit vârf turnant.

Se determină valorile parametrului t ale intersecţiilor liniei ce conţine P_iP_{i+1} cu laturile ferestrei. Două intersecţii sunt potenţial de intrare, t_{in_1}, t_{in_2}, două de ieşire t_{out_1}, t_{out_2}. Valoarea t_{in_1} este cea mai mică, iar t_{out_2} este cea mai mare deoarece orice linie neorizontală şi neverticală porneşte dintr-o regiune de colţ şi sfârşeşte într-o altă regiune de colţ. Celelalte valori se află între ele şi pot fi în orice ordine. Dacă $t_{in_2} < t_{out_1}$, linia intersectează fereastra, altfel linia trece printr-o regiune de colţ (figura 3.16.c).

Relaţiile dintre valorile $t = 0$, $t = 1$ şi t_{in_1}, t_{in_2}, t_{out_1}, t_{out_2} caracterizează contribuţiile laturii la poligonul de ieşire. Dacă $0 \leq t_{in_2} < t_{out_1} \leq 1$, segmentul de dreaptă determinat de valorile t_{in_2} şi t_{out_1} este latură a poligonului decupat. În caz contrar, linia nu intersectează fereastra, ci porneşte dintr-o regiune de colţ, trece prin alta şi se termină în a treia. Intrarea segmentului în regiunea de colţ intermediară (ce necesită adăugarea unui colţ al ferestrei în lista vârfurilor poligonului decupat) este caracterizată prin $0 \leq t_{out_1} < t_{in_2} \leq 1$.

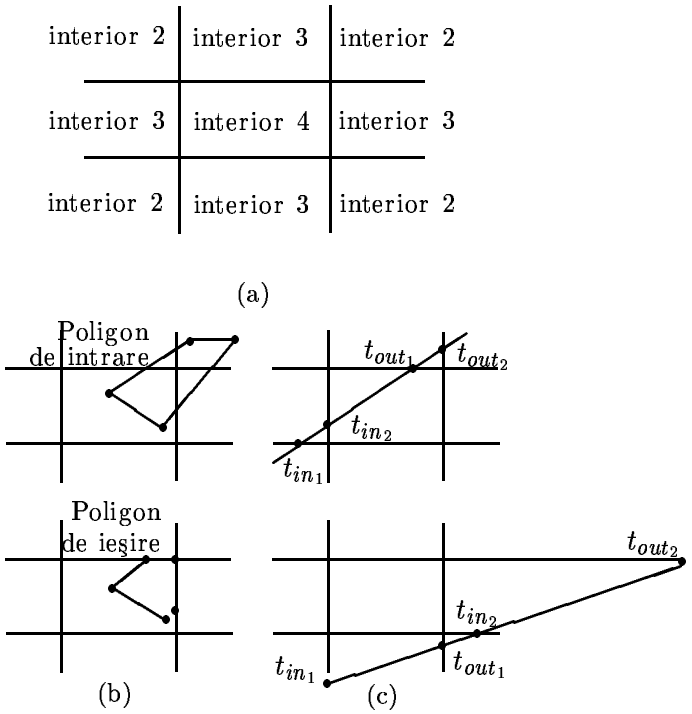


Figura 3.16: Algoritmul Liang-Barsky (a) Definirea regiunilor (b) Vârf turnant (c) Intersecţiile cu laturile ferestrei

3.12.4 Decuparea curbelor plane

Cercurile şi elipsele pot fi aproximate cu secvenţe de linii scurte, astfel încât pot fi tratate ca linii poligonale cu număr mare de segmente.

Conicele, în general, sunt construite prin metode incrementale care permit includerea testelor de vizibilitate în fereastră.

În cazul cercurilor sau elipselor se poate efectua un test preliminar de acceptare/respingere asupra dreptunghiului minim ce încadrează respectiva conică.

Structura algoritmului incremental de trasare a curbei generale de grad doi $f(x, y) = 0$ cuprinde trei faze:

1. se calculează δ, Δ şi se separă cazul trasării conicelor degenerate;
2. se calculează toate intersecţiile curbei cu laturile ferestrei de vizualizare. În total pot exista maximum 8 puncte de intersecţie. Pentru fiecare punct de intersecţie se cercetează semnul funcţiei f la dreapta arcului de curbă care porneşte din acest punct şi este orientat spre interiorul ferestrei. Dacă $f < 0$ la dreapta curbei, atunci punctul de intersecţie este neglijat, deoarece arcul de curbă trebuie trasat începând de la celălalt capăt, care corespunde altui punct de intersecţie (algoritmul incremental porneşte de

la presupunerea că $f < 0$ la stânga direcţiei de înaintare). În caz contrar se trasează arcul. Pentru calculul intersecţiei cu laturile ferestrei, de exemplu cu $y = y_{min}$, se recomandă modificarea ecuaţiei $f(x, y_{min}) = 0$. Se observă că $f(x, y_{min}) + f(x, y_{min} - 1) < 0$ (respectiv > 0) pentru toate punctele (x, y_{min}) mai apropiate (respectiv mai depărtate) de curbă decât $(x, y_{min} - 1)$. În cazul limită, de egalitate cu 0, punctele menţionate sunt egal depărtate de curba ideală. Astfel pentru determinarea intersecţiei cu $y = y_{min}$ se recomandă rezolvarea ecuaţiei (analog factorului de decizie de la trasarea cercurilor)

$$f(x, y) + f(x, y - 1) = 0,$$

cu $y = y_{min}$.

3. se trasează curbele care nu intersectează marginile ferestrei de vizualizare (cercuri sau elipse complet incluse în fereastră): se calculează centrul curbei (x_c, y_c) şi se apelează procedura de trasare incrementală începând cu punctul din dreapta de intersecţie cu orizontala $y = y_c$.

3.12.5 Decuparea 3D

Pentru corpurile descrise prin mai multe puncte, dintre care doar o parte se află în volumul de observare, se preferă retezarea segmentelor în spaţiu, păstrând numai porţiunile interioare volumului observabil, şi abia după aceea proiecţia în 2D. Procesul este cunoscut sub denumirea de decupare 3D (3D clipping).

Pot fi generalizaţi atât algoritmul Cohen-Sutherland cât şi cel al bisecţiei.

În algoritmul Cohen-Sutherland 3D, spaţiul este împărţit în 27 de zone, pentru care se alcătuieşte câte un cod de 6 cifre. Primele patru sunt aceleaşi ca şi în plan; a 5-a valoare este 1 dacă punctul se află în faţa volumului de observare şi 0 altfel, a 6-a valoare este 1 dacă punctul se află în spatele volumului de observare (figura 3.17).

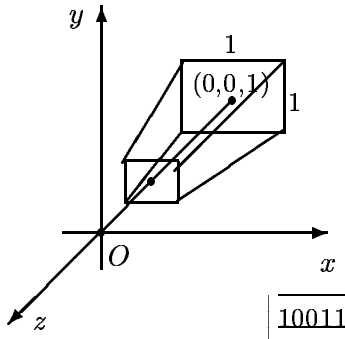
Pentru calculul intersecţiilor cu planele limitatoare, în cazul algoritmului Cohen-Sutherland 3D, este comod să se transforme mai întâi volumul de observare în volum de observare canonic. În acest caz calculele şi comparaţiile se simplifică deoarece planele limitatoare înclinate au panta 1 sau -1.

Coordonatele intersecţiei se pot calcula folosind forma parametrică a ecuaţiei unei drepte ce trece prin două puncte. Fie segmentul P_1P_2 cu extremităţile $P_1(x_1, y_1, z_1)$, $P_2(x_2, y_2, z_2)$. Se calculează intersecţia cu planul $y = z$. Forma parametrică a dreptei este

$$\begin{cases} x = (x_2 - x_1)t + x_1, \\ y = (y_2 - y_1)t + y_1, \\ z = (z_2 - z_1)t + z_1, \end{cases} \qquad 0 \le t \le 1.$$

Deoarece $y = z$, are loc $(y_2 - y_1)t + y_1 = (z_2 - z_1)t + z_1$, de unde se obţine t corespunzător punctului de intersecţie:

$$t_{plan(y=z)} = \frac{z_1 - y_1}{(y_2 - y_1) - (z_2 - z_1)}.$$



100101	100001	101001
000101	000001	001001
010101	010001	011001

Spate

100100	100000	101000
000100	000000	001000
010100	010000	011000

Mijloc

100110	100010	101010
000110	000010	001010
010110	010010	011010

Faţă

Figura 3.17: Algoritmul Cohen-Sutherland pentru decupare 3D

Segmentul P_1P_2 intersectează planul $y = z$, dacă $0 \leq t_{plan(y=z)} \leq 1$.

3.13 Reprezentarea corpurilor tridimensionale

3.13.1 Scheme de reprezentare ale solidelor rigide

Atunci când dorim reprezentarea pe un ecran a unui solid din spaţiul 3D apar o serie de probleme a căror soluţie trebuie găsită:

(a) cum modelăm corpul în spaţiul 3D;
(b) cum construim o reprezentare a modelului;
(c) cum transpunem reprezentarea 3D în 2D.

Reprezentările simple care permit recunoaşterea unor suprafeţe şi efectuarea de calcule relative la aceste suprafeţe se obţin printr-unul din următoarele procedee de modelare a corpurilor:

- modelarea suprafeţelor;
- modelarea solidului.

În primul caz, un corp este modelat prin precizarea frontierei sale, deci prin modelarea unui suprafeţe. Aceasta din urmă poate fi obţinută ca suprafaţă strâmbă în spaţiu, compusă din porţiuni de suprafeţe curbe. În cazul modelării solidului, corpul este construit prin alăturarea unor volume elementare, cuburi, piramide, tetraedre, sfere, cilindri care să aproximeze cât mai bine forma dorită. Ambele procedee se încadrează în clasa mai generală a modelării geometrice (numită şi modelare a formelor).

În literatura de specialitate sunt considerate următoarele scheme de reprezentare a solidelor rigide:

1. Instanţierea primitivelor pure. Se definesc familii de entităţi, numite primitive generice sau pure, ai căror membrii se numesc instanţe (exemplare) ale primitivelor. O primitivă generică poate fi privită ca un prototip

parametrizat al unei piese mecanice. Schema se pretează la tehnologiile de grup când se modelează familii de obiecte, în care membrii au aceeaşi structură, dar diferă prin parametrii, un set dat de parametrii identificând în mod unic un membru al familiei. Schema promovează standardizarea prototipurilor şi este cel mai des utilizată în industrie. Dezavantajul este imposibilitatea de combinare a primitivelor generice pentru a crea o nouă primitivă mai complexă.

2. Enumerarea ocupării spaţiale. Spaţiul este împărţit într-o reţea fină de celule tridimensionale, de obicei cuburi, numite voxel (volume element) prin analogie cu pixel (picture element). Un solid este reprezentat prin lista celulelor pe care le ocupă, adică a celulelor care conţin material.
3. Descompunerea în celule. Un solid este descompus în celule solide elementare, fără găuri, ale căror interioare sunt două câte două disjuncte. Un solid este rezultatul lipirii celulelor componente care satisfac anumite condiţii de potrivire a frontierelor. Schema constituie o generalizare a celei de a doua scheme, eliminând restricţiile referitoare la amplasarea celulelor în poziţii fixe şi la dimensiune şi forme fixe pentru celule. Se bazează pe teoria triangulaţiei.
4. Geometrie solidă constructivă. O schemă CSG (Constructive Solid Geometry) defineşte un solid ca o combinaţie de blocuri constructive solide, combinaţie realizată prin intermediul operaţiilor de tip adunare sau scădere volumetrică. Este o generalizare a celei de a treia scheme, înlocuind operatorul de lipire cu operatori care nu necesită satisfacerea unor condiţii de frontieră. Schema se bazează pe teoria mulţimilor regulate.
5. Măturare. Un solid sau o suprafaţă mărginită care se deplasează de-a lungul unei traiectorii "mătură" un volum. În această schemă un solid poate fi reprezentat printr-un cuplu: (corp în mişcare, traiectorie). Metoda foloseşte noţiunile matematice de produs de mulţimi sau operaţii cu mulţimi asupra unui număr infinit de mulţimi.
6. Interpolare. Un solid poate fi definit ca reuniunea tuturor segmentelor de dreaptă PQ pentru care P şi Q aparţin unei mulţimi A şi respectiv B din spaţiul bidimensional. Deci un solid se poate reprezenta prin cuplul (A, B).
7. Reprezentarea prin frontiere. În timp ce toate metodele anterioare reprezintă solidele direct, reprezentările prin frontiere sunt indirecte prin aceea că ele reprezintă direct (explicit) frontierele topologice ale solidului şi nu solidul însuşi. Această metodă înlocuieşte problema reprezentării unei anumite mulţimi prin reprezentarea unei mulţimi de dimensionalitate redusă.

3.13.2 Metode de construcţie a reprezentărilor

Există trei metode de a construi modele geometrice ale solidelor rigide:

- Descrierea procedurală. Această metodă constă în reprezentarea unui obiect printr-un program a cărui execuţie produce un exemplar din res-

pectivul obiect. O procedură obiect poate avea parametrii, devenind un obiect generic. Descrierea procedurală este convertită într-un model prin compilarea şi executarea procedurilor obiect în contextul în care aceste proceduri sunt activate. Practic, limbajele de descriere oferă facilităţi de apelare ale unor proceduri standard de descriere: pentru linie, cerc, curbe spline, cilindri. Procedurile se pot apela unele pe altele, permiţând descrierea unor obiecte complexe. Metoda este anevoioasă deoarece utilizatorii trebuie să aibă cunoştiinţe de programare. Se pretează în cazul instanţierii primitivelor pure.

- Conversia bazei de date. Constă în construirea modelului unui obiect prin vârfuri şi muchii – tip cadru de sârmă (wire frame) – pornind de la mai multe proiecţii în 2D ale obiectului. Proiecţiile obiectului în 2D se obţin fie prin metoda descrierii procedurale, fie prin grafică interactivă.
- Grafică interactivă. Se bazează pe organizarea tuturor comenzilor necesare realizării şi manipulării desenelor în grupuri relaţionate logic, numite liste de meniu. Utilizatorul transmite comanda dorită sistemului prin selectarea acesteia din lista de meniu cu ajutorul unui dispozitiv de interacţiune. Această modalitate de a dirija un sistem grafic se numeşte tehnica meniului. Listele de meniuri sunt organizate ierarhic într-o structură de arbore. Componenţa listelor de meniuri şi structura ierarhică se obţin în urma unui proces de generare.

3.13.3 Reprezentarea corpurilor prin frontiere

Variantele cele mai des utilizate sunt:

(a) reprezentarea prin puncte (reprezentarea prin secţiuni transversale);
(b) reprezentarea tip wire-frame (cadru de sârmă);
(c) reprezentarea prin reţea de poligoane (reprezentarea poliedrală).

Reprezentarea prin puncte şi secţiuni transversale

Dacă obiectul de modelat are o suprafaţă compusă din faţete poligonale, se poate obţine un model exact al acestuia înmagazinând în memorie coordonatele vârfurilor acestor faţete.

Deoarece corpurile reale au de obicei suprafeţe curbe, modelele sunt, în majoritatea cazurilor, aproximative. Pentru mulţimea punctelor care modelează un corp (în general suprafaţa acestuia) se pot pune două tipuri de condiţii:

i) suprafaţa corpului să treacă efectiv prin punctele date;
ii) distanţa dintre suprafaţa reală şi punctele care definesc modelul să fie mică.

Reprezentarea prin puncte se utilizează în domeniul medicinei şi chimiei. În medicină, corpul se reprezintă printr-o reţea de puncte dispuse după secţiuni transversale. În chimie, atomii sunt modelaţi prin sfere, reprezentate prin puncte situate în nodurile unui reţele geodezice.

Elementul de bază al structurilor de date folosite pentru modelare este lista de vârfuri. Un vârf este un punct de pe suprafaţa unui model în care se întâlnesc

sau din care pornesc una sau mai multe linii folosite în reprezentarea corpului respectiv.

O listă completă de vârfuri conţine coordonatele tuturor punctelor care descriu corpul. O listă de vârfuri trunchiată conţine numai informaţiile referitoare la o submulţime de puncte, precum şi alte informaţii necesare pentru determinarea coordonatelor celorlalte puncte folosind simetrii, rotaţii, translaţii şi scalări. Formele trunchiate sunt folosite pentru depozitarea informaţiei necesare regenerării periodice a corpului.

Un corp de observat poate avea mai multe componente. Este util ca acestea să poată fi manipulate independent. Pentru aceasta, un program trebuie să permită utilizatorului alegerea unui subcorp sau a unui grup de subcorpuri asupra cărora va opera în continuare. Fiecare subcorp poate fi tratat la rândul său ca fiind compus din mai multe părţi. În acest fel se stabileşte o ierarhie pe mai multe nivele, a modelului creat. Fiecare element de la baza unui astfel de ierarhii este descris ca un corp independent. El este identificat printr-o pereche de indici în lista de vârfuri care indică primul şi ultimul punct ce aparţin subcorpului considerat.

O metodă utilizată pentru selectarea subcorpurilor este cea a atributelor activ/pasiv. Un atribut activ/pasiv poate lua de exemplu forma unei matrici $n_t \times n_{max}$, unde n_t este numărul nivelelor ierarhice, iar n_{max} este numărul maxim se subcorpuri ce se află pe un nivel ierarhic. În interiorul acestei matrici se pot folosi trei coduri: 0-inexistent, 1-activ, 2-pasiv. Fiecare element al matricii corespunde unui subcorp.

Reprezentarea folosind listă de vârfuri, ierarhie şi masive de atribute este întâlnită în literatură sub denumirea de reprezentare prin masive de cote şi atribute pe mai multe nivele (multilevel coordinate and attribute arrays).

Reprezentările prin secţiuni transversale decurg direct din reprezentările prin puncte situate în secţiuni transversale paralele între ele. Punctele situate în aceeaşi secţiune se unesc între ele astfel încât să pară reprezentat conturul secţiunii respective. Uneori, pentru o mai bună precizare a formei obiectului modelat se folosesc două curbe sau linii poligonale denumite ecuator şi meridian. Adăugând curbe longitudinale unei reprezentări prin secţiuni transversale se obţin reprezentările de tip wire-frame.

Reprezentarea prin cadru de sârmă

Într-o asemenea schemă, un corp este reprezentat ca o mulţime de segmente de dreaptă sau porţiuni de curbe.

Datele de intrare pot fi organizate astfel:

(a) segmente explicite: pentru fiecare segment se precizează un indice (opţional) şi coordonatele extremităţilor segmentului. Nu se utilizează lista de vârfuri.

(b) segmente implicite: fiecare segment este precizat printr-o pereche de indici care identifică capetele acestuia într-o listă de vârfuri (necesarul de memorie faţă de forma precedentă scade);

(c) linii date prin indici: atunci când o linie poligonală poate fi descrisă prin concatenarea unui şir de segmente, este mult mai indicată utilizarea informaţiilor: indice linie, număr de puncte pe linie, indici care reprezintă capetele segmentelor ce compun linia în lista de vârfuri;

(d) secţiuni transversale şi linii longitudinale: se procedează precum în cazul liniilor date prin indici, dar curbele transversale sunt în marea lor majoritate chiar reprezentările secţiunilor transversale.

Reprezentarea wire-frame a unui obiect nu permite definirea unor suprafeţe şi, deci, calcularea ariilor, volumelor, maselor, centrelor de greutate sau afişarea pe ecran a porţiunii vizibile a obiectului analizat.

Deşi reprezentarea wire-frame este simplistă şi nu furnizează informaţii complete asupra geometriei corpului, datorită uşurinţei de utilizare şi a rapidităţii de afişare a reprezentării pe terminal, este mult utilizată. Folosind această tehnică se pot obţine viteze mari de lucru şi chiar efecte interesante de animaţie.

Reprezentarea prin reţea de poligoane

O reţea de poligoane este o colecţie de laturi, vârfuri şi poligoane conectate astfel încât fiecare latură este partajată de cel mult două poligoane (o latură conectează exact două vârfuri, iar un poligon este o secvenţă închisă de laturi).

Dacă obiectul real are suprafeţe curbe, modelul poligonal este aproximativ. Aproximarea poate fi îmbunătăţită prin mărirea numărului de faţete poligonale plane care modelează o suprafaţă curbă. Dezavantajul constă în necesarul sporit de memorie, dar este de reţinut faptul că algoritmii care procesează suprafeţele poligonale plane sunt mult mai simpli decât cei pentru suprafeţe curbe.

Elementul de bază pentru modelarea poliedrală este lista de vârfuri. Punctele ale căror coordonate sunt înscrise în listă sunt vârfurile poliedrului. Acestea împreună cu muchiile poliedrului şi cu faţetele poliedrului constituie elementele definitorii ale reţelei poligonale.

Stocarea informaţiilor necesare reprezentării vârfurilor, muchiilor şi faţetelor poligonale se face sub diferite forme. Criteriile de alegere a formei de stocare privesc două caracteristici ale programului: memoria necesară şi viteza de lucru.

Vârfurile unui faţete sunt precizate în două moduri:

- explicit: vârfurile care determină o faţetă fie ca indici în lista de vârfuri, fie prin coordonate (parcugerea faţetelor prin citire) în ordinea descrierii poligoanelor;
- implicit: vârfurile care determină o faţetă sunt determinate prin indici în lista de vârfuri pe baza unui algoritm de parcurgere a acestei liste (parcurgerea faţetelor prin generare) în ordinea activării faţetelor.

Parcurgerea faţetelor prin citire. Reprezentarea reţelelor poligonale se face:

(a) explicit, prin listă de poligoane date explicit. În acest caz lista de vârfuri lipseşte deoarece fiecare poligon este reprezentat de o structură de forma (indice-poligon, listă de coordonate). Lista de coordonate are structura

unei înregistrări (nr.de vârfuri, coordonatele vârfurilor). Când modelul este un poliedru, un vârf aparţine simultan mai multor poligoane, ceea ce presupune reprezentarea repetată a acestuia şi, deci, risipă de memorie. Identificarea muchiilor comune a două poligoane, a poligonelor care au un vârf comun şi a muchiilor incidente într-un vârf este dificilă. Atunci când se face reprezentarea pe ecran, fiecare muchie este trasată de cel puţin două ori deoarece aparţine la cel puţin două poligoane distincte. Dacă se folosesc poligoane de acelaşi tip se poate renunţa la specificarea numărului de vârfuri.

(b) prin pointeri la o listă de vârfuri: datele de intrare sunt lista de vârfuri şi lista de poligoane dată implicit prin indici în lista de vârfuri. Se precizează o singură dată coordonatele fiecărui punct care intervine în model. Un poligon se defineşte prin numărul de vârfuri şi indicii corespunzători ai acestor vârfuri în lista de vârfuri. Avantajul constă în stocarea o singură dată a vârfurilor. Rămâne problema retrasării laturilor.

(c) prin pointeri la o listă de laturi: datele de intrare sunt lista de vârfuri, lista de muchii şi lista de poligoane dată implicit prin indici în lista de muchii. Lista de muchii conţine informaţii de tipul (indicele muchiei, doi indici care selectează capetele muchiei din lista de vârfuri, [numărul de poligoane la care aparţine muchia,] indicii care selectează poligoanele care conţin muchia dintr-o listă de poligoane). Lista de poligoane conţine informaţii de tipul (indicele poligonului, [numărul de muchii,] indicii care selectează muchiile poligonului din lista de muchii). De exemplu, reţeaua din figura 3.18 (a) este descrisă prin datele următoare:

$$V = \{V_1 : (x_1, y_1, z_1), V_2 : (x_2, y_2, z_2), V_3 : (x_3, y_3, z_3), V_4 : (x_4, y_4, z_4), \emptyset\},$$

$$M = \{M1 : (V_1, V_2, P_1, \emptyset), M2 : (V_2, V_3, P_2, \emptyset), M3 : (V_3, V_4, P_2, \emptyset),$$
$$M4 : (V_4, V_2, P_1, P_2, \emptyset), M5 : (V_4, V_1, P_1, \emptyset), \emptyset\},$$
$$P = \{P_1 : (M_1, M_4, M_5, \emptyset), P2 : (M_2, M_3, M_4, \emptyset), \emptyset\}.$$

Atunci când se face reprezentarea pe ecran, reţeaua se transpune prin trasarea tuturor muchiilor după lista de muchii, ceea ce duce la evitarea reprezentărilor duble.

Exemplu: Fişierul care descrie casa din figura 2.14 conţine următoarele informaţii:

/*nr.vârfuri nr.muchii nr.poligoane*/
10 15 7
/*lista coordonatelor celor 10 vârfuri*/
0 0 54; 16 0 54; 16 10 54; 8 16 54; 0 10 54; 0 0 30; 16 0 30; 16 10 30; 8 16 30; 0 10 30
/*lista celor 15 muchii descrise prin indicii vârfurilor din lista anterioară*/
0 1; 1 2; 2 3; 3 4; 4 0; 5 6; 6 7; 7 8; 8 9; 9 5; 0 5; 1 6; 2 7; 3 8; 4 9
/*lista celor 7 poligoane descrise prin indicii muchiilor din lista anterioară*/
0 1 2 3 4; 5 6 7 8 9; 0 11 5 10; 10 9 14 4; 11 6 12 1; 12 7 13 2; 13 8 14 3;

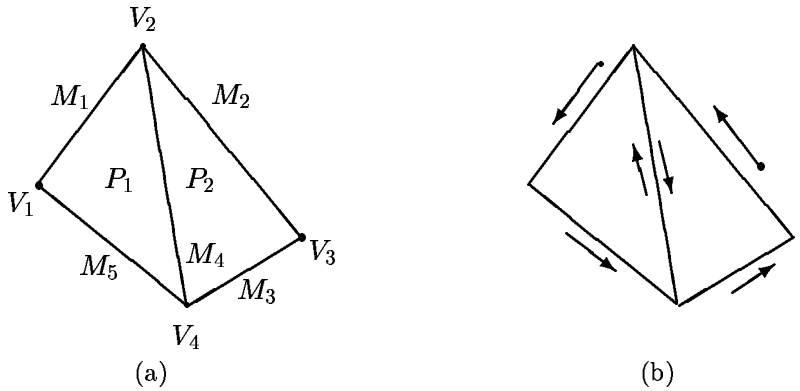


Figura 3.18: (a) Descrierea unei reţele poligonale (b) Parcurgerea în sens trigonometric a laturilor

Parcugerea faţetelor prin generare. Când se dispune de o memorie redusă şi se modelează corpuri de complexitate relativ mare printr-un număr mare de puncte este recomandată utilizarea unui liste de vârfuri şi un procedeu de parcurgere a acesteia pentru a determina rând pe rând indicii corespunzători vârfurilor fiecărei faţete. Se consideră cazul generării unui corp prin faţete triunghiulare. Pentru a genera reţeua de faţete sunt necesare:

1. o listă de vârfuri în care un punct este menţionat printr-un indice şi coordonatele sale;
2. o listă de distribuţie a punctelor pe secţiuni în care pentru fiecare secţiune transversală se menţionează numărul şi indicii punctelor care o descriu;
3. o listă de ramificaţie în care pentru fiecare secţiune a unui corp, cu excepţia ultimei secţiuni, se precizează pentru fiecare punct, în sensul creşterii indicilor, numărul de triunghiuri distincte care nu au mai fost considerate şi au un vârf în acesta. Pentru primul punct dintr-o secţiune, se numără numai triunghiurile formate cu puncte de pe secţiunea următoare, în sensul creşterii indicilor, pornind de la primul punct al secţiunii următoare.

În mod curent se utilizează reţele de faţete patrulatere plane sau triunghiulare. Cele mai bune aproximări ale unor suprafeţe strâmbe prin număr dat de vârfuri se obţin folosind faţete triunghiulare.

Pentru cazurile în care este necesar a deosebi la un poligon o faţă exterioară şi una interioară se stochează vârfurile sale într-o ordine convenită în baza de date. Ordinea normală este aceea pentru care vârfurile se stochează astfel încât laturile poligonului privite din exterior să fie aşezate în sens trigonometric. În acest fel, direcţia normalei exterioare la poligonul dat se poate obţine ca produs vectorial a două laturi consecutive neconfundate şi nedegenerate pentru orice poligon convex. Această tehnică permite evitarea trasării unei laturi date de două ori, după regula: o latură nu se trasează decât atunci când este parcursă de la un vârf de indice inferior celui final (figura 3.18.b).

C a p i t o l u l 4

P r e l u c r a r e a r e p r e z e n t ă r i l o r s i m p l e

4.1 Realism vizual

Prelucrarea reprezentărilor simple este o etapă de procesare grafică de complexitate superioră transformărilor simple ale imaginii. Scopul este mărirea cantităţii de informaţie disponibilă pe ecran pentru a permite utilizatorului să înţeleagă relaţiile 3D existente între mai multe corpuri sau între componentele unui corp.

Obiectele tridimensionale proiectate pe o suprafaţă bidimensională pierd din informaţia asociată, fapt ce crează ambiguităţi în analiza imaginilor create. Un exemplu concludent este reprezentarea prin cadru de sârmă a unui cub. Astfel, nu putem spune dacă cubul (a) din figura 4.1 reprezintă cubul solid (b) sau (c).

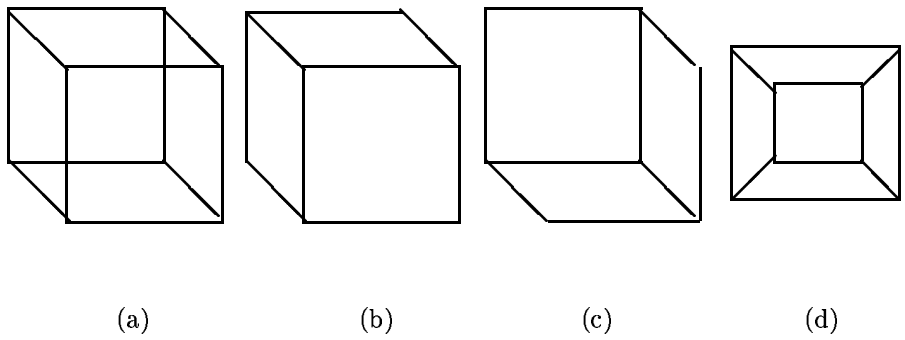


Figura 4.1: Iluzia cubului (a) Cub transparent (b) Cub privit de sus (c) Cub privit de jos (d) Cub în perspectivă

Alegerea transformării proiective este de asemenea o problemă. Cele mai simple proiecţii sunt cele paralele ortografice în care planul de proiecţie este perpendicular pe una din axele principale, însă imaginile create sunt dificil de înţeles. În proiecţia axonometrică, oblică sau perspectivă coordonata z influenţează coordonatele x şi y ale proiecţiei, producându-se ambiguităţi. De exemplu, imaginea din figura 4.1 (d) poate fi o proiecţie paralelă a unui trunchi de piramidă sau proiecţia perspectivă a unui paralelipiped dreptunghic.

Reprezentările realiste ale corpurilor 3D se pot obţine prin combinarea cel puţin a următoarelor patru metode:

1. eliminarea tuturor elementelor şi părţilor corpului care sunt mascate în anumite condiţii de privire;
2. simularea iluminării corpului cu un sistem de surse de lumină bine precizat (punctiforme, surse distribuite, lumină paralelă, lumină ambiantă);
3. reconstituirea detaliilor existente pe suprafaţă (a texturii materialului).
4. reconstituirea culorilor şi nuanţelor în care obiectul (corpul) original ar apărea, luminat în condiţiile precizate.

Aspectul texturii este purtător de informaţie cu privire la orientarea şi adâncimea unei suprafeţe. Cele 4 caracteristici principale ale texturii sunt: mărimea, forma, orientarea şi densitatea. Pe ecranele monocrome, metoda texturii poate fi folosită pentru simularea umbrelor prin densitate de puncte.

Metodele de asistare a procesului cognitiv prin mijloace grafice utilizate în sistemele CAD-CAM propun următoarele:

- utilizarea imaginii perspective, care încorporează informaţii de profunzime;
- efectul cinetic de profunzime (proiecţii dinamice): această tehnică se bazează pe mişcarea obiectului relativ la poziţia observatorului. O mişcare edificatoare este rotaţia în jurul unei axe verticale: liniile apropiate se mişcă mai rapid decât cele îndepărtate, liniile din părţi opuse ale axei de rotaţie se mişcă în sensuri contrare.
- indicii de intensitate: această metodă implică variaţia intensităţii liniilor. Liniile îndepărtate vor apărea mai şterse decât cele mai apropiate de utilizatorul sistemului;
- secţionarea în adâncime: informaţii referitoare la a 3-a dimensiune pot fi oferite prin deplasarea unui plan perpendicular pe axa Oz de-a lungul acesteia. Planul numit limitator posterior dezvăluie observatorului, pe măsura deplasării sale, obiectul, oferind astfel informaţii relative la profunzime;
- secţionarea frontală: planul de secţionare este frontal, iar partea eliminată de acesta este cea mai apropiată de observator. Metoda este mai utilă decât cea anterioară, deoarece poate furniza, printr-o poziţionare adecvată a obiectului, orice secţiune plană prin corpul respectiv;
- umbrirea: cunoscând direcţia luminii (pentru iluminarea paralelă) sau poziţia sursei (pentru iluminarea cu o sursă punctiformă), umbrele pot furniza informaţii utile pentru sugerarea poziţiei în spaţiu.
- variaţia de culoare: de-a lungul axei Oz se poate introduce o variaţie spectrală de culoare, de exemplu, de la roşu (punctele apropiate) la violet

(punctele îndepărtate), care să furnizeze informaţii cu privire la relaţiile de profunzime între elementele corpului.

4.2 Determinarea liniilor şi suprafeţelor vizibile

Date fiind un set de obiecte 3D şi o specificare a vizualizării, se cere determinarea liniilor şi suprafeţelor obiectelor care sunt vizibile, fie din centrul de proiecţie pentru proiecţiile în perspectivă, fie de-a lungul direcţiei de proiecţie în cazul proiecţiilor paralele.

Se pot imagina următoarele două tipuri de coduri pentru rezolvarea problemei:

- pentru fiecare obiect din scenă execută următoarele:
  (a) determină acele părţi ale obiectului care nu sunt acoperite de părţi ale altui obiect;
  (b) construieşte imaginea acestor părţi;
- pentru fiecare pixel al imaginii execută următoarele:
  (a) determină cel mai apropiat obiect de observator care este proiectat pe pixel;
  (b) aprinde corespunzător pixelul.

Există, corespunzător acestor modalităţi de abordare a problemei, două tipuri de algoritmi:

1. algoritmi spaţiu-obiect (precizie obiect): depind de precizia cu care obiectele sunt definite şi determină vizibilitatea fiecărui obiect;
2. algoritmi spaţiu-imagine (precizie imagine): depind de rezoluţia mediului de afişare şi determină vizibilitatea pe pixel.

Pentru n obiecte şi p pixeli efortul de calcul este proporţional, în primul caz, cu n^2 şi, în al doilea caz, cu np. Deşi în majoritatea situaţiilor $n^2 \ll np$, algoritmii spaţiu-imagine sunt mai rapizi deoarece subrutinele de comparare individuală sunt mult mai simple decât în cazul algoritmilor spaţiu-obiect şi deci consumă mai puţin timp.

Algoritmii spaţiu-obiect oferă o acurateţe deosebită a imaginii. Algoritmii spaţiu-imagine oferă rapiditate în construcţia imaginii. Calculul în spaţiu-obiect se realizează cu precizie arbitrară. Imaginea finală va fi corectă chiar şi mărită de mai multe ori. Soluţiile în spaţiu-imagine sunt calculate cu o rezoluţie mai mică, de obicei cea a dispozitivului de afişare.

4.2.1 Simplificarea calculelor

Poziţia observatorului

Se consideră că planul de proiecţie este xOy al sistemului în care este definit obiectul, iar

- în cazul proiecţiei perspective, observatorul se află pe axa z la o cotă pozitivă,

- în cazul proiecţiei paralele, direcţia de proiecţie este paralelă cu axa z şi în sens invers acesteia.

Dacă condiţiile de vizualizare nu respectă aceste reguli, se impune o fază de preprocesare în care obiectul este rotit şi translatat pentru a se încadra în condiţiile standard.

Teste de adâncime

Se pune problema dacă două puncte se acoperă unul pe celălalt în proiecţiile pe planul xOy.

În cazul proiecţiei paralele, două puncte $A(x_1, y_1, z_1)$, $B(x_2, y_2, z_2)$ sunt pe aceeaşi linie de proiecţie dacă şi numai dacă $x_1 = x_2$, $y_1 = y_2$.

În cazul proiecţiei perspective, pentru simplificarea calculelor, comparaţiile individuale privind relaţiile de profunzime se fac după aplicarea transformării care duce centrul de proiecţie în origine. Relaţiile de verificat pentru ca A şi B să fie pe aceeaşi linie de proiecţie sunt: $x_1/z_1 = x_2/z_2$, $y_1/z_1 = y_2/z_2$. Pentru fiecare comparare a două puncte trebuie efectuate patru împărţiri. O soluţie pentru evitarea acestor împărţiri este aplicarea unei transformări 3D obiectului real astfel încât obiectul transformat obţinut să aibă o proiecţie paralelă identică cu proiecţia perspectivă a obiectului real. Această transformare mută centrul de proiecţie la $-\infty$ pe axa z şi transpune liniile de proiecţie în linii paralele cu axa z, deci paralele între ele (vezi capitolul 2).

Teste de interioritate

Se pune problema interiorităţii unui punct faţă de un contur poligonal. Fie $F = (P_1, P_2, \ldots, P_n)$ un poligon în planul de proiecţie cu vârfurile $P_i(x_i, y_i)$, $i = 1, \ldots, n$ şi $P_n = P_1$. Fie $P(x, y)$ punctul pentru care trebuie să se efectueze testul de interioritate. În geometria elementară există două proceduri bine cunoscute pentru realizarea testului de interioritate a unui punct faţă de un contur poligonal în planul de proiecţie:

1. calcularea sumei unghiurilor. Se notează cu α_i măsura unghiului făcut de segmentele PP_i şi PP_{i+1}, $(i = 1, \ldots, n-1)$. Atunci, P este în exteriorul conturului poligonal dacă $\sum_{i=1}^{n-1} \alpha_i = 0$ şi este în interiorul conturului dacă $\sum_{i=1}^{n-1} \alpha_i = 2\pi$ (figura 4.2.a);
2. calcularea numărului de intersecţii (testul par-impar). Se consideră o semidreaptă, care porneşte din punctul P şi nu trece prin niciunul din vârfurile poligonului. Punctul este exterior poligonului dacă numărul de intersecţii ale semidreptei cu laturile lui F este par şi este interior lui F dacă acest număr este impar (figura 4.2.b).

Teste minimax

Testele minimax sunt utile în diverse momente ale algoritmilor: la calculul suprapunerii a două poligoane în planul de proiecţie, la calculul interiorităţii

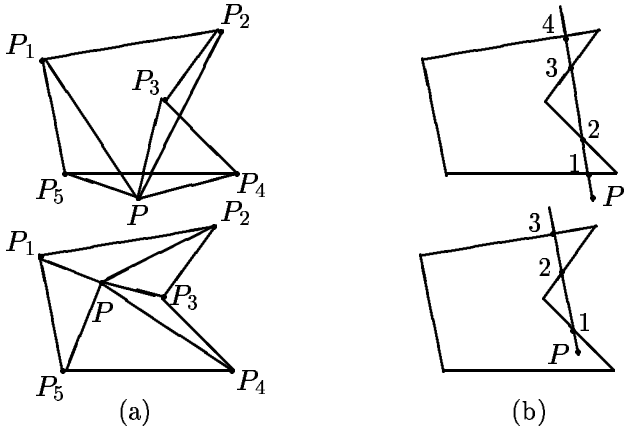


Figura 4.2: Teste de interioritate prin (a) suma unghiurilor (b) număr de intersecţii

proiecţiei unui punct faţă de proiecţia unui poligon, la calculul intersecţiei a două segmente, la calculul de adâncime a unui punct faţă de un poligon sau chiar a două poligoane între ele în spaţiul obiect.

Aceste teste presupun doar comparaţii. Ele pot duce la respingerea a numeroase cazuri nefavorabile, dacă sunt utilizate înaintea calculelor propriu-zise. De exemplu, două poligoane din planul de proiecţie nu au nici un punct de intersecţie dacă valoarea minimă a proiecţiei pe axa x a celui de al doilea este mai mare decât valoarea maximă a proiecţiei pe aceeaşi axă a primului.

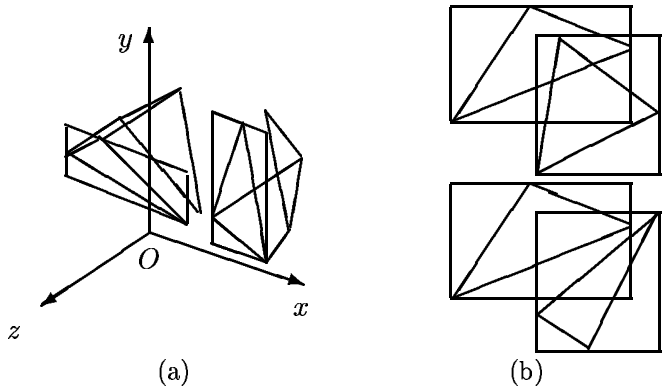


Figura 4.3: (a) Extinderile plane ale poligoanelor (b) Suprapunerea extinderilor plane

Se numeşte extindere plană (extindere ecran) a unui poligon dat din 3D, dreptunghiul de arie minimă circumscris proiecţiei poligonului şi care are o

latură orizontală (figura 4.3.a). Pentru determinarea extinderii ecran se calculează maximul şi minimul coordonatelor proiecţiilor pe ecran ale vârfurilor poligonului (patru valori ce definesc colţurile dreptunghiului).

Două poligoane ale căror extinderi ecran sunt disjuncte (nu se suprapun) nu se pot acoperi unul pe altul. Dacă extinderile se suprapun, cu privire la suprapunerea poligoanelor nu se poate afirma nimic. În acest caz există două situaţii (figura 4.3.b):

(a) atât extinderile, cât şi proiecţiile poligoanelor, se suprapun;

(b) extinderile se suprapun, dar proiecţiile poligoanelor sunt disjuncte.

Analog se pot defini extinderile unidimensionale şi spaţiale.

O extindere unidimensională se defineşte, relativ la un plan de coordonate, ca fiind zona bandă de lăţime minimă în care se înscrie proiecţia poligonului în acel plan, situată între două drepte paralele şi paralele cu una din axele de coordonate ale planului. Cazuri particulare sunt extinderile după x, y, z ale poligoanelor (figura 4.4.a).

Când extinderile după z a două poligoane nu se suprapun, atunci poligonul cu extinderea după z mai apropiată de observator va acoperi celălalt poligon (cel puţin parţial) dacă extinderile ecran se intersectează. Determinarea extinderii după z se face căutând maximul şi minimul valorilor lui z pentru vârfurile poligonului considerat.

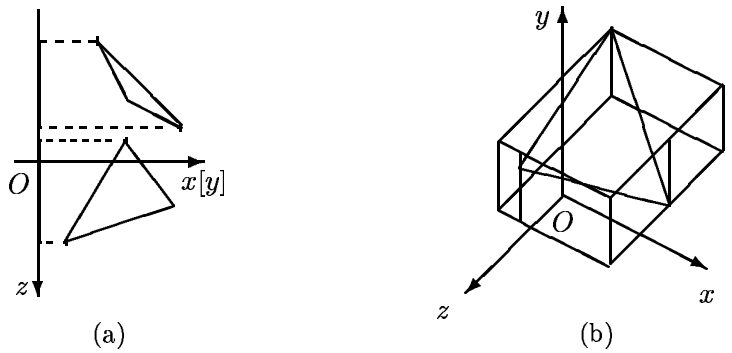


Figura 4.4: (a) Extinderea unidimensională după z (b) Extinderea spaţială

Extinderile spaţiale ale poligoanelor sunt paralelipipedele de volum minim în care se înscriu poligoanele respective şi care au feţele paralele cu planele triedrului de referinţă (figura 4.4.b).

Înainte de a compara două poligoane, se compară extinderile acestora. Se elimină astfel o bună parte din calculele de intersecţii, simplificând şi accelerând tratarea imaginii.

4.2.2 Vizualizarea suprafeţelor descrise explicit

Se consideră cazul special al unui suprafeţe descrise explicit printr-o ecuaţie de tipul

$$y = f(x, z).$$

Prin discretizarea funcţiei f pe o grilă regulată în x şi z se obţine o matrice de valori y reprezentând înălţimi faţă de grila în x şi z aflată în planul $y = 0$. Indicii şi valoarea elementului matriceal specifică în mod unic un punct din spaţiu.

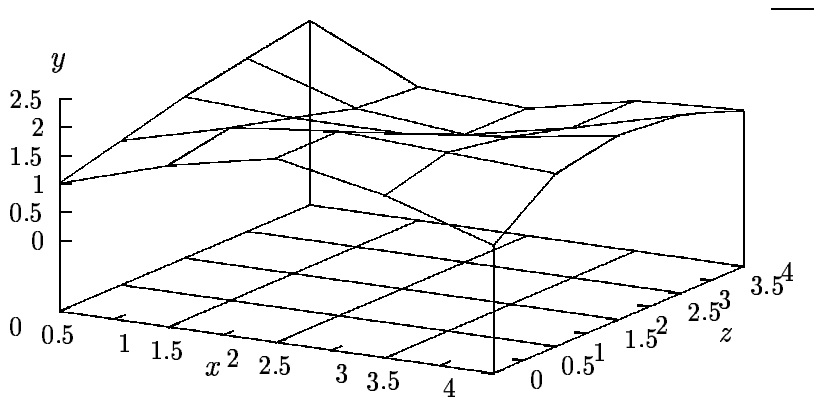


Figura 4.5: Reprezentarea prin cadru de sârmă a unei suprafeţe

O trasare tip cadru de sârmă poate fi construită prin aproximarea liniară (figura 4.5). Se trasează un set de linii poligonale ce trec prin punctele definite de fiecare linie a matricii (linii poligonale de z constant) şi un set de linii poligonale ortogonale care trec prin punctele definite de fiecare coloană (linii poligonale cu x constant).

Se consideră în primul rând problema trasării liniilor poligonale de z constant. Presupunem că cea mai apropiată linie poligonală este o latură a suprafeţei. Liniile poligonale ce urmează a fi trasate se află în plane paralele de z constant. Se trasează liniile în ordinea crescătoare a distanţei de la observator.

Se consideră liniile poligonale din planele cu z constant din figura 4.6 (a). Dacă se trasează o nouă linie poligonală, ea este vizibilă numai acolo unde proiecţia sa se ridică deasupra celei mai înalte sau se află sub cea mai joasă

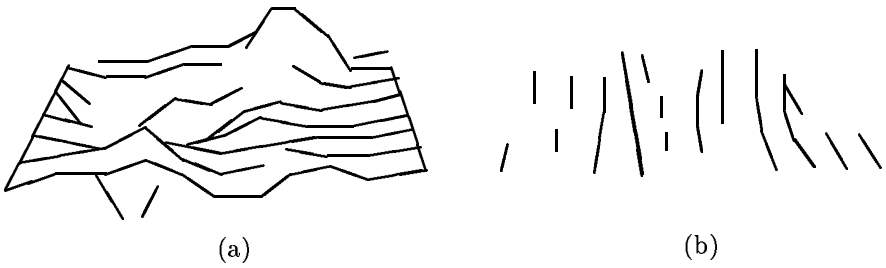


Figura 4.6: Trasarea cadrului de sârmă a unei suprafeţe (a) Linii poligonale cu z constant (b) Linii poligonale cu x constant

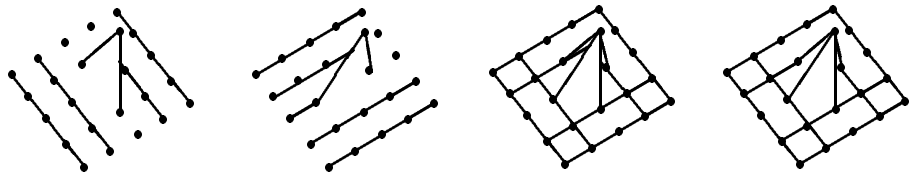
siluetă. Deoarece noua linie poligonală se află într-un plan mai îndepărtat decât precedentele, nu poate acoperi liniile poligonale anterioare. De aceea, pentru a determina care părţi ale liniei poligonale vor fi trasate, se compară valorile y ale liniei dorite cu valorile y ale siluetelor (valorile y ale proiecţiilor).

Algoritmul bazat pe aceste siluete (valorile y minime şi maxime corespunzătoare x din grilă) se numeşte algoritmul liniei orizontului (algoritmul Wright). Din punct de vedere al clasificării algoritmilor de determinare a liniilor vizibile, acesta este un exemplu de algoritm spaţiu-imagine.

Pentru reprezentarea siluetelor se utilizează două tablouri unidimensionale, Y_{min} şi Y_{max}, care conţin valorile minime şi maxime curente ale valorilor proiectate y pentru o mulţime finită de valori x (valorile sunt actualizate după calcularea valorilor y ale fiecărei noi linii).

Dacă se trasează liniile poligonale cu x constant, linia poligonală cu x constant cea mai apropiată de observator (îngroşată în figura 4.6.b) nu formează o latură a suprafeţei. Pentru a trasa suprafaţa corect, se trasează liniile poligonale de la dreapta celei mai apropiate în ordinea de la stânga la dreapta, iar cele la stânga celei mai apropiate, de la dreapta la stânga. În ambele cazuri, trasarea liniei poligonale se face în ordinea crescătoare a distanţei de la observator.

Suprapunerea liniilor poligonale cu x constant şi z constant nu permite întotdeauna o reprezentare corectă a suprafeţei. Un exemplu concludent este prezentat în figura 4.7. Soluţia corectă presupune trasarea concomitentă a liniilor poligonale. Setul de linii poligonale care sunt aproape paralele cu planul de vizualizare (în figura dată se consideră cele de z constant) sunt procesate în ordinea descrisă mai sus. După ce o linie poligonală de z constant este trasată, se construiesc segmentele de linie poligonală cu x constant dintre linia poligonală nou trasată şi următoarea linie poligonală. Segmentele de linie cu x constant se trasează utilizând aceeaşi structură de date-siluete care a fost utilizată pentru trasarea liniei poligonale de z constant. Procesarea are loc în ordinea crescătoare a distanţei faţă de observator.



(a) (b) (c) (d)

Figura 4.7: Suprapunerea reprezentările wire-frame (a) Linii cu z constant (b) Linii cu x constant (c) Suprapunere (a) Soluţia corectă

4.2.3 Algoritmi spaţiu-obiect pentru determinarea liniilor vizibile

Algoritmul reprezentării incomplete

Se consideră o reprezentare poliedrală a unui corp convex şi o mulţime de puncte care definesc această reprezentare (mulţimea include cel puţin vârfurile faţetelor poligonale). Imaginea se reconstituie pe principiul: un segment este vizibil atunci când capetele sale sunt vizibile. Se alege ca obiectiv un anumit punct de pe suprafaţa corpului şi se depistează eventualele obstacole ce se interpun între punct şi observator.

Pentru determinarea vizibilităţii punctelor reprezentării poliedrale se parcurg următoarele etape:

(a) se iniţializează cu valoarea zero un vector de dimensiune egală cu numărul de puncte prin care este reprezentat corpul. Acest vector este numit zona atributelor de vizibilitate. Simbolul 0 marchează punctele vizibile. Valoarea 1 indică un punct invizibil.

(b) pentru fiecare faţă poligonală se parcurg toate punctele reprezentării căutând acelea a căror proiecţie pe ecran se află în interiorul sau pe frontiera proiecţiei faţetei, numite puncte obiectiv. Pentru fiecare punct obiectiv,

   1. se calculează intersecţia dintre dreapta de la punctul obiectiv la observator şi suprafaţa poligonală curentă, rezultând un punct numit punct mască potenţial;
   2. se compară distanţa de la observator la punctul obiectiv cu cea de la observator la punctul mască potenţial. Dacă prima este mai mare atunci se modifică atributul corespunzător punctului din 0 în 1 (punctul este invizibil).

În final zona atributelor de vizibilitate stochează informaţii complete cu privire la vizibilitatea punctelor prin care corpul este aproximat.

În figura 4.8 (b) se prezintă rezultatul aplicării algoritmului la paralelipipedul (a).

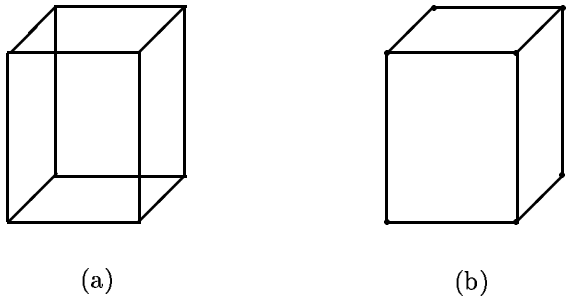


Figura 4.8: (a) Reprezentarea poliedrală a unui paralelipiped (b) Imagine prin algoritmul reprezentării incomplete

Algoritmul spaţiu-obiect incomplet, ce reprezintă numai curbele sau segmentele cu ambele capete vizibile, se poate aplica cu succes dacă corpul descris este convex, are un număr mic de faţete sau dacă imaginea ocupă o zonă mare pe ecran. Pentru corpurile concave, problema se complică (figura 4.9).

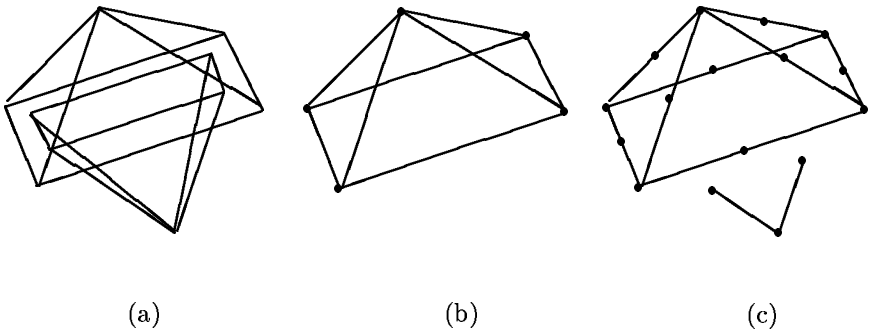


Figura 4.9: (a) Reprezentarea poliedrală a unui corp concav (b) Imagine prin algoritmul reprezentării incomplete (c) Imagine îmbunătăţită prin creşterea numărului de puncte ce descriu suprafaţa

Algoritmul invizibilităţii cantitative

Algoritmul invizibilităţii cantitative (algoritmul Appel) operează asupra solidelor formate din feţe plane mărginite de poligoane. Metoda folosită este aceea de a testa vizibilitatea segmentelor din care sunt alcătuite feţele solidului.

Invizibilitatea cantitativă a unui punct al unei linii este numărul de poligoane care maschează respectivul punct. Când o linie trece în spatele unui poligon, invizibilitatea cantitativă a punctelor sale este incrementată cu 1; când iese, este

decrementată cu 1. O linie este vizibilă numai dacă invizibilitatea punctelor sale este 0 (figura 1).

Se presupune că poligoanele din reprezentare nu se întrepătrund. Atunci o linie îşi schimbă invizibilitatea cantitativă dacă trece prin spatele unei linii de contur.

O linie de contur este o muchie comună unei feţe de spate şi a uneia frontale. Se presupune că toate poligoanele au normala la suprafaţă orientată spre exteriorul corpului. Un poligon de spate este identificat printr-un produs scalar pozitiv între normala la suprafaţă şi vectorul ce porneşte din centrul proiecţiei la un punct oarecare al poligonului. În figura 4.10 (a) AB, CD, DF, KL sunt linii de contur, iar CE, EF, JK nu sunt linii de contur.

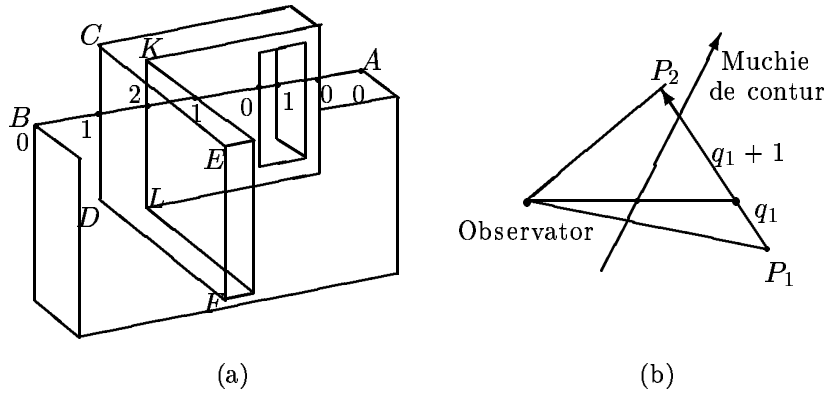


Figura 4.10: (a) Definirea liniilor de contur în algoritmul Appel (b) Schimbarea invizibilităţii cantitative

O linie de contur trece prin faţa unei muchii dacă străpunge triunghiul format de capetele muchiei şi ochiul observatorului (figura 4.10.b). Se aplică testul punct interior unui triunghi.

Într-un prim pas al algoritmului, se elimină muchiile care aparţin unor feţe total invizibile (poligoane de spate). În pasul al doilea se determină liniile de contur. Apoi se determină invizibilitatea cantitativă a vârfurilor.

Pentru determinarea vizibilităţii vârfurilor se consideră un vârf iniţial care se proiectează pe toate feţele poligonale ale reprezentării. Se reţin doar proiecţiile interioare feţelor. Numărul proiecţiilor mai apropiate de observator decât vârful studiat indică invizibilitatea cantitativă a acestuia. Această valoare se propagă de-a lungul unei muchii, fiind modificată la trecerea prin spatele unei linii de contur. Când se atinge capătul final al unei muchii, invizibilitatea cantitativă asociată (emanată prin muchie) devine invizibilitatea cantitativă a tuturor liniilor care pornesc din respectivul punct.

În propagarea invizibilităţii cantitative apar probleme la vârfurile liniilor de contur care sunt comune mai multor feţe frontale (de exemplu JK are invizibilitatea cantitativă 0, iar KL, 1). Această schimbare poate rezulta dintr-un test

suplimentar al muchiei faţă de laturile care împart vârful.

Aplicaţie. Se consideră o reprezentare poliedrală prin triunghiuri. Mulţimea liniilor de contur (conturul aparent) este mulţimea segmentelor care au ambele capete vizibile şi sunt laturi ale unor faţete triunghiulare cu două vârfuri vizibile şi unul invizibil. Segmentele din conturul aparent se pot depista uşor după parcurgerea algoritmului spaţiu-obiect incomplet pe baza consultării zonei atributelor de vizibilitate. Se observă că, în cazul când ambele capete ale unui segment oarecare au acelaşi atribut de vizibilitate, numărul intersecţiilor cu conturul aparent este par, iar atunci când atributele sunt diferite, impar. Fiecare faţă este descrisă prin coordonatele celor trei vârfuri şi proiecţiile lor. Pentru fiecare faţetă se parcurg laturile acesteia şi pentru fiecare latură, numită latură curentă se parcurge din nou corpul, faţetă cu faţetă. Pentru fiecare faţetă parcursă, dacă are două vârfuri vizibile şi unul invizibil, atunci latura faţetei cu două capete vizibile face parte din conturul aparent şi este numită segment curent. Se proiectează latura curentă şi segmentul curent pe planul de proiecţie. Se calculează intersecţia dintre dreapta suport a laturii curente proiectate şi dreapta suport a segmentului curent proiectat. Dacă intersecţia se află atât între extremităţile laturii curente proiectate cât şi ale segmentului curent proiectat, atunci se stochează coordonatele intersecţiei. Se ordonează intersecţiile obţinute, pornind de la un capăt (ales arbitrar ca origine a laturii curente) către celălalt. Se obţin astfel anumite intervale pe latura curentă. Se consideră intervalul dintre prima extremitate a laturii curente şi prima intersecţie. Pe acest interval latura se consideră vizibilă dacă extremitatea laturii este vizibilă şi invizibilă în caz contrar. Această informaţie se reprezintă printr-o variabilă numită atribut de vizibilitate al intervalului (care are două valori, 0 şi 1, la fel ca atributul punctului). Până la epuizarea intervalelor, se trasează sau nu intervalul după cum precizează atributul de vizibilitate al intervalului, apoi se complementează atributul de vizibilitate faţă de 1.

4.2.4 Algoritmi spaţiu-imagine

Algoritmul tamponului de adâncime

Algoritmul z-buffer sau al tamponului de adâncime (algoritmul Catmull) este destinat eliminării suprafeţelor ascunse şi este unul dintre cei mai simplii algoritmi de vizibilitate. Oferă avantajul de a putea trata suprafeţe strâmbe (cuadrice, suprafeţe parametrice bicubice). Dezavantajul constă în necesarul de memorie pentru funcţionarea eficientă. Pentru o rezoluţie de 512×512 şi o reprezentare a valorilor z pe 4 octeţi, necesarul de memorie depăşeşte 1Mo. Numărul de elemente ale matricii-tampon (z-buffer-ului) este acelaşi cu numărul de intrări ale zonei tampon cadru (a frame-buffer-ului).

Procesul implicat este căutarea unui maxim pentru fiecare submulţime a unei mulţimi date. Mulţimea dată este cea a tuturor suprafeţelor elementare a căror reuniune descrie suprafaţa corpului de observat. Submulţimile sunt constituite din suprafeţele elementare care au aceeaşi imagine pe ecran. Z-buffer-ul serveşte la stocarea valorilor maxime temporar depistate pentru o

submulţime, pentru ca în final să conţină maximele căutate.

Generarea z-buffer-ului se poate realiza odată cu conversia scan a obiectului. În timpul baleierii, dacă punctul primitivei care este convertită este mai apropiat de observator decât punctul a cărui culoare şi adâncime sunt momentan înscrise în tampoane, atunci culoarea şi adâncimea noului punct sunt înscrise peste vechile valori în frame-buffer, respectiv z-buffer. Astfel z-buffer-ul şi frame-buffer-ul înregistrează informaţia asociată cu cel mai mare z întâlnit la un moment dat pentru fiecare (x, y).

Etapele algoritmului z-buffer-ecran pentru o reprezentare poliedrală sunt următoarele:

(a) se creează o matrice, fiecărui pixel corespunzându-i un element din matrice, numită z-buffer-ecran; se iniţializează z-buffer-ul la o valoare foarte mică pe care suntem siguri că z-tul punctelor corpului nu o pot atinge şi se iniţializează ecranul la valoarea de intensitate corespunzătoare fondului.

(b) pentru fiecare faţetă poligonală se transpune poligonul astfel: pentru fiecare punct-pixel de coordonate (q, w) din interiorul sau de pe frontiera proiecţiei poligonului:

1. se reţine cota z a punctului care are pe ecran proiecţia în pixelul (q, w);
2. se compară z cu $z(q, w)$, valoarea existentă în z-buffer la linia w, coloana q. Dacă este mai mare (punctul este mai apropiat de observator) scrie z în z-buffer la linia w şi coloana q şi aprinde pixelul (q, w) conform informaţiei de culoare sau umplere a poligonului curent.

Algoritmul liniei de baleiaj

Algoritmul scan-line operează în spaţiul-imagine creând imaginea linie cu linie. Diferenţa faţă de algoritmul de conversie scan constă în faptul că se tratează o mulţime de primitive, nu numai una singură.

Se consideră o reprezentare poliedrală. Se ţine o gestiune a intersecţiilor şi a proiecţiilor muchiilor şi poligoanelor pe xOy prin:

1. o listă a muchiilor ET (edge table). Intrările în tabel sunt sortate în grupuri pe baza coordonatei y cea mai mică a laturii şi, intrările unui grup, pe baza coordonatelor x ale punctului de căpăt cu y minim, iar intrările cu acelaşi punct de minim, în ordinea crescătoare a x-ilor corespunzător celuilalt capăt (cu y maxim). Fiecare intrare conţine:
   (a) coordonata x a extremităţii cu cel mai mic y,
   (b) coordonata y a celuilalt capăt (maximă),
   (c) incrementul în x, inversul pantei muchiei, utilizat în saltul de la o linie de baleiaj la alta,
   (d) codul de identificare al poligonului care conţine muchia (o muchie apare de mai multe ori în listă, corespunzător fiecărui poligon);
2. o listă a poligoanelor PT (poligon table). Intrarea corespunzătoare unui poligon conţine informaţiile:

(a) codul de identificare,
(b) coeficienţii ecuaţiei planului suport,
(c) informaţii privind culoarea sau umplerea poligonului,
(d) o variabilă de stare, iniţializată de-a lungul procesului de baleiaj, care indică poziţia poligonului relativă la linia de baleiaj (aceasta se află în interior sau exterior poligon);

3. o listă a muchiilor active AET (active edge table) ţine evidenţa muchiilor intersectate de linia de baleiaj în ordinea crescătoare a x-ilor corespunzători intersecţiilor.

Primul pas al algoritmului constă în crearea primelor două liste. Apoi se baleiază ecranul linie cu linie. Pentru fiecare linie, se crează lista muchiilor active şi se ordonează intersecţiile în aceasta. Se parcurge pixel cu pixel linia curentă. Pentru fiecare pixel se compară abscisa curentă cu intersecţiile din al treilea tabel. Dacă abscisa corespunde unei intersecţii, atunci:

1. se identifică poligonul, se modifică variabila de stare prin complementare (presupunem 1=în intersecţie, 0=afară);
2. se testează variabilele de stare:
   (a) dacă o singură variabilă are valoarea 1, linia de baleiaj taie în acel punct proiecţia unui singur poligon şi valorile corespunzătoare umplerii poligonului se trec în frame-buffer;
   (b) altfel, se calculează pentru fiecare poligon cota z din ecuaţia planului (din PT). Se aprinde pixelul conform informaţiilor corespunzătoare poligonului celui mai apropiat (z-ul cel mai mare în ipoteza observatorului la z pozitiv).

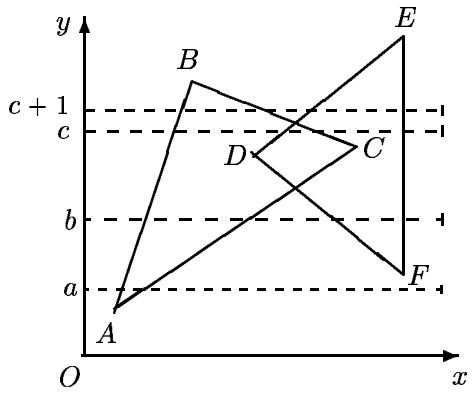


Figura 4.11: Algoritmul scan-line

Se consideră cazul celor două poligoane din figura 4.11, pentru care

$$\text{ET} = (AB, AC, FD, FE, DE, CB),$$

$$\text{PT} = (ABC, DEF).$$

Lista muchiilor active variază astfel:

$$\text{AET} = \begin{cases} (AB, AC), & y = a, \\ (AB, AC, FD, FE), & y = b, \\ (AB, DE, CB, FE), & y = c, \\ (AB, CB, DE, FE), & y = c + 1. \end{cases}$$

Când linia de baleiaj cu $y = a$ intersectează AB, variabila de stare a poligonului ABC se schimbă. Fiind o unică intersecţie, intersecţia dintre poligon şi linia de baleiaj este vizibilă. La intersecţia cu AC, variabila de stare este complementată. Înainte de trecerea la noua linie de baleiaj se actualizează AET. La $y = b$ există două poligoane care se intersectează cu linia de baleiaj, dar la un moment dat doar un poligon este vizibil. La $y = c$ şi intersecţia cu DE, două poligoane au variabila de stare setată. Se evaluează z din ecuaţia planelor pentru $y = c$ şi x corespunzător intersecţiei. În cazul dat, z-ul poligonului DEF este mai mare, deci punctul este vizibil şi determină caracteristicile pixelului. Dacă poligoanele se întrepătrund, algoritmul este modificat pentru a determina punctele de penetraţie în linia de baleiaj (liste noi ET, PT şi AET).

Algoritmi de subdivizare a ariilor în arii elementare

Se consideră o reprezentare poliedrală. Se presupune că poligoanele nu se întrepătrund. O arie elementară este o zonă a imaginii în care se poate determina uşor care poligon dintr-o mulţime dată este vizibil.

Algoritmii bazaţi pe subdivizarea ariilor utilizează o strategie divide-et-impera. Prin împărţirea imaginii în zone de arie din ce în ce mai mică se caută obţinerea de zone în care să se situeze cel mult o muchie a unui poligon din reţea. Divizarea se opreşte cel târziu în stadiul în care elementul de arie devine un pixel.

În algoritmul Warnock fiecare arie este divizată în patru arii de mărimi egale. În fiecare etapă a procesului recursiv, proiecţia fiecărui poligon se află, relativ la aria de interes, în una din următoarele situaţii (figura 4.12.a):

- conţine aria de interes;
- intersectează aria;
- este inclus în arie;
- este disjunct ariei.

În procesul recursiv, aria de interes este tratată astfel:

1. dacă aria nu conţine nici un poligon, atunci este afişată culoarea de fond;
2. dacă există un singur poligon cu care se intersectează sau care este conţinut în arie, atunci aria este colorată iniţial cu culoarea fondului şi apoi partea poligonului conţinută în arie este convertită prin baleiere;
3. dacă există un singur poligon înconjurător, dar nu şi poligoane intersectate sau poligoane conţinute, atunci aria este umplută cu culoarea poligonului;
4. cazul contrar este cel al mai multor poligoane intersectate, conţinute sau înconjurătoare. Algoritmul impune divizarea ariei până când există un poligon înconjurător care se află în faţa celorlalte poligoane, întreaga arie fiind umplută cu culoarea acestuia.

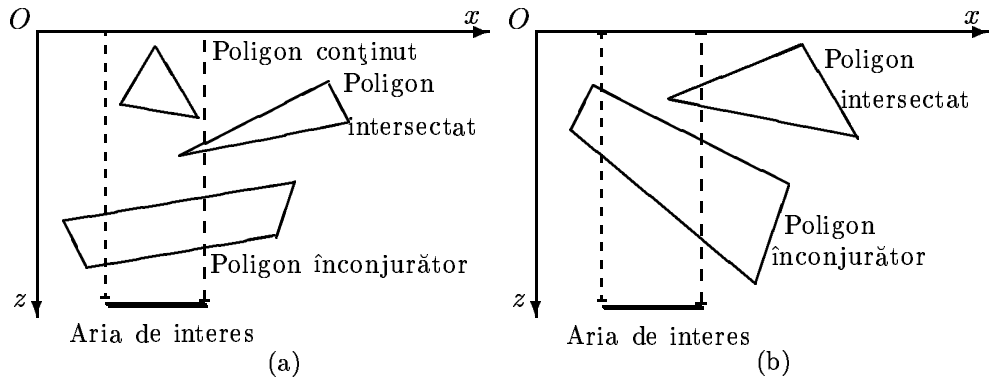


Figura 4.12: Două situaţii de înconjurare a ariei de interes în algoritmul Warnock

Testul de poligon frontal (cazul 4) se realizează calculând coordonatele z ale punctelor din planurile de sprijin ale poligoanelor în vârfurile (x, y) ale ariei. Dacă poligonul înconjurător are valorile z cele mai mari, se consideră poligon aflat în faţa celorlalte (figura 4.12.a). În situaţia din figura 4.12 (b), deşi poligonul intersectat se află de partea opusă planului poligonului înconjurător, algoritmul cere divizarea ariei până când poligonul înconjurător are valorile z cele mai mari sau se întâlnesc cazurile 1, 2 sau 3.

În figura 4.13 se consideră un exemplu de subdivizare (triunghiul se află în faţa dreptunghiului). Numerele înscrise în pătrate se referă la cazurile mai sus menţionate.

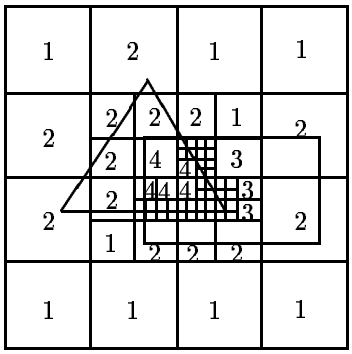


Figura 4.13: Exemplu pentru algoritmul Warnock

Algoritmul operează în spaţiul obiect, cu excepţia conversiei scan. Cea mai mică arie este pixelul. La o rezoluţie de 1024×1024 sunt necesare cel mult zece

nivele de subdivizare.

În algoritmul Weiler-Atherton subdivizarea ecranului este realizată de-a lungul frontierei poligoanelor (în locul dreptunghiurilor din algoritmul Warnock). Algoritmul este asociat cu o metodă de clipping. Primul pas constă în sortarea poligoanelor funcţie de valoarea z cea mai mare. Poligonul cel mai apropiat de observator este utilizat în procesul de decupare al celorlalte poligoane, rezultând două liste conţinând părţi ale acestora situate în interiorul, respectiv exteriorul proiecţiei primului poligon. Poligoanele din lista de interior se află, teoretic, în spatele poligonului şi, deci, trebuie şterse, fiind invizibile. Dacă un poligon din lista de interior este mai apropiat decât poligonul de decupare, sortarea iniţială nu oferă o ordine corectă a priorităţilor. Un asemenea poligon frontal este procesat recursiv pentru decupare faţă de părţile aflate în lista de interior. Când divizarea recursivă este terminată, algoritmul continuă procesarea poligoanelor din lista de exterior. Decuparea este realizată cu o copie a poligonului iniţial şi nu cu fragmentele acestuia, procesul de decupare fiind mai puţin costisitor. Algoritmul utilizează o stivă pentru a trata cazurile de suprapunere, ca cel din figura 4.14. Stiva conţine o listă a poligoanelor care sunt poligoane curente de decupare, dar a căror utilizare a fost întreruptă datorită subdivizării recursive. Dacă un poligon din lista de interior se află în faţa poligonului de decupare, el este căutat în stivă. Dacă este găsit în stivă, nu mai sunt necesare alte recursii, deoarece toate bucăţile de poligon din interiorul poligonului aflate în spate au fost deja eliminate.

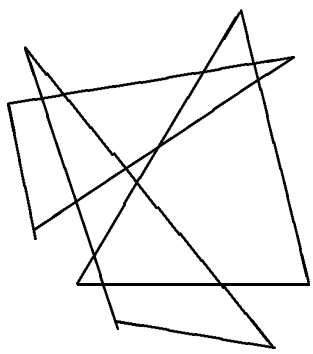


Figura 4.14: Suprapunerea a trei poligoane ce conduce la un ciclu infinit în algoritmul Weiler-Atherton fără stivă

Algoritmul Encarnacao al reţelei de explorare se aplică la suprafeţele curbe care sunt definite ca reţele de linii. Fragmentele (careurile) suprafeţelor curbe sunt specificate numai prin cele 4 vârfuri ale lor, iar liniile drepte sunt folosite pentru a defini muchiile lor. Numele algoritmului derivă din faptul că o reţea rectangulară bidimensională de linii, numită reţea de explorare, este suprapusă pe planul-imagine al suprafeţelor. Se constituie o listă de fragmente de suprafaţă pentru fiecare arie a reţelei de explorare. Dacă în careu se află

prea multe elemente de suprafaţă se descompune careul de rastru în patru subcareuri. Testele mimimax sunt folosite pentru a stabili care arii de explorare vor fi suprapuse de un fragment specific de suprafaţă. Imediat ce s-a terminat preselectarea (constituirea listei), se execută algoritmul principal de vizibilitate. Un segment de linie este rupt într-un şir de puncte de test, spaţiate în mod egal între vârfuri. Fiecare punct este testat pentru vizibilitate faţă de fragmentele de suprafaţă care se află în interiorul aceleiaşi arii de explorare cu punctul de testare. Ori nici un fragment din suprafaţă nu acoperă punctul de testare, în care caz punctul de test este vizibil, ori cel puţin un fragment acoperă punctul de testare, în care caz este invizibil (se compară coordonata z a punctului de test cu coordonatele z ale planurilor asociate fiecărui fragment de suprafaţă din lista asociată ariei curente de explorare). Dacă toate punctele care se află de-a lungul muchiei fragmentului sunt vizibile, întreaga muchie se trasează. Dacă se constată că unele puncte sunt invizibile, sunt trasate numai acele secţiuni ale muchiei fragmentului care se află între punctele vizibile de test.

Algoritmul drumului optic

Algoritmul ray-tracing determină vizibilitatea suprafeţelor trasând o rază imaginară de lumină de la ochiul observatorului la obiectele din scenă. Sunt date un centru al proiecţiei (ochiul observatorului) şi o fereastră într-un plan arbitrar.

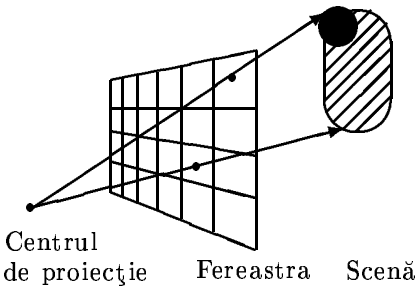


Figura 4.15: Algoritmul drumului optic

Fereastra este divizată într-o grilă dreptunghiulară ale cărei elemente corespund pixelilor din imaginea finală. Pentru fiecare element al grilei din fereastră, raza vizuală este trasată de la centrul proiecţiei prin centrul elementului spre scenă (figura 4.15). Pixelul corespunzător elementului se aprinde conform informaţiilor obiectului care este intersectat primul (prototipul algoritmilor spaţiu-imagine). Raza este definită parametric. Pentru fiecare obiect este necesară o reprezentare care permite determinarea cu uşurinţă a intersecţiei cu raza.

Dacă algoritmul z-buffer calculează informaţia numai pentru acele obiecte care se proiectează pe pixel, algoritmul ray-tracing intersectează (teoretic) fiecare rază cu fiecare obiect al scenei. Algoritmul z-buffer aproximează obiectele ca mulţimi de valori z în lungul liniilor de proiecţie ce intersectează obiectul, pe când algoritmul ray-tracing aproximează obiectele ca mulţimi de intersecţii în lungul fiecărei linii de proiecţie ce intersectează scena.

Problema principală asociată cu acest algoritm este timpul de calcul al intersecţiilor. Soluţiile acestei probleme sunt:

1. Optimizarea calculelor de intersecţii. Pot fi utlizate volumele minime ce încradrează obiectele, poliedre convexe formate din intersecţia unui set de "plăci" infinite, fiecare definită de o pereche de plane paralele care mărginesc obiectul (figura 4.16.a). Intersecţia razei cu volumul este definită de intersecţiile razei cu plăcile de margine.

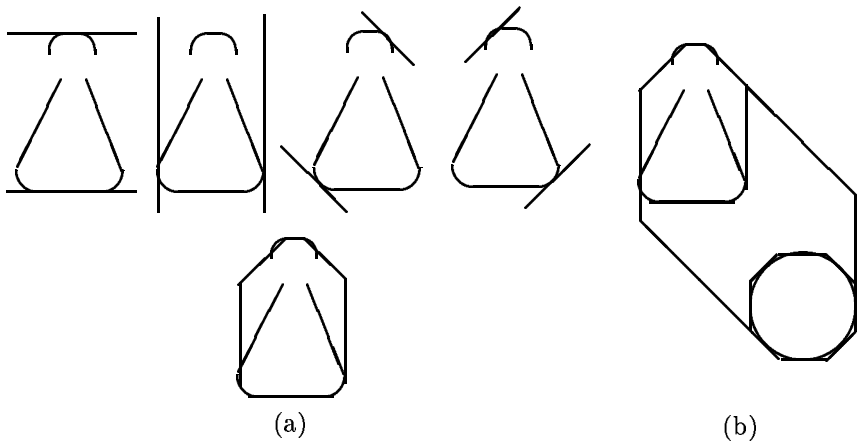


Figura 4.16: (a) Plăcile infinite ce încadrează un obiect bidimensional (b) Ierarhia volumelor

2. Evitarea calculelor de intersecţii. Se aplică o tehnică de preprocesare pentru partiţionarea razelor şi obiectelor în clase pentru a limita numărul de intersecţii cerute, pe bază de:
   (a) ierarhie: volumele mărginite pot fi organizate într-o ierarhie cu obiectele scenei la nivelul inferior şi volume mărginite ca noduri interne (figura 4.16.b). Un volum-fiu nu va intersecta o rază dacă volumul-tată nu o intersectează. Astfel testul de intersecţie începe cu rădăcina şi multe ramuri ale ierarhiei sunt respinse uşor. Procedura poate fi îmbunătăţită printr-o evidenţă a intersecţiilor corespunzătoare unei raze.
   (b) partiţionare: ierarhia volumelor mărginite organizează obiectele de jos în sus, iar partiţionarea divide spaţiul de sus în jos. Se determină prima dată volumul care mărgineşte scena. Acest volum este divizat

în subvolume egale, fiecărei partiţii fiindu-i asociată o listă de obiecte pe care le conţine parţial sau în întregime. Se vor căuta intersecţiile numai cu acele obiecte care se află în volumele prin care trece raza. În cazul intersecţiei unui corp cu raza este necesar testul de apartenenţă a punctului de intersecţie la partiţie (figura 4.17). Pentru a nu efectua intersecţiile unui corp cu raza de mai multe ori este necesară întreţinerea unei liste a intersecţiilor funcţie de obiect.

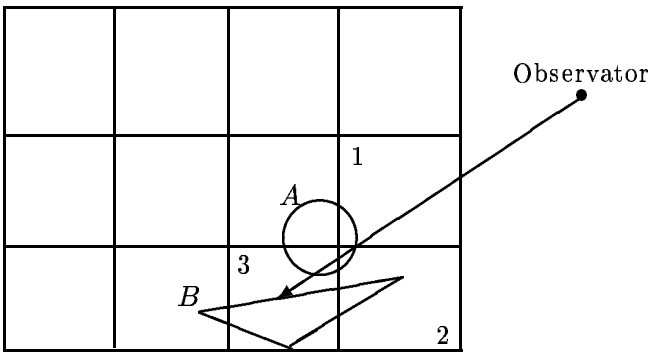


Figura 4.17: Divizarea volumului care mărgineşte scena

4.2.5 Algoritmi hibrizi: algoritmi cu listă de prioritate

Algoritmii cu listă de prioritate determină ordinea vizibilităţii obiectelor. De exemplu, dacă nici un obiect nu se suprapune cu altul în extensiile după z, atunci este necesară doar o sortare a obiectului în ordinea crescătoare a lui z şi trasarea lor în ordine. Poligoanele mai apropiate acoperă ulterior poligoanele mai îndepărtate deja trasate. Dacă obiectele se suprapun în extensiile lor după axa z este necesară împărţirea lor în mai multe obiecte astfel încât ordonarea menţionată să fie posibilă.

Un algoritm de acest tip este hibrid, deoarece combină operaţii în spaţiul obiect cu operaţii în spaţiul imagine. Comparările şi împărţirea obiectelor se realizează cu precizie obiect. Conversia scan care permite rescrierea pixelilor este realizată cu precizie imagine. Deoarece lista obiectelor sortate este creată cu precizie obiect, se poate realiza o retrasare corectă la orice rezoluţie.

Algoritmul de sortare în adâncime

Algoritmul de sortare în adâncime (depth-sort), cunoscut şi sub numele de algoritmul pictural sau algoritmul Newell-Newell-Sancha, converteşte poligoanele unei reprezentări poliedrale în frame-buffer în ordinea descrescătoare a distanţei de la observator. Se parcurg trei etape:

- se sortează poligoanele conform ordinii crescătoare a valorii z celei mai mici a vârfurilor (sortare în spaţiu-obiect);
- rezolvă orice ambiguitate cauzată de suprapunerea extensiilor z (comparare în spaţiu-imagine) prin descompunerea poligoanelor la nevoie;
- se transpun prin baleiaj, pe rând, poligoanele pe ecran, în ordinea crescătoare a celei mai mici coordonate z. Poligoanele îndepărtate sunt primele transpuse pe ecran. Cele mai apropiate le maschează prin suprascriere.

Compararea dintre două poligoane presupune cinci nivele de testare (eliminatorii). Dacă extinderile z a două poligoane consecutive P_1 şi P_2 din lista sortată se suprapun, atunci:

(1) Se compară extinderile după x. Dacă sunt disjuncte, poligoanele nu se suprapun;

(2) Se compară extinderile după y. Dacă sunt disjuncte, poligoanele nu se suprapun;

(3) Se testează dacă P_1 este în partea opusă observatorului faţă de planul lui P_2. Se determină ecuaţia planului lui P_2. Planul împarte spaţiul în 2 zone. Dacă P_1 este în întregime în zona mai îndepărtată, atunci este transpus primul pe ecran;

(4) Se testează dacă P_2 este de aceeaşi parte cu observatorul faţă de planul lui P_1. Se determină ecuaţia planului lui P_1. Dacă P_2 se află în întregime în zona mai apropiată de observator, P_1 este transpus pe ecran;

(5) Dacă primele patru teste nu au succes, se presupune că P_1 îl maschează pe P_2 şi P_2 trebuie trasat înaintea lui P_1. Se interschimbă poziţia în listă şi se repetă testele (3) şi (4).

Dacă cele cinci teste eşuează, se compară proiecţiile celor două poligoane. Se calculează intersecţiile latură cu latură ale celor 2 proiecţii. Dacă cel puţin una din intersecţii este simultan inclusă în laturile ambelor poligoane, atunci poligoanele se suprapun, se determină intersecţia celor două poligoane şi fiecare poligon este înlocuit în listă prin poligoanele componente. În caz contrar, poligoanele nu se maschează unul pe altul şi se transpun pe ecran în oricare ordine.

4.3 Iluminare şi umbre

Pentru ca imaginea să fie cât mai realistă este utilă simularea iluminării obiectului precizând:

1. sistemul de surse luminoase cu care se face iluminarea (tipul surselor, poziţia acestora în sistemul de referinţă adoptat),
2. caracteristicile optice ale suprafeţelor. Există astfel:
   (a) zone mate, care dispersează lumina reflectând-o în multe direcţii;
   (b) zone sclipitoare, care reflectă lumina numai după anumite direcţii;
   (c) zone transparente sau translucide cu fenomene de refracţie şi atenuare;
3. poziţia relativă a suprafeţelor corpului şi a sistemului de surse luminoase.

Procesele de eliminare a suprafeţelor ascunse şi simulare a iluminării se pot desfăşura simultan.

4.3.1 Surse de lumină

Sursele de lumină pot fi punctiforme (punctuale) sau distribuite. Exemple de surse punctiforme sunt becurile cu incandescenţă, flăcările de dimensiuni mici. Ca exemplu de sursă distribuită se poate menţiona tubul fluorescent. Sursele punctiforme produc un efect realist deoarece iluminarea unei suprafeţe depide de orientarea ei, mai precis de cosinusul unghiului dintre normala la suprafaţă în fiecare punct considerat pe aceasta şi dreapta care uneşte punctul respectiv cu sursa. Iluminarea este mai puternică pentru suprafeţe orientate perpendicular faţă de razele de lumină provenite de la sursă şi scade odată cu înclinarea suprafeţelor faţă de direcţia luminii.

Pe lângă aceste tipuri de surse se pot întâlni situaţii în care obiectele sunt scăldate într-o lumină de intensitate în general moderată şi care pare să vină din toate părţile - aceasta poartă numele de lumină ambiantă (produs al reflexiilor multiple de la suprafeţele aflate în scenă). Este cea mai uşor de modelat deoarece iluminarea suprafeţelor este constantă şi independentă de orientarea lor. Imaginea formată depinde astfel numai de intensitatea luminii ambiante şi de proprietăţile suprafeţei corpului modelat. Acest tip de iluminare nu produce o imagine realistă (muchiile de joncţiune ale poligoanelor cu aceleaşi proprietăţi superficiale nu se pot distinge). În realitate este puţin probabil să se întâlnească situaţii în care un anumit corp să se afle numai în lumina ambiantă.

Dacă sursa se află la o distanţă suficient de mare de obiectul iluminat pentru ca razele incidente să fie considerate paralele între ele, iluminarea este paralelă. Prin acest procedeu poate fi modelată mai bine lumina solară. Pentru precizarea unei surse de lumină paralelă sunt suficiente intensitatea iniţială, direcţia şi sensul de iluminare. Dacă în cazul iluminării cu surse punctiforme, distanţa la sursă este o informaţie importantă, în cazul iluminării paralele influenţa ei devine nesemnificativă, intensitatea fiind practic invariantă cu distanţa de la sursă.

Variaţiile intensităţii luminoase pe suprafaţa unui obiect oferă informaţii importante cu privire la forma suprafeţei respective. Informaţii suplimentare pot fi obţinute prin analiza umbrelor pe care elemente ale corpului modelat le aruncă asupra unui fundal al imaginii sau asupra altor elemente ale aceluiaşi corp.

4.3.2 Modele de iluminare

Ecuaţia iluminării variază în funcţie de tipul sursei luminoaase.

Lumina mediului înconjurător (lumina ambiantă) se răsfrânge egal pe toate suprafeţele şi în toate direcţiile, astfel încât ecuaţia iluminării este

$$I = I_a k_a$$

unde I este intensitatea rezultată, I_a este intensitatea luminii ambiante, constantă pentru fiecare obiect, iar k_a este coeficientul de reflexie ambiantă, procentul în care suprafaţa reflectă lumina ambiantă (între 0 şi 1), caracteristică a materialului.

În cazul reflexiei difuze (reflexia lambertiană), se consideră obiectul luminat de o sursă punctuală de lumină a cărui raze emană uniform în toate direcţiile ce pornesc de la un singur punct. Suprafeţele mate au proprietatea de a împrăştia lumina incidentă, prin reflecţie difuză, în mod aproximativ egal după toate direcţiile posibile. Oriunde s-ar afla observatorul, o aceeaşi placă va avea o imagine de strălucire constantă. Intensitatea luminii reflectate de o anumită faţetă din reprezentarea poliedrală a unui corp este independentă în acest caz de poziţia observatorului. Ea depinde numai de unghiul θ dintre direcţia L de la sursa de lumină şi normala la suprafaţă N (figura 4.18.a). Faţetele luminate după o direcţie mai apropiată de normală vor reflecta lumina cu intensitate mai mare.

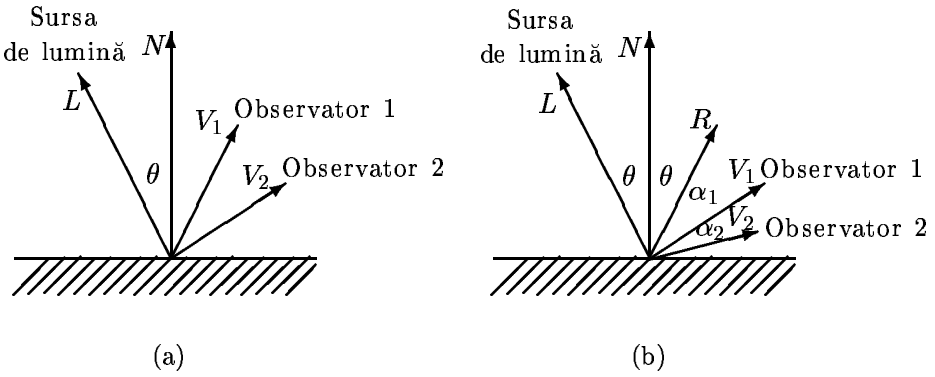


Figura 4.18: Iluminare (a) reflexie difuză (b) reflexie speculară

Intensitatea luminii reflectate se exprimă după legea reflexiei cosinus:

$$I = I_p k_d \cos\theta,$$

unde I este intensitatea luminii reflectate, I_p este intensitatea sursei de lumină punctuală, iar k_d este un coeficient care depinde de natura şi proprietăţile optice ale materialului ce produce reflexia difuză, numit coeficient de reflexie difuză (între 0 şi 1). Dacă L şi N sunt versorul direcţiei de propagare a luminii (raza incidentă) şi, respectiv, versorul normalei la faţetă, atunci are loc egalitatea $I = I_p k_d (L \cdot N)$. Această relaţie nu ţine seama de variaţia intensităţii luminoase în raport cu distanţa la sursă. Astfel, proiecţiile a două suprafeţe paralele de material identic se pot suprapune într-o imagine uniformă. Pentru o sursă punctiformă ideală, intensitatea luminoasă este invers proporţională cu pătratul distanţei R dintre sursă şi punctul în care se măsoară această intensitate. În majoritatea situaţiilor reale, obiectele observate sunt scăldate şi în

lumină ambiantă, astfel încât ecuaţia iluminării este:

$$I = I_a k_a + (1/R^2) I_p k_d \cos\theta.$$

Sunt situaţii în care observatorul se află faţă de sursă la o distanţă mult mai mică decât sursa luminoasă, ceea ce face ca influenţa lui R^2 să fie exagerată şi imaginea nerealistă. Astfel, reprezentările cele mai realiste se obţin folosind formula

$$I = I_a k_a + I_p k_d \cos\theta/(d + D),$$

unde d este o cotă care dă depărtarea relativă a faţetei considerate de sursă (pentru cea mai apropiată faţetă faţă de sursă d se poate lua 0). D este termenul de atenuare şi este un număr suficient de mare, astfel ales încât variaţia lui d pe domeniul său să provoace variaţia lui I în limitele dorite. Mai general,

$$I = I_a k_a + f_{at} I_p k_d \cos\theta,$$

unde f_{at} este factorul de atenuare. Lumina colorată este tratată scriind ecuaţiile separat pentru fiecare componentă a modelului de culoare. Cele trei componente primare ale luminii I_{pR}, I_{pG}, I_{pB} sunt reflectate în proporţiile $k_d O_{dR}$, $k_d O_{dG}$, $k_d O_{dB}$. De exemplu, pentru componenta roşie ecuaţia iluminării este

$$I_R = I_{aR} k_a O_{dR} + f_{at} I_{pR} k_d O_{dR} \cos\theta.$$

Reflexia speculară (selectivă) este asociată cu suprafeţele lucioase. Se consideră, de exemplu, un glob roşu metalizat din pomul de iarnă, sau un măr lustruit, iluminat cu o sursă punctiformă. Pe suprafaţa acestora se pot distinge zone de roşu închis, care reflectă lumina mai puţin intens, zone care reflectă lumina cu un roşu intens, precum şi o mică zonă care apare albă şi pare o sursă de lumină de aceeaşi natură cu lumina incidentă aflată pe suprafaţa obiectului. La schimbarea poziţiei observatorului pata de lumină îşi schimbă şi ea poziţia pe suprafaţa corpului. Această pată este produsul reflexiei selective sau speculare, în timp ce intensităţile provenind de la celelelte puncte de pe suprafaţa corpului sunt rezultatul reflexiei difuze.

Reflexia selectivă se produce deoarece suprafeţele netede şi lustruite reflectă lumina în mod inegal. O suprafaţă perfect lucioasă (o oglindă perfectă) reflectă lumina numai după direcţia din planul de incidenţă pentru care unghiul de reflexie este egal cu cel de incidenţă (în direcţia R, simetrica lui L faţă de N din figura 4.18.b). Se consideră α unghiul dintre direcţia de observare V şi direcţia de reflexie speculară R.

Suprafeţele pot fi şi oglinzi imperfecte (de exemplu, obiectele din vinilin). Pentru astfel de suprafeţe intensitatea reflexiei speculare scade brusc odată cu creşterea diferenţei dintre unghiul de reflexie real şi unghiul de incidenţă. Modelul empiric de simulare a reflexiei speculare presupune descreşterea intensităţii luminoase proporţional cu $\cos^m|\alpha|$ unde m este un număr natural nenul, care depinde de natura suprafeţei iluminate, exponentul de reflexie speculară al materialului (de la 1 la ordinul sutelor). Pentru o oglindă perfectă, $m \to \infty$, astfel încât intensitatea reflectată este maximă pentru $\alpha = 0$. Se introduce noţiunea

de factor de reflexie speculară, care este o funcţie de unghiul de incidenţă θ, $S : [0, 90^o] \rightarrow [0, 1]$ şi care arată ce fracţie din intensitatea luminoasă incidentă este reflectată specular în cazul unei incidenţe θ. Ecuaţia iluminării este

$$I_\lambda = I_a k_a O_{d\lambda} + f_{at} I_p [k_d O_{d\lambda} \cos\theta + S(\theta) \cos^m |\alpha|],$$

unde $\lambda = R, G, B$.

Pentru simularea transparenţei nerefractate se utilizează două metode:

1. transparenţă interpolată: dacă poligonul transparent 1 se intercalează între observator şi poligonul opac 2, ecuaţia iluminării este

$$I_\lambda = (1 - k_{t1}) I_{\lambda 1} + k_{t1} I_{\lambda 2},$$

unde k_{t1} este coeficientul de trasparenţă al poligonului 1 (între 0 şi 1). Dacă $k_{t1} = 0$, poligonul este opac şi nu transmite lumină. Dacă $k_{t1} = 1$, poligonul este perfect transparent. Valoarea $1 - k_{t1}$ desemnează opacitatea poligonului;

2. transparenţă filtrată: poligonul 1 este un filtru transparent, astfel încât

$$I_\lambda = I_{\lambda 1} + k_{t1} O_{t\lambda} I_{\lambda 2},$$

unde $O_{t\lambda}$ este transparenţa culorii λ a poligonului 1.

4.3.3 Iluminarea reprezentărilor poliedrale

Metodele de modelare a iluminării reprezentărilor poliedrale cele mai des utilizate sunt următoarele:

1. metoda iluminării constante;
2. metoda interpolării intensităţii luminoase;
3. metoda interpolării normalei la suprafaţă.

În metoda iluminării constante intensitatea se calculează o singură dată pentru fiecare poligon, deci unui poligon îi este caracteristică în condiţiile precizate de iluminare şi observare o anumită intensitate, constantă pe toată suprafaţa sa. Există situaţii pentru care calculele de iluminare se pot face cu viteză foarte mare şi anume cele în care direcţiile normalelor la faţetele corpului sunt în număr mic, faţetele fiind în număr relativ mare, dar grupate în clase de faţete paralele (situaţie des întâlnită în scene arhitecturale). Pentru fiecare direcţie din scenă se calculează intensitatea corespunzătoare. Faţetele se sortează în clase după direcţiile normalelor. Dacă nu se ţine cont de variaţia iluminării cu distanţa la sursă, atunci fiecare clasă se transpune pe ecran cu intensitatea calculată pentru direcţia reprezentativă clasei respective. Dacă se ţine seama de variaţia iluminării cu distanţa, atunci faţetele din fiecare clasă se sortează în subclase după distanţa la sursă, intensitatea corespunzătoare fiecărei subclase calculându-se prin incrementare.

Pentru ca o situaţie de observare reală să dea o imagine de tip iluminare constantă, e nevoie să fie îndeplinite trei condiţii:

(a) suprafaţa poliedrală este chiar suprafaţa corpului reprezentat şi nu o aproximare a acestuia;
(b) observatorul este la infinit astfel încât α să fie constant pentru fiecare suprafaţă poligonală;
(c) sursa de lumină se află la infinit, (θ constant pentru fiecare suprafaţă poligonală).

Datorită acestor condiţii, mai ales a doua, situaţia este rar întâlnită în practică. Totuşi, datorită simplităţii de calcul, metoda este cea mai folosită. Calculule pentru fiecare poligon se fac într-un singur punct interior privilegiat al acestuia, de obicei centrul de greutate.

Principalul dezavantaj al iluminării constante este acela că muchiile faţetelor poligonale sunt accentuate artificial datorită efectului de bandă Mach. Astfel, dacă faţeta A este luminată, iar faţeta B umbrită, zona de muchie a faţetei A va apărea puternic luminată mărind contrastul cu faţeta B (deci scoţând muchia în evidenţă), iar zona de frontieră a faţetei B apare mai umbrită. În cazul imaginilor monocrome, când umbra şi lumina sunt simulate prin densitatea de puncte, efectul este mai puţin sesizabil.

Metoda interpolării intensităţii luminoase dă rezultate calitativ superioare. Interpolarea vectorului de intensitate se face astfel:
(a) se calculează normalele la fiecare poligon;
(b) se calculează vectorii normali medii în fiecare vârf al poliedrului, adică se calculează în fiecare vârf media versorilor direcţiilor normale la toate faţetele alăturate vârfului;
(c) se calculează în fiecare vârf intensităţile locale, folosind normalele medii anterior calculate;
(d) pentru fiecare linie de baleiaj şi pentru fiecare poligon se parcurg etapele următoare:
   - se calculează intensităţile locale în intersecţiile liniei de baleiaj cu laturile poligonului (extremităţile segmentului de linie situat în poligon) prin interpolare liniară între valorile corespunzătoare vârfurilor ce mărginesc laturile respective;
   - pentru fiecare pixel de pe segmentul de linie de baleiaj inclus în poligon se calculează intensitatea corespunzătoare prin interpolare liniară între valorile calculate la extremităţile segmentului, valorile obţinute transpunându-se pe ecran.

De exemplu, pentru triunghiul din figura 4.19 intensităţile punctelor de pe linia de baleiaj sunt:

$$I_a = I_1-(I_1-I_2)\frac{y_1-y_p}{y_1-y_2},\ I_b = I_1-(I_1-I_3)\frac{y_1-y_p}{y_1-y_3},\ I_p = I_b-(I_b-I_a)\frac{x_b-x_p}{x_b-x_a}.$$

Cele mai bune rezultate se obţin prin metoda interpolării normalei la suprafaţă. După ce se calculează normalele medii locale în toate punctele care modelează corpul, se calculează, pentru fiecare segment de linie de baleiaj conţinut într-o faţetă, normalele în extremităţi prin interpolare între normalele medii locale calculate în capetele laturii pe care se află fiecare extremitate. Apoi

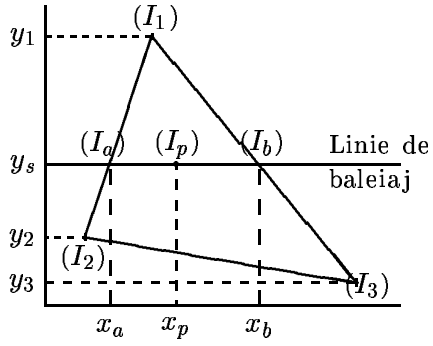


Figura 4.19: Interpolarea intensităţii luminoase

se interpolează normala într-un pixel de pe linia de baleiaj, între normalele calculate în extremităţi (relaţiile sunt analoage celor din metoda descrisă anterior, intensităţile fiind înlocuite cu vectori). Calculele sunt complicate căci normalele calculate sunt mărimi vectoriale şi deci interpolarea se face după cele 3 componente ale vectorului considerat în sistemul de referinţă xyz. Intensitatea este calculată numai după determinarea în fiecare pixel a normalei locale. Metoda nu este indicată atunci când display-ul este monocrom (când este necesară simularea intensităţii prin densitate de puncte).

4.3.4 Umbre

Dacă iluminarea nu este frontală, atunci devin importante poziţiile surselor punctiforme şi direcţiile după care se face iluminarea paralelă. Există posibilitatea ca părţi vizibile ale corpului să nu fie iluminate, fiind umbrite de alte părţi. Pentru reprezentarea corectă a umbrelor în cazul iluminării cu surse punctiforme, calculele de vizibilitate pentru corpul considerat se fac de două ori: din poziţia observatorului şi din poziţia sursei. Pentru fiecare pixel se urmăreşte:

- dacă este văzut atât de sursă cât şi de observator, atunci pixelul este vizibil şi luminat de sursă;
- dacă nu se vede din poziţia observatorului, atunci pixelul este invizibil;
- dacă se vede din poziţia observatorului, dar nu se vede din poziţia sursei, atunci este vizibil, dar umbrit.

Algoritmii pentru determinarea umbrelor, respectiv a vizibilităţii, sunt de fapt aceiaşi.

Cele mai des utilizate metode sunt următoarele:

1. Generarea scan a umbrelor. Pentru o reprezentare poliedrală, utilizând sursa de lumină drept centru al proiecţiei, laturile poligoanelor care potenţial produc umbre sunt proiectate pe poligoanele care intersectează linia curentă de baleiaj. Când linia de baleiaj intersectează una din laturile acestor poligoane de umbră, culoarea imaginii este modificată

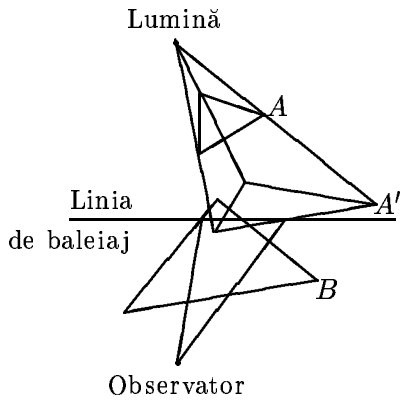


Figura 4.20: Poligonul A produce umbra A' în planul poligonului B

(figura 4.20). O implementare brută a acestui algoritm presupune calculul proiecţiilor fiecărui poligon pe oricare alt poligon. Se poate introduce o fază de preprocesare în care toate poligoanele reprezentării sunt proiectate pe o sferă centrată în sursa de lumină. Perechile de proiecţii ale căror extensii nu se suprapun pot fi eliminate.

2. Utilizarea algoritmilor spaţiu-obiect cu două parcurgeri. Se determină în primul rând suprafeţele vizibile din punctul de vedere al sursei de lumină, ieşirea fiind o listă de poligoane luminate, fiecare asociat cu numărul de identificare a poligonului din care provine. Se aplică apoi algoritmul din punctul de vedere al observatorului, utilizând o copie a bazei de date anterior create, rezultând în final o nouă listă de poligoane.
3. Utilizarea algoritmilor spaţiu-imagine cu două parcurgeri (algoritmul z-buffer). În primul rând se construieşte z-buffer-ul imaginii din punctul de vedere al sursei de lumină. Apoi z-buffer-ul şi imaginea sunt modificate funcţie de poziţia observatorului. Când un pixel este determinat a fi vizibil coordonatele sale în spaţiul obiect din punctul de vedere al observatorului, (x_o, y_o, z_o), sunt transformate în coordonatele sursei de lumină $(x_o{}', y_o{}', z_o{}')$. Coordonatele transformate, $x_o{}'$ şi $y_o{}'$, sunt utilizate pentru a selecta valoarea z_L din z-buffer-ul sursei de lumină. Dacă z_L este mai apropiat de lumină decât $z_o{}'$, atunci există ceva care blochează lumina de la sursă şi pixelul este umbrit; altfel pixelul este vizibil din punctul de vedere al luminii, deci nu este umbrit.

4.4 Umplerea poligoanelor

Umplerea poligoanelor presupune, în primul rând, un algoritm pentru determinarea poziţiei unui punct relativ la un poligon (se testează dacă punct este sau nu în interiorul poligonului şi în caz afirmativ, se aprinde, eventual cu o anumită culoare sau intensitate).

Fiecare algoritm de umplere poate fi logic descompus în următoarele:
1. metoda de propagare care determină următorul punct ce va fi tratat;
2. procedura de start care iniţializează algoritmul;
3. procedura de determinare a interiorului poligonului;
4. procedura de setare care schimbă starea pixelului.

Algoritmii de umplere se diferenţiază funcţie de următoarele două situaţii:
- umplerea se realizează odată cu trasarea poligonului;
- umplerea se realizează după ce poligonul a fost trasat.

În primul caz, problema principală este aceea de a defini interiorul şi exteriorul. Problema nu este simplă dacă poligonul nu este convex.

Regula par-impar, menţionată la algoritmii de determinare a liniilor şi suprafeţelor ascunse, este des utilizată pentru determinarea interiorului unui poligon. Se alege un punct oarecare din interiorul regiunii ce urmează a fi testată şi o linie dreaptă ce pleacă din acel punct spre o direcţie arbitrară şi nu trece prin nici un vârf. Fie pe această dreaptă un segment delimitat de punct şi altul oarecare ce nu aparţine interiorului dreptunghiului ce încadrează figura. Dacă segmentul intersectează laturile poligonului de un număr impar de ori, regiunea este considerată de interior. În figura 4.21 (a) se prezintă regiunile interioare ale unui poligon concav.

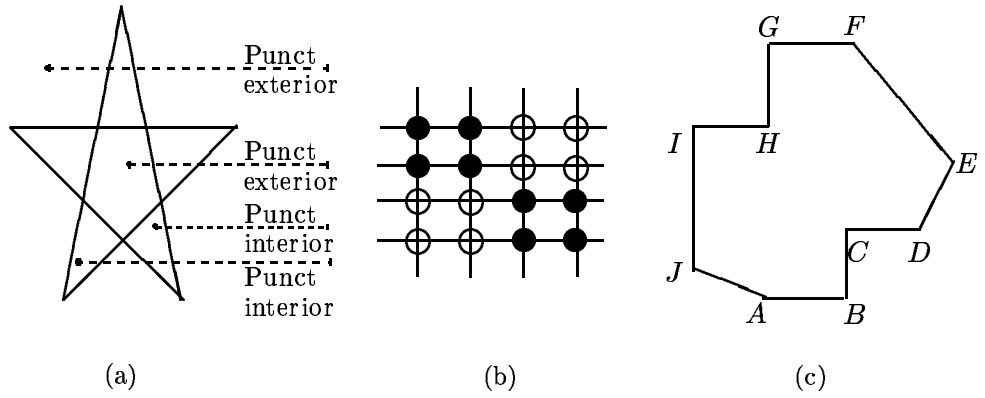


Figura 4.21: (a) Determinarea regiunilor interioare prin regula par-impar (b) Regiune 8-conexă în rastru (c) Exemplu pentru conversia scan

Tehnica par-impar este utilizată şi în procesul de selectare a unui obiect cu ajutorul cursorului grafic.

În cazul umplerii unui poligon deja trasat, intervin o serie de probleme adiţionale în implementarea algoritmilor. Aceste probleme sunt generate de structura discretă a frame buffer-ului. Una dintre acestea este determinarea laturilor unui poligon după ce acesta a fost deja figurat pe ecran. O altă problemă este determinarea interiorului.

O regiune din rastru este o colecţie de pixeli. Există două tipuri de regiuni conexe. O regiune este 4-conexă dacă oricare doi pixeli ai regiunii pot fi uniţi printr-o secvenţă de pixeli utilizând doar mutări în direcţiile sus, jos, stânga, dreapta. O regiune este 8-conexă dacă oricare doi pixeli ai regiunii pot fi uniţi printr-o secvenţă de pixeli prin mişcările anterior menţionate, plus cele pe diagonale. O regiune 4-conexă este 8-conexă, dar nu şi reciproc (vezi colecţia de pixeli aprinşi din figura 4.21.b). O regiune poate fi definită în două moduri:

1. o regiune definită prin interior relativ la un pixel P este cea mai mare regiune conexă de pixeli a căror valoare din frame-buffer este aceeaşi cu a lui P şi care conţine pe P;
2. o regiune definită prin frontieră relativ la un pixel de interior P şi unul de frontieră Q este cea mai mare regiune conexă de pixeli a căror valoare nu coincide cu valoarea frontierei (valoarea lui Q) şi care conţine pe P (interiorul poate conţine pixeli de culori diferite).

Algoritmii care umplu regiunile definite prin interior sunt numiţi algoritmi flood-fill, iar cei pentru regiunile definite prin frontieră, boundary-fill.

4.4.1 Conversia scan a poligoanelor pline

Distincţia dintre interior şi exterior este realizată la trecerea liniei de baleiaj. Se presupune că iniţial se porneşte din afara oricărui poligon. Astfel după prima intersecţie cu o latură, se trece de la exterior la interior. La a doua intersecţie cu o latură a unui poligon se trece de la interior la exterior. Repetând acest procedeu, se observă că după un număr impar de intersecţii, pixelii sunt în interior, iar după un număr par, în exterior (consecinţă a regulii par-impar). Prin convenţie, laturile se consideră interioare poligonului.

Pentru fiecare linie de baleiaj, algoritmul de umplere presupune:

1. determinarea intersecţiilor liniei de baleiaj cu fiecare din laturile poligonului;
2. sortarea intersecţiilor în ordinea crescătoare a absciselor;
3. aprinderea pixelilor între perechi de intersecţii care se află în interiorul poligonului pe baza parităţii în curs. Paritatea iniţială este pară. Aprinderea pixelilor are loc numai dacă paritatea este impară. Paritatea se schimbă la intersecţia laturilor poligonului.

La intersecţia unui vârf ale cărui coordonate coincid cu centrul unui pixel, paritatea se schimbă, cu excepţia cazului în care vârful are ordonata y maxim local.

Se consideră, de exemplu, poligonul din figura 4.21 (c). Se porneşte de la latura de jos. Cum la A, y este minim, iar AB coincide cu linia de baleiaj, paritatea este impară şi segmentul AB este trasat. Cum BC are B ca minim, paritatea devine, după B, pară. La vârful J, latura IJ are J ca minim pe axa y, dar nu şi latura JA, astfel încât paritatea devine impară şi intervalul de umplere începe cu J. Intervalul de umplere care porneşte de la latura IJ şi atinge C continuă, deoarece C este maxim local pe axa y (pentru BC şi CD). La D, minim local, paritatea se schimbă şi intervalul de umplere este închis. La I, maxim local, paritatea rămâne şi segmentul IH nu este umplut. La H,

minim local, paritatea se schimbă şi intervalul de umplere este deschis la H şi închis la intersecţia cu EF. În mod similar, GF nu este trasat.

Evidenţa mulţimii de laturi ce intersectează o linie de baleiaj se realizează printr-o structură numită tabelul laturilor active (AET). Laturile din acest tabel sunt sortate după abscisele x ale intersecţiilor. La trecerea la o nouă linie de baleiaj, se actualizează tabelul: se elimină laturile care nu mai intersectează linia de baleiaj, se adaugă noile laturi intersectate şi se calculează valorile abscise pentru laturile care au rămas tabel, printr-un algoritm incremental. Pentru simplificarea operaţiei de adăugare de laturi se creează un tabel al laturilor (ET) care conţine toate laturile sortate după coordonatele y cele mai mici. Tabelul de laturi este constituit din mai multe subtabele corespunzătoare în număr cu numărul liniilor de baleiaj. Fiecare subtabel conţine descrierea laturilor în ordinea crescătoare a absciselor x a punctelor ce se află pe linia de baleiaj. Fiecare intrare a subtabelelor conţine coordonata y maximă a laturii, coordonata x a capătului laturii cu y minim şi incrementul lui x utilizat la trecerea de la o linie de baleiaj la alta (1/pantă). În figura 4.22 se prezintă un exemplu sugestiv.

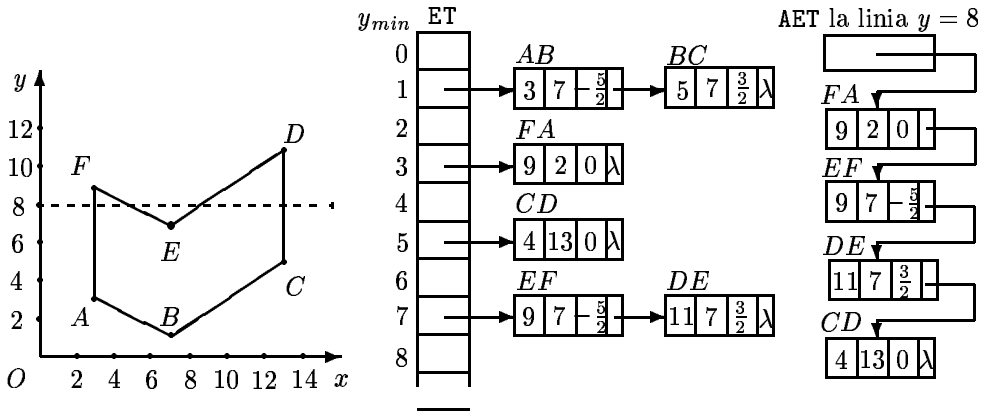


Figura 4.22: Tabele AET şi ET pentru conversia scan a unei linii

4.4.2 Algoritmi recursivi şi pe bază de stivă

În cazul în care poligonul este deja trasat în rastru, se pot utiliza tehnicile recursive de umplere. Se presupune că se cunoaşte un punct din interiorul poligonului. Dacă este aprins, atunci este pe o latură a poligonului. Se aplică tehnica par-impar pentru depistarea unui punct de interior care nu este pe nici o latură a poligonului. Dacă punctul interior considerat nu e aprins, se aprinde şi se continuă algoritmul pentru cei patru (sau opt) pixeli vecini. Dacă pixelul curent este aprins, se renunţă la apelul recursiv. Anumiţi pixeli pot fi vizitaţi însă de mai multe ori. Pentru suprafeţe mari, pot interveni probleme

de memorie – numărul mare de proceduri neterminate poate necesita o stivă sistem de dimensiune mare.

Un algoritm care evită problemele algoritmilor recursivi este cel care lucrează cu intervale de umplere. Aceste intervale sunt mărginite la ambele capete de pixeli de valoarea frontierei. Un interval este identificat prin pixelul său cel mai din dreapta. Se consideră un punct de interior cunoscut. Se umple intervalul continuu de pixeli ce conţine punctul de start. Apoi linia de deasupra este examinată de la dreapta la stânga pentru a determina cel mai din dreapta pixel al intervalului. Adresa acestui pixel este memorată în stivă. La fel se procedează cu linia de sub cea umplută. În figura 4.23 se dă un exemplu: intervalul de umplere ce conţine punctul de start este figurat în (a) (punctul de start este îngroşat). Adresele pixelilor numerotaţi sunt introduşi în stivă. Numerele indică ordinea din stivă, 1 fiind procesat ultimul. Algoritmul se opreşte când stiva este goală.

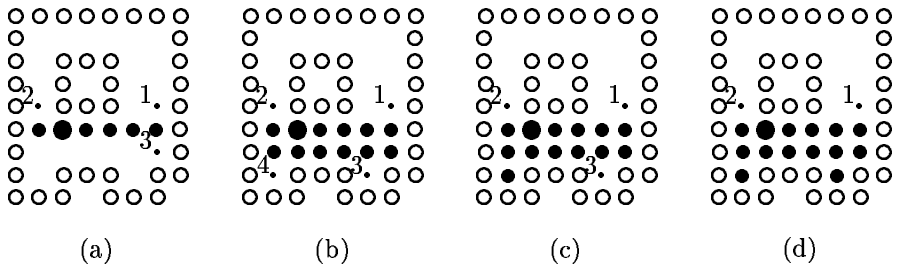


Figura 4.23: Paşii algoritmului de umplere pe bază de stivă

4.5 Culoare

Lumina acromatică este ceea ce se distinge pe ecranul unui televizor alb-negru. Cantitatea luminii (intensitatea, luminozitatea) este singurul atribut al luminii acromatice. Diferitelor niveluri de intensitate li se asociază anumiţi scalari (0, negru, 1, alb, între 0 şi 1, diferite scale de gri).

Pentru descrierea luminii cromatice se utilizează trei măsuri:

1. nuanţa, care face distincţie între culorile fundamentale;
2. saturaţia: culorile nesaturate includ mai multă lumină decât cele saturate. De exemplu, roşu este saturat, roz este nesaturat relativ la roşu;
3. luminozitatea, care este intensitatea luminii percepute.

Culoarea poate fi specificată prin tente, tonuri şi umbre. O tentă rezultă prin adăugarea unui pigment alb la un pigment de culoare pură, descrescând saturaţia. O umbră se obţine adăugând un pigment negru la culoarea pură, descrescând luminozitatea. Un ton este rezultatul adăugării atât a pigmentului

negru cât şi a unuia alb la pigmentul pur. În figura 4.24 se prezintă relaţia dintre tente, umbre şi tonuri.

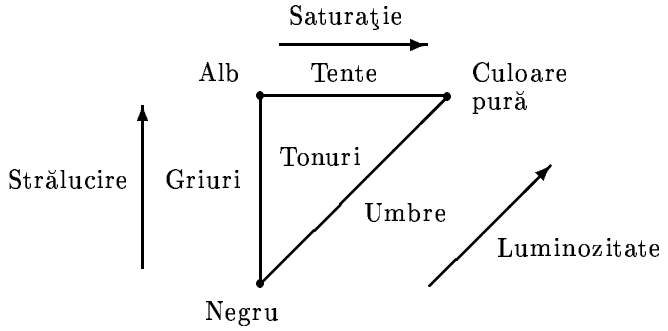


Figura 4.24: Relaţia dintre tente, umbre şi tonuri

4.5.1 Modele de culoare

Un model de culoare constă într-o specificare a unui sistem de coordonate tridimensionale şi a unei mulţimi de coordonate, în sistemul dat, ale cărui elemente sunt culori.

Exemple:

1. Modelul RGB (Red-Green-Blue) este cubul unitate din sistemul de coordonate carteziene a cărui vârfuri au semnificaţiile din figura 4.25 (a). Diagonala principală reprezintă nivelurile de gri. Acest model este utilizat la CRT color.
2. Modelul CMY (Cyan-Magenta-Yellow) este specificat printr-un cub asemănător şi este utilizat la anumite imprimante color.
3. Modelul HSB (Hue-Saturation-Brigtness, nuanţă-saturaţie-strălucire) este bazat pe intuiţia artistică pentru tente, umbre şi tonuri (figura 4.24). Sistemul de coordonate este cilindric şi subsetul din spaţiu în care modelul este definit este o piramidă cu şase feţe (figura 4.25.b). Nuanţa H este unghiul din jurul axei verticale, pornind de la roşu considerat 0^0. Culorile complementare formează un unghi de 180^0. Saturaţia S se măsoară de la axa verticală la laturile hexagonului. Valorile intermediare ale strălucirii V de la 0 la 1, pentru $S = 0$, corespund nivelurilor de gri. Adăugarea pigmentului alb corespunde descreşterii lui S fără schimbarea lui V. Umbrele sunt create păstrând $S = 1$ şi descrescând V. Tonurile sunt create descrescând atât S cât şi V.
4. Modelul HLS (Hue-Lightness-Saturation, nuanţă-luminozitate-saturaţie) este definit prin corpul din figura 4.25 (c).

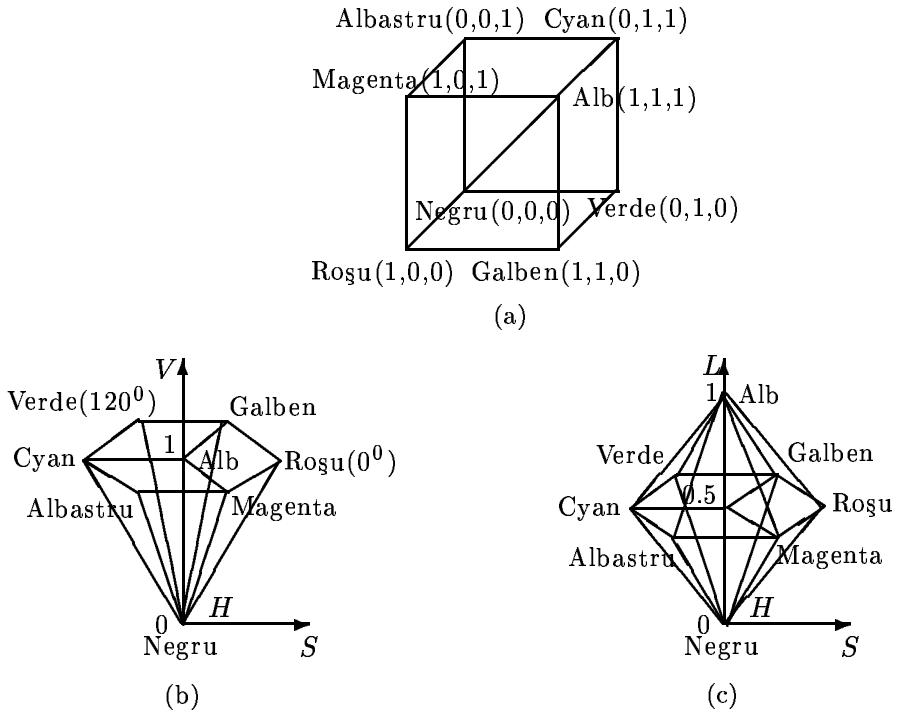


Figura 4.25: Modele de culoare (a) Modelul RGB (b) Modelul HSB (c) Modelul HLS

4.5.2 Metoda tabelului de culori

Într-un sistem cu rastru ce permite afişarea mai multor culori, numărul de biţi necesari pentru reprezentarea unui pixel în frame buffer depinde de:

- numărul de culori disponibile,
- metoda de stocare a valorilor-culoare.

Tabelele de culoare pot fi structurate pentru a permite utilizatorului folosirea unor combinaţii de culori fără a cere o dimensiune mare a zonei tampon. Un exemplu concludent este tabelul 4.1.

Când o culoare particulară este specificată într-un program aplicativ, valoarea binară corespunzătoare este stocată în frame buffer pentru fiecare pixel ce va fi afişat în acea culoare.

O altă metodă pentru stocarea codurilor de culoare a pixelilor (ce necesită o zonă tampon mai redusă ca dimensiune decât pentru metoda anterioară) este cea a tabelului de celule-culori (color lookup table, lut). Valorile din zona tampon cadru reprezintă indici în tabelul de celule-culori. Presupunem că se rezervă câte 4 biţi pentru informaţiile de intensitate (adică avem 16 niveluri de intensitate) pentru fiecare dintre cele trei tunuri de electroni (roşu, verde, albastru). Spre deosebire de cazul tabelului 4.1, fiecare intrare în tabel utilizează

Cod culoare	Roşu	Verde	Albastru	Culoare afişată
0	0	0	0	Negru
1	0	0	1	Albastru
2	0	1	0	Verde
3	0	1	1	Cyan
4	1	0	0	Roşu
5	1	0	1	Magenta
6	1	1	0	Galben
7	1	1	1	Alb

Tabelul 4.1: Exemplu de tabel de culori

12 biţi pentru specificarea unei culori, deci un total de 16^3=4096 culori diferite sunt disponibile. O referinţă de 6 biţi din frame-buffer (limitare datorată dimensiunii memoriei video) permite accesarea a 2^6=64 de poziţii în tabelul de celule.

Sistemele ce utilizează această tehnică permit utilizatorului să selecteze orice combinaţie de 64 de culori dintr-o paletă de 4096 de culori. Intrările în tabelul de celule pot fi schimbate oricând pentru a permite noi combinaţii de culori (schimbarea paletei de culori).

În figura 4.26 este schiţată organizarea tabelului de celule pentru o referinţă de 8 biţi. Pixelul de coordonate (x, y) ce are valoarea 67 în frame buffer este afişat pe ecran cu o culoare ce corespunde tunului roşu de electroni la 9/15 din maxim, cel verde la 10/15, iar cel albastru la 1/15.

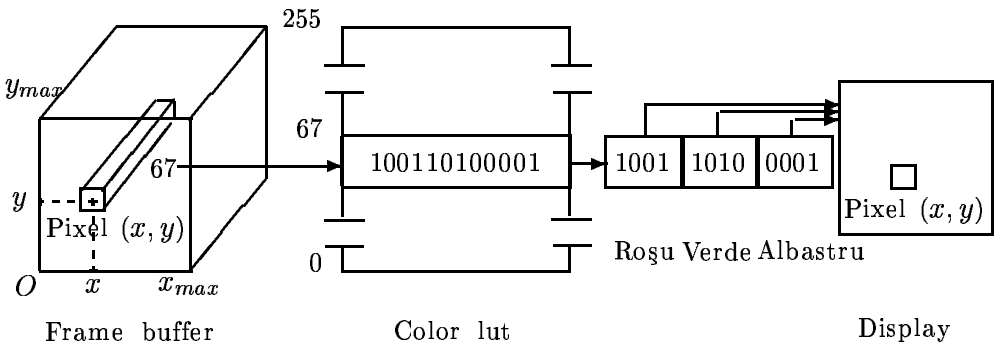


Figura 4.26: Tabelul de celule-culori pentru un sistem cu 8 biţi per pixel

Utilizarea tabelului de celule poate creşte puternic numărul opţiunilor de culori fără creşterea mărimii imaginii în memorie.

Sistemele cu culori de calitate înaltă utilizează 24 de biţi pentru fiecare poziţie în tabelul de celule şi 9 biţi per pixel în frame buffer. Astfel se permite se-

lectarea de către utilizator a 512 culori dintr-o gamă de aproximativ 16 milioane de culori-intrări ale tabelului de celule.

Pentru monitoarele care nu au capabilităţi privind culoarea, comanda de selectare a culorii poate fi utilizată în programele aplicative pentru a seta nivelul de intensitate, sau scala de gri. Numeroase aplicaţii utilizează valori numerice între 0 şi 1 pentru a seta nivelurile scalei de gri. Un exemplu este prezentat în tabelul 4.2.

Intensitate	Valorile în frame buffer	Culoarea afişată
0.0	0 (00)	Negru
0.33	1 (01)	Gri închis
0.67	2 (10)	Gri deschis
1.0	3 (11)	Alb

Tabelul 4.2: Codificarea nivelurilor de gri

Stocarea nivelurilor de intensitate este similară cu stocarea codurilor de culori. Dacă se alocă un bit pentru un pixel, sunt posibile doar două valori din scala de gri (negru şi alb). Dacă se alocă trei biţi per pixel, se permit 8 nivele diferite de intensitate. Sistemele cu calitate înaltă a imaginii permit 24 sau mai mulţi biţi pentru stabilirea nivelului de intensitate per pixel.

4.6 Texturi

4.6.1 Clasificare

Texturile sau detaliile ce apar pe suprafeţele obiectelor de modelat pot fi:

1. constante ca mărime şi orientare pe suprafaţa obiectului;
2. de mărime şi orientare variabilă, corespunzător poziţiei faţetelor a căror textură o reprezintă;
3. neregulate.

Texturile de primul tip sunt folosite pentru aplicaţiile 2D, iar în 3D se preferă texturile de al doilea tip.

Texturile de primul tip se codifică în baza de date prin matrice. Se alege un element repetabil prin translaţie pe orizontală şi verticală. Funcţie de precizia dorită şi rezoluţia disponibilă se digitizează acest element printr-o partiţie $m \times n$ a ariei pe care este reprezentat. Fiecărui pixel astfel obţinut i se asociază un element al unei matrice E cu m linii şi n coloane, ocupând poziţia corespunzătoare şi având valoarea 1 dacă simbolul conţine pixelul şi 0 dacă pixelul este neutilizat. Dacă se foloseşte un monitor care admite mai multe niveluri de intensitate pe pixel, atunci valoarea elementului E_{ij} poate fi chiar intensitatea corespunzătoare pixelului din poziţia i, j.

Detaliile superficiale, cum sunt textele sau simbolurile marcate pe suprafaţa obiectelor, pot fi tratate ca texturi de categoria a doua. O metodă de a le

reda constă în a le introduce în baza de date prin puncte caracteristice, pentru ca apoi să li se aplice aceleaşi transformări ca şi tuturor punctelor corpului. Se marchează însă apartenenţa texturii la faţetele pe care apare efectiv, iar transpunerea pe ecran se face numai pentru punctele vizibile ale poligoanelor respective.

Transpunerea pe ecran a texturii se poate face şi la o anumită scară. Dacă textura apare mărită, nu se pun probleme deosebite, pur şi simplu se construieşte o altă matrice pe baza lui E dar având $s \cdot m$ linii şi $s \cdot n$ coloane, unde s este factorul de scară. Dacă $s \cdot m$ şi $s \cdot n$ nu sunt numere întregi, ele se aproximează prin adaus, iar intensităţile în pixeli se aleg din gama disponibilă ca fiind cele mai apropiate de intensităţile medii calculate.

4.6.2 Simularea intensităţii pe monitoarele monocrome

Se consideră că alb şi negru sunt cele două niveluri de intensitate de care se dispune la nivelul unui pixel pe display.

Tehnica folosită este cea a texturilor constante. În principiu, se asociază fiecărui nivel de intensitate o textură cu o densitate corespunzătoare de puncte deschise. Simularea nivelurilor de intensitate luminoasă prin texturi de tip constant se bazează pe proprietatea ochiului uman de a acţiona ca integrator spaţial (la distanţe mari faţă de obiectul privit sau atunci când zona observată are dimensiuni mici). Metoda reduce evident din rezoluţie, căci un element de textură va fi reprezentat pe o arie din ecran de câţiva pixeli.

Tehnica aproximării nivelurilor de intensitate luminoasă prin densitate de puncte albe şi negre pe o anumită suprafaţă elementară (half-toning) este folosită în tipărirea fotografiilor alb-negru. Fiecare element de imagine are o fineţe precizată prin numărul de niveluri de intensitate care pot fi simulate.

Cel mai simplu element de imagine este o arie de 2×2 pixeli pe un ecran monocrom, pe care se pot produce 5 niveluri diferite de intensitate (figura 4.27.a). În general, o arie de $n \times n$ pixeli poate reproduce $n^2 + 1$ niveluri de intensitate. Pentru a folosi arii mai mari de 4×4 pixeli (peste 17 niveluri de intensitate) este necesar un monitor cu o rezoluţie mai mare de 512×512 pentru a obţine rezultate satisfăcătoare. Peste 32 de niveluri pe pixeli fac ca tranziţiile să fie abia percetibile. Imaginile realizate prin discretizarea intensităţii luminoase în 64 de clase au practic acelaşi aspect cu originalul.

Pentru o matrice 3×3 de pixeli se recomandă setul din figura 4.27 (b). Funcţie de ordinea în care se aprind pixelii se poate construi o matrice caracteristică. În cazul dat,

$$D = \begin{pmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{pmatrix}.$$

Pentru a aprinde o porţiune de pixeli cu intensitatea I între 0 şi 9 se aprind pixelii a căror valoare corespunzătoare în matrice este mai mică decât I. Dacă imaginea are dimensiunea matricii-rastru, problema aprinderii unui pixel (x, y) depinde de intensitatea dorită $I(x, y)$ în acel punct şi de matricea D. Fie n

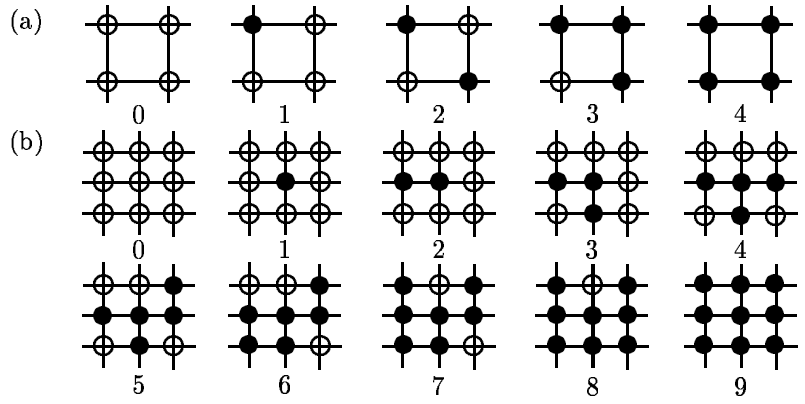


Figura 4.27: Simularea intensităţii – şabloane pentru tehnica half-toning cu (a) cinci niveluri de intensitate (b) nouă niveluri de intensitate

dimensiunea acesteia din urmă. Se calculează $i = x \bmod n$ şi $j = y \bmod n$. Atunci, dacă $I(x, y) > D_{ij}$, pixelul (x, y) se aprinde.

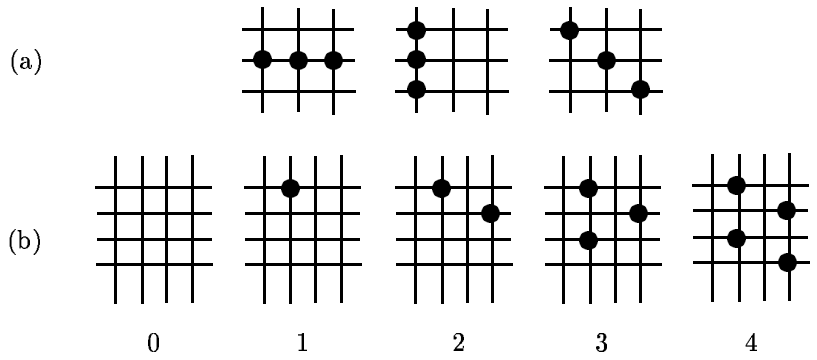


Figura 4.28: Texturi contraindicate

În construirea setului de texturi (pattern-uri) trebuie să se ţină seama de anumite reguli:

1. Texturile trebuie astfel proiectate încât să nu producă efecte vizuale artificiale în arii mari de intensitate identică. De exemplu, texturile din figura 4.28 (a) pot produce senzaţia de haşurare.
2. Texturile trebuie să constituie o secvenţă crescătoare, astfel încât orice pixel aprins pentru nivelul de intensitate j să rămână aprins şi pentru nivelurile de intensitate $k > j$. Se minimizează astfel diferenţele dintre texturi succesive şi efectele de contur.
3. Texturile trebuie să crească de la centru înspre exterior pentru a crea

efectul de creştere a mărimii punctului.

4. Pentru uşurarea trasării imaginilor pe imprimante, pixelii aprinşi trebuie să fie asociaţi cu pixeli vecini aprinşi. De exemplu, texturile din figura 4.28 (b) nu pot fi acceptate.

Tehnica half-toning este utilizată şi la display-uri binivel. Considerând un display cu doi biţi per pixel şi deci patru niveluri de intensitate, prin înjumătăţirea rezoluţiei se creşte numărul nivelurilor de intensitate la 13 (se consideră texturi de dimensiune 2×2: avem patru pixeli la dispoziţie, fiecare având posibilitatea de a lua trei valori în afara negrului, adică se obţin $4 \times 3+1=13$ niveluri de intensitate).

Utilizând un display color cu trei biţi per pixel, câte unul pentru fiecare culoare (roşu, verde, albastru) se pot utiliza texturi 2×2 pentru a obţine 125 culori diferite: fiecare textură poate afişa cinci intensităţi pentru roşu, verde sau albastru, rezultând 5^3 combinaţii de culori.

4.7 Modele fractale

Termenul de fractal din matematică a fost generalizat de comunitatea graficii pe calculator. În grafică, un fractal se referă la orice obiect care are o măsură substanţială a similarităţii în sine, statistică sau exactă. Astfel, numai fractalii generaţi prin procese recursive infinite sunt obiecte fractale adevărate. Pe de altă parte, obiectele generate prin procese finite pot conduce la schimbări nevizibile în detaliu după o anumită etapă, astfel încât sunt aproximări adecvate ale modelului ideal.

Ceea ce se înţelege prin similaritate în sine este sugerat de exemplul curbei lui Koch. Pornind de la o linie cu un salt (figura 4.29), se înlocuieşte fiecare segment al liniei cu o figură identică cu cea originală, redusă cu factorul trei. Acest proces este repetat: fiecare segment al figurii (b) este înlocuit cu figura asemănătoare întregii figuri (b). Dacă procesul este repetat la infinit, rezultatul este similar în sine, adică întregul obiect este similar (prin translaţie, rotire, scalare) cu o porţiune a sa.

Un obiect care arată la fel dacă este scalat se spune că este substanţial similar în sine.

Asociată cu noţiunea de similaritate în sine este cea de dimensiune fractală. Pentru a defini această noţiune se consideră următoarele obiecte:

1. Un segment de linie, care este unidimensional, divizat în n părţi egale. Ficare parte a sa arată precum originalul scalat cu factorul $n = n^{1/1}$.
2. Un pătrat (figură bidimensională) divizat în n subpătrate egale. Fiecare arată la fel precum originalul redus cu factorul $\sqrt{n} = n^{1/2}$ (de exemplu, considerînd nouă subpătrate, factorul este de reducere este 3).

Dacă divizăm imaginea rezultată prin aplicarea procedeului din figura 4.29, repetat la infinit, în patru părţi (analog bucăţilor asociate cu cele patru segmente originale din figura 4.29.a), fiecare bucată rezultată arată ca şi originalul redus cu factorul trei. Spunem că fractalul construit are dimensiunea d, dacă d verifică ecuaţia $4^{1/d} = 3$, adică $d = \log_2 4/\log_2 3 \approx 1.26$.

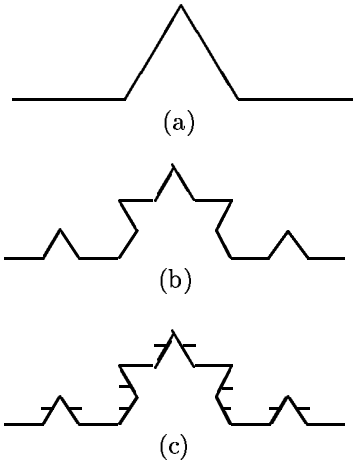


Figura 4.29: Construirea curbei lui Koch

Cele mai cunoscute obiecte fractale sunt

- mulţimea Julia-Fatou,
- mulţimea Mandelbrot.

Aceste obiecte sunt generate urmărind regula $x \to x^2 + c$, unde x este un număr complex.

Se reaminteşte faptul că dacă un număr complex are modulul subunitar, şirul recursiv al pătratelor converge la zero, iar dacă modulul este supraunitar, pătratele devin din ce în ce mai mari. Dacă modulul este unitar, şirul pătratelor are tot modulul unu. Numerele de modul unitar formează o frontieră între cele atrase spre zero şi cele atrase spre infinit.

Dacă se aplică repetat regula $x \to x^2 + c$ fiecărui număr complex, pentru o anumită valoare nenulă a lui c, anumite numere complexe vor fi atrase la infinit, altele spre numere finite. Trasând frontiera dintre cele două submulţimi se obţine mulţimea Julia-Fatou, care depinde de valoarea numărului complex c. Două exemple sunt prezentate în figura 4.30.

Dacă se calculează mulţimile Julia-Fatou pentru toate valorile c complexe şi se colorează cu negru punctele c pentru care mulţimea Julia-Fatou este conexă şi cu alb punctele pentru care mulţimea nu este conexă, se obţine mulţimea Mandelbrot din figura 4.31.

Din fericire, există o cale mai uşoară de a genera aproximaţii ale mulţimii Mandelbrot. Pentru fiecare valoare c se aplică regula $x \to x^2 + c$ cu $x = 0 + 0i$ de un număr finit de ori (de exemplu, de 1000 de ori). Dacă după aceste iteraţii numărul complex obţinut este în afara discului centrat în zero şi de rază dată (de exemplu, 100), atunci c va fi colorat alb, altfel negru. Cu cât numărul de iteraţii şi raza discului cresc, imaginea rezultată este o aproximare mai bună a mulţimii Mandelbrot.

(a) (b)

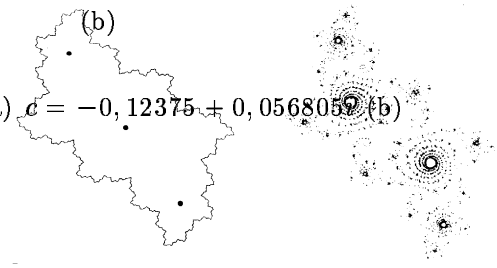


Figura 4.30: Mulţimea Julia-Fatou pentru (a) $c = -0,12375 + 0,0568059$ (b) $c = -0,012 + 0,74i$

Figura 4.31: Mulţimea Mandel

Imaginea pe ecran poate fi generată recur nulţi paşi astfel încît schimbările sub dimensiunile pixelului nu mai treb uate.

Rezultatele privind fractalii sunt extrem de sug ive pentru modelarea formelor naturale ca munţii, arborii sau coasta mării. De exemplu, se consideră următorul algoritm de generare a unui munte fractal, algoritm bazat pe subdivizarea recursivă.

Fie cazul unidimensional. Presupunem că pornim de la un segment de linie de pe axa x. Împărţim segmentul în două bucăţi şi mutăm mijlocul la o

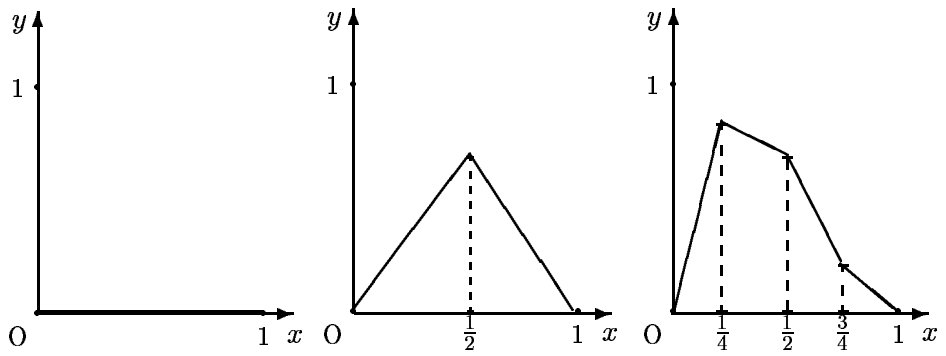


Figura 4.32: Generarea secţiunii printr-un munte fractal

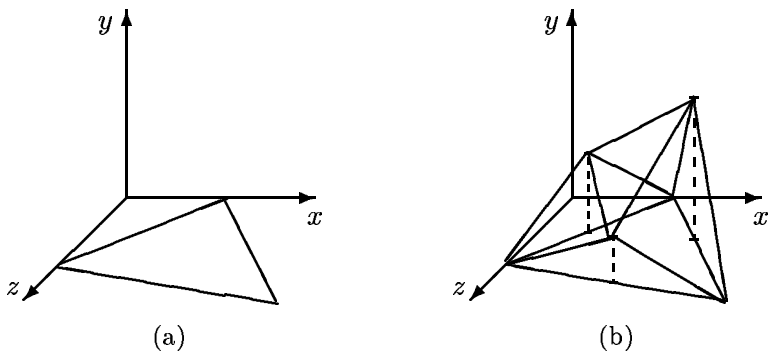


Figura 4.33: Generarea unui munte fractal

distanţă arbitrară în direcţia y, obţinând figura 4.32 (b). Se continuă subdivizarea fiecărui segment şi se calculează poziţia nouă a punctului de mijloc al segmentului de la (x_i, y_i) la (x_{i+1}, y_{i+1}):

$$x_{nou} = \frac{x_i + x_{i+1}}{2}, \; y_{nou} = \frac{y_i + y_{i+1}}{2} + P(x_{i+1} - x_i)R(x_{nou}),$$

unde $P(\cdot)$ este o funcţie ce determină extinderea perturbaţiei în termenii dimensiunii segmentului care se perturbă şi $R(\cdot)$ este un număr aleator între 0 şi 1 selectat pe baza lui x_{nou}. Dacă $P(s) = s$, atunci primul punct nu poate fi afişat mai sus de 1, fiecare din următoarele două puncte, nu mai sus de 1/2 ş.a.m.d., încât toate punctele rezultate se încadrează în pătratul unitate. Dacă $P(s) = s^a$ forma liniei frânte depinde de parametrul a. Pot fi utilizate şi alte funcţii, ca de exemplu $P(s) = 2^{-s}$.

Algoritmul poate fi uşor generalizat la două dimensiuni. Se porneşte de la un triunghi precum cel din figura 4.33. Se marchează mijloacele laturilor. Coordonatele y ale fiecărui punct de mijloc sunt modificate în maniera descrisă în cazul unidimensional, astfel încît rezultatul să fie un set de mai multe triunghiuri. Cele patru triunghiuri înclinate se procesează în mod analog. Repetarea recursivă a acestei divizări conduce la o imagine realistă a unui munte.

4.8 Animaţie

Animaţia, arta de a da viaţă unor imagini, presupune nu numai mişcare, ci şi o serie de efecte vizuale. Tehnica animaţiei constă în variaţia în timp a poziţiei, umbre, culori, transparenţă, texturi, schimbări de lumină şi puncte de vedere, apropiere şi îndepărtare (zoom).

Animaţia presupune o reprezentare a corpurilor în 4D.

Domeniile de aplicaţie sunt extrem de variate: industrie, educaţie, industrie, cercetare ştiinţifică, artă. Animaţia în vizualizarea ştiinţifică este realizată adesea pe baza unor seturi de date foarte largi ce reprezintă imagini 2D sau 3D ale unui anumit fenomen ce evoluează în timp.

Dacă anumite obiecte sau aspecte din imaginile animate se schimbă prea repede relativ la numărul de imagini afişate într-o secundă, apare fenomenul de clipire (temporal aliasing). Animaţia imaginilor video presupune o schimbare de cel puţin 30 de imagini per secundă (fps = frame per second).

4.8.1 Animaţie convenţională

Se realizează pe baza standardului următor: se concepe povestea, se înregistrează sunetele, se desenează imaginile cheie (key frames), se construiesc imaginile intermediare (inbetweening). Rezultatul este filmul de test (pencil test) care ulterior este colorat. Acest model de construcţie a imaginilor animate poartă denumirea de animaţie pe bază de imagini cheie (key-frame animation). Modelul se aplică şi în animaţia pe calculator.

Anumite etape ale animaţiei convenţionale se pretează la asistenţă computerizată, în special fazele de construire a imaginilor intermediare şi colorare. Înainte de aceste două faze, imaginile cheie sunt digitizate (prin scanare sau trasare cu ajutorul unor echipamente grafice specifice) şi sunt postprocesate (filtrate pentru eliminarea zgomotelor şi netezirea contururilor). În faza de compunere a imaginilor, în care figurile principale şi imaginile de fond sunt combinate pentru a genera imagini distincte, se pot utiliza de asemenea tehnici automate.

4.8.2 Interpolare

În procesul de creare a imaginilor intermediare apar o serie de probleme. Calculatorul primeşte o poziţie iniţială şi una finală. Ochiul uman percepe circumstanţele în care are loc interpolarea: de exemplu, dacă este vorba de

o aruncare la un anumit unghi a unei mingi sau o alunecare pe o suprafaţă. Interpolarea liniară (adecvată celei de a doua situaţii), care este uşor de realizat, are numeroase limitări. Date valorile v_i şi v_f ale unor atribute (poziţie, culoare, mărime) din imaginea iniţială, respectiv cea finală, valoarea v_t a unei imagini intermediare este $v_t = (1-t)v_i + tv_f$, cu variabila temporală $t \in [0,1]$. Valoarea v_t variază neted de la v_i la v_f. Dându-se trei imagini-cheie în aruncarea unei mingi sub anumit unghi, respectiv poziţia iniţială, cea finală şi cea corespunzătoare înălţimii maxime, prin interpolare liniară se creează o mişcare continuă, dar cu o schimbare bruscă a vitezei în punctul de maxim (interpolarea liniară generează discontinuităţi ale derivatelor). Astfel, pentru interpolarea imaginilor cheie consecutive se utilizează adesea funcţiile spline (vezi capitolul anterior, secţiunea curbe cubice).

O figură reprezentată printr-o polilinie poate fi interpolată între imaginile cheie prin interpolarea fiecărui vârf al poliliniei, de la poziţia sa iniţială la cea finală. Această idee este fundamentală în tehnica de animaţie pe bază de schelet, în care obiectelor în mişcare le sunt asociate "scheletele" (acestea sunt interpolate, şi nu obiectele în totalitate). Fie exemplul mişcării unui picior: scheletul poate fi constituit dintr-o linie poligonală cu patru vârfuri, respectiv vârful tălpii, mijlocul acesteia, centrul genunchiului şi un punct ce defineşte legătura cu corpul. Trasarea a 5-6 polilinii-schelet corespunzătoare, interpolarea cu funcţii spline a poziţiilor consecutive ale vârfurilor şi reconstituirea piciorului pe bază de schelet pot conduce la o imagine realistă a efectuării unui pas.

În animaţia unor obiecte tridimensionale trebuie interpolate atât poziţia cât şi orientarea acestora. Poziţia poate fi interpolată ca şi în cazul 2D prin specificarea poziţiei centrului unui obiect în fiecare din imaginile cheie. Problema principală este interpolarea orientării corpurilor. De obicei, o rotaţie este specificată prin trei rotaţii după axele principale. Ordinea aplicării acestor rotaţii de bază nu poate fi aleatoare. Mulţimea tuturor rotaţiilor posibile corespunde unei structuri algebrice: cuaternionii. Rotaţiile sunt cuaternionii unitari, care sunt simboluri de forma $a + b\overline{v_1} + c\overline{v_2} + d\overline{v_3}$, unde a, b, c, d sunt numere reale ce satisfac relaţia $a^2 + b^2 + c^2 + d^2 = 1$; cuaternionii se multiplică pe baza legii de distributivitate şi pe baza următoarelor reguli:

$$\overline{v_1}^2 = \overline{v_2}^2 = \overline{v_3}^2 = -1,$$

$$\overline{v_1 v_2} = \overline{v_3} = -\overline{v_2 v_1}, \quad \overline{v_2 v_3} = \overline{v_1} = -\overline{v_3 v_2}, \quad \overline{v_3 v_1} = \overline{v_2} = -\overline{v_1 v_3}.$$

Rotaţia de unghi φ în jurul vectorului unitate $(b, c, d)^T$ corespunde cuaternionului $\cos(\varphi/2) + b\sin(\varphi/2)\overline{v_1} + c\sin(\varphi/2)\overline{v_2} + d\sin(\varphi/2)\overline{v_3}$. Efectuarea unor rotaţii succesive presupune multiplicarea cuaternionilor corespunzători. Deoarece $a^2 + b^2 + c^2 + d^2 = 1$, cuaternionii pot fi priviţi ca puncte ale unei sfere în 4D. Pentru interpolarea dintre doi cuaternioni, se poate considera cel mai scurt drum de pe sferă dintre aceştia (arc). Denumirea acestui procedeu este acela de interpolare sferică liniară.

4.8.3 Efecte simple de animaţie

Dacă se construiesc imagini color cu 8 biţi per pixel într-un frame-buffer de 640×512, numai o singură imagine conţine 320 Kb de informaţie. Transferarea unei noi imagini înspre frame buffer la fiecare 1/30 parte dintr-o secundă necesită o rată de transfer de 9Mb informaţie per secundă, astfel încât realizarea animaţiei este practic imposibilă pentru majoritatea calculatoarelor mici. Pentru evitarea stocării unor asemenea cantităţi de informaţii se recurge la metoda tabelului de celule-culori (look-up tables sau lut animation) şi compunerea imaginilor prin utilizarea tabelului de culori. De exemplu, scurgerea printr-o conductă poate fi simulată prin ciclarea unei secvenţe de culori din lut. Schimbările bruşte de culoare pot fi evitate prin schimbarea gradată a indicelui din lut în mai multe imagini intermediare, de la culoarea iniţială, la cea finală.

În cazul animaţiei unor obiecte de dimensiuni mici pe un fond fix se utilizează tehnica sprite-urilor. Un sprite este o regiune dreptunghiulară mică care este suprapusă peste frame-buffer la nivel video. Locaţia sprite-ului la fiecare moment în frame buffer este specificată în anumiţi regiştrii, modificarea valorilor acestora cauzând deplasarea sprite-ului. Sprite-ul poate ascunde valori din frame-buffer sau poate fi ascuns de acestea. Se utilizează la implementarea cursoarelor în frame-buffer şi generarea animaţiei prin mutarea sprite-ului deasupra unei imagini de fond. Mai multe sprite-uri presupun definirea unei liste de priorităţi (cine pe cine acoperă).

O aplicaţie a sprite-urilor, extrem de populară, este cea a jocurilor video, în care animaţia constă aproape în întregime din sprite-uri care se mişcă pe un fundal fix. Deoarece locaţia şi mărimea fiecărui sprite este stocat în regiştrii, este uşor de verificat coliziunea dintre sprite-uri.

Traiectoriile obiectelor pot fi generate prin urmărirea unui film real. O asemenea tehnică este rotoscope: se realizează un film în care oameni sau animale interpretează roluri din scenariul animaţiei, apoi animatorii, pe fondul existent, înlocuiesc personajele reale cu echivalentul lor desenat. Mişcarea personajelor animate poate fi realizată urmărind mişcarea personajelor pe care le înlocuiesc. Această tehnică produce o animaţie cu mişcări realistice. O altă tehnică este cea care asociază o serie de indicatori punctelor cheie ale corpului unei persoane. De exemplu, mici luminiţe pot fi ataşate punctelor cheie din arhitectura umană şi poziţiile acestor luminiţe sunt înregistrate din direcţii diferite pentru a furniza o poziţie 3D pentru fiecare punct cheie în orice moment (tehnica marionetei grafice).

4.8.4 Limbaje de animaţie

Se clasifică în trei mari categorii:

1. limbaje bazate pe notaţii tip listă liniară,
2. limbaje cu scop general care înglobează şi directive de animaţie,
3. limbaje grafice.

Limbaje bazate pe notaţii tip listă liniară

Fiecare eveniment din animaţie este descris printr-un număr de imagine iniţială şi unul de imagine finală, precum şi o acţiune care are loc. Acţiunile presupun anumiţi parametrii. De exemplu:

20, 34, C ROTATE "CASA",1,45

se poate traduce prin "roteşte obiectul CASA între imaginile 20 şi 34 în jurul axei 1 cu 45 de grade, determinând creşterea rotaţiei la fiecare imagine din tabelul C".

Scefo (SCEne FOrmat), pe lângă asemenea notaţii, include noţiunile de grup şi ierarhie a obiectelor. Un fişier în Scefo descrie numai animaţia, obiectele presupunându-se a fi descrise anterior.

Limbaje generale ce includ directive de animaţie

Avantajul faţă de limbajele mai sus-menţionate este abilitatea de generare procedurală a obiectelor şi a animaţiei. Valorile variabilelor limbajului pot fi utilizate ca parametrii pentru rutinele ce generează animaţie, astfel încât asemenea limbaje sunt indicate pentru simulări.

ASAS este un exemplu de asemenea limbaj, construit pe baza limbajului LISP şi ale cărui entităţi primitive includ vectori, culori, poligoane, colecţii de poligoane (solide), colecţii de obiecte (grupuri), puncte de vedere, lumini, volum de vedere. Obiectele pot fi transformate geometric (exemple: deplasare sus, jos, stânga, dreapta, înainte, înapoi, mărire, micşorare). O asemenea transformare primeşte ca argument un obiect şi returnează o copie transformată a obiectului.

Limbaje grafice

Problema principală a limbajelor textuale este dificultatea animatorului de a-şi imagina rezultatul execuţiei unui şir de comenzi.

În locul unor descrieri explicite ale acţiunii, în limbajele grafice de animaţie, animatorul construieşte o imagine a acţiunii.

Un prim pas a fost realizat prin introducerea curbelor P. O curbă P este o reprezentare parametrică a mişcării (sau a unui alt atribut) a unui obiect sau ansamblu de obiecte din interiorul unei scene. Animatorul descrie mişcarea unui obiect prin specificarea grafică a coordonatelor sale ca funcţie de timp.

Limbajul de animaţie diagramatic DIAL presupune o descriere a acţiunilor sub formă de listă liniară şi delimitarea în timp a fiecărei acţiuni. În S-Dynamics, fiecare acţiune este descrisă ca un dreptunghi a cărui lăţime indică extinderea în timp. Fiecare acţiune poate fi detaliată prin "deschiderea" dreptunghiului. Fiecare acţiune poate fi astfel constituită dintr-o mulţime de alte acţiuni seriale sau paralele, fiecare putând fi la rândul ei "deschisă" pentru detaliere, inclusiv pentru indicarea grafică a dependenţei de timp a unui parametru al acţiunii.

4.8.5 Atenuarea efectelor datorate discretizării temporale

Fenomenul de clipire a imaginii, ce pare în cazul unei mişcări foarte rapide temporal aliasing), poate fi parţial rezolvat prin creşterea rezoluţiei temporale.

Imaginile consecutive sunt similare, aşa cum în 2D o linie de scan nu diferă prea mult de următoarea. Se poate ţine seama, pentru simplificare, de o anumită coerenţă în construirea imaginilor. Fiecare obiect descrie o regiune din 4D. De exemplu, o sferă care nu se mişcă descrie în 4D un cilindru. În mod similar, un cerc din 2D ce se deplasează dintr-un colţ al ecranului în cel opus descrie un cilindru în spaţiul 2D+timp. Aplicând divizarea hiperspaţiului 4D şi utilizarea unor tehnici de tip ray-tracing se poate economisi timp considerabil în construirea de imagini consecutive.

O altă tehnică pentru antialiasing temporal este animaţia pe câmpuri. O imagine video convenţională este trasată în două etape: prima dată liniile scan pare, apoi cele impare. Fiecare linie scan este retrasată, aproximativ, la fiecare 1/30 dintr-o secundă (presupunem că raza tubului de electroni parcurge ecranul de 60 de ori pe secundă). Dacă culorile pixelilor liniilor pare sunt calculate la timpul t, iar liniile impare la $t+60^{-1}(s)$, apoi imaginea este compusă într-o singură hartă de biţi, iar acest proces se repetă pentru fiecare imagine, atunci animaţia este realizată cu un efect de 60fps, chiar dacă fiecare line scan este reactualizată la un interval de 1/30 dintr-o secundă. Dezavantajul acestei tehnici constă în faptul că imaginile de pe ecran constau din linii ale unor imagini construite la timp diferiţi şi poate apărea un fenomen de clipire.

4.8.6 Metode de control a animaţiei

Controlul explicit este forma cea mai simplă de control al animaţiei. Animatorul descrie tot ceea ce se realizează în animaţie, specificând atât schimbări simple precum scalarea, translaţia, rotaţia, cât şi informaţiile despre imaginile cheie şi metodele de interpolare. Această interpolare poate fi dată explicit (prin descriere matematică) sau, într-un sistem interactiv, prin directa manipulare a unui mouse, joystick sau alt dispozitiv de intrare.

În cazul unui control procedural, elementele distincte ale unui model comunică în scopul determinării proprietăţilor lor. Acest tip de control se utilizează pentru generarea mişcărilor greu de specificat prin control explicit.

Anumite obiecte din lumea fizică se mişcă descriind linii drepte, dar mişcările majorităţi obiectelor sunt determinate de alte obiecte cu care vin în contact, astfel încât mişcările rezultate nu mai sunt liniare. Mişcările acestora pot fi descrise prin constrângeri. Animaţia se poate realiza doar ţinând seama de aceste constrângeri, prin control bazat pe constrângeri. Se generează părţile unui ansamblu (linii, cercuri, etc), apoi se definesc constrângerile de tip punct ("linia are un capăt fix"), legătură ("linii unite la un capăt"), unghiuri ("linii paralele"). Mişcarea ansamblului se realizează respectând regulile de constrângere.

Utilizarea actorilor este o formă înaltă de control procedural. Un actor, în

animaţie, este un program mic, invocat odată pentru o imagine, pentru determinarea caracteristicilor unui obiect (actorul este un obiect în sensul programării orientate obiect). Un actor, poate trimite mesaje altor actori pentru a controla comportarea lor.

C a p i t o l u l 5

M e t o d a d r u m u l u i o p t i c

Metoda drumului optic (ray-tracing) este o metodă de generare a imaginilor des utilizată în practică deoarece răspunde dezideratului creării de imagini cu un înalt grad de realism.

5.1 Definiţii

Ray-tracing-ul este o tehnică pentru sintetizarea de imagini, adică crearea unei imagini bidimensionale a unei scene tridimensionale.

În cele ce urmează, se va va folosi termenul de pixel pentru a desemna unul din următoarele trei concepte:

1. o regiune "mică" (atomică) de pe monitor;
2. o locaţie adresabilă din frame-buffer;
3. o regiune "mică" din planul imaginii în lumea virtuală 3D.

5.2 Principiul de bază

În optică, proiecţia printr-o lentilă, presupune trasarea grafică a drumului unei raze de lumină de la sursă, prin lentilă, până la planul imaginii. Acest proces poartă denumirea de ray-tracing şi a fost considerat un bun model pentru crearea de imagini sintetice pe calculator (anii '80). La data abordării acestei metode posibilităţile de calcul fiind scăzute, ideea a fost abandonată, însă s-a revenit la ea în ultimii 15-20 de ani.

5.2.1 Modelul camerei obscure cu orificiu punctual

Ideea metodei provine de la principiul camerei obscure cu orificiu punctual (figura 5.1.a). Un film fotografic este plasat în spatele cutiei. Există un singur orificiu pe faţa opusă, acoperit cu un capac opac (obturator). Pentru crearea

imaginii, capacul opac este îndepărtat, iar lumina ce intră prin orificiu impresionează filmul fotografic. Orificiul permite doar unui număr mic de raze de lumină să pătrundă de la scenă la film. Teoretic, fiecare punct al filmului poate recepta lumina numai de-a lungul liniei ce uneşte acel punct şi orificiul (figura 5.1.b). Cu cât orificiul se măreşte, fiecare punct al filmului recepţionează mai multe raze, astfel încât imaginea devine mai strălucitoare şi mai neclară.

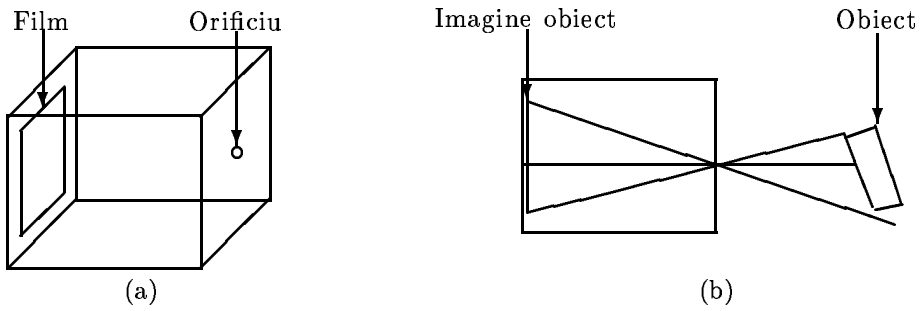


Figura 5.1: (a) Modelul camerei obscure (b) Orificiul permite numai anumitor raze să lovească filmul

5.2.2 Modelul modificat al camerei obscure

Din motive ce ţin de comoditatea programării, modelul utilizat în grafică, mută planul filmului în faţa orificiului şi redenumeşte orificiul ochi. Singurele obiecte pe care "ochiul" le percepe direct sunt cuprinse între "pereţii" determinaţi de liniile care unesc ochiul cu colţurile ecranului (figura 5.2.a).

Prin convenţie, doar obiectele din faţa ecranului pot fi "văzute". "Lumea" vizibilă este un trunchi de piramidă infinit (vezi volumul de vedere corespunzător unei proiecţii perspective, capitolul 2). Feţele laterale ale trunchiului de piramidă se numesc plane de tăiere (clipping planes).

5.2.3 Pixeli şi raze

Generarea unei imagini presupune, în esenţă, precizarea culorii fiecărui pixel al acesteia. Fiecare pixel al ecranului corespunde unei zone din modelul camerei obscure (figura 5.2.b).

Zona din planul imagine corespunzătoare unui "pixel" absoarbe un număr de raze de lumină, astfel încât culoarea pixelului este o combinaţie a culorii razelor care îl intersectează (care îl lovesc). Este posibil ca puncte diferite ale zonei care reprezintă pixelul să fie diferit iluminate. În acest caz problema culorii asociate pixelului nu este simplă. Soluţia în general adoptată constă în determinarea unei medii a culorii razelor care lovesc anumite puncte semnificative ale zonei care reprezintă pixelul.

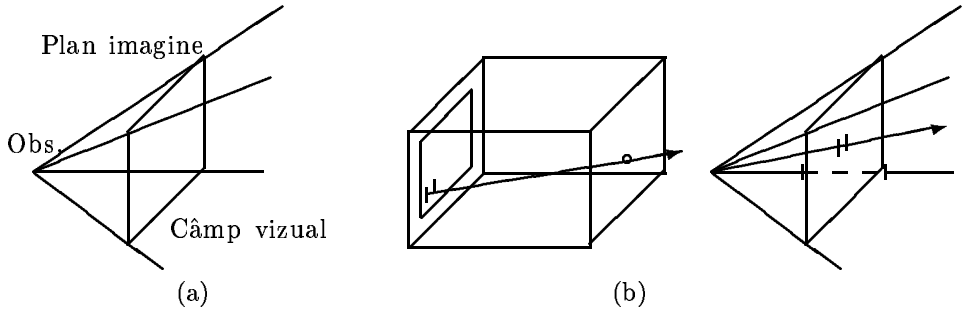


Figura 5.2: (a) Câmp vizual (b) Corespondenţa directă dintre un pixel de pe ecran şi o regiune de pe filmul fotografic din modelul camerei obscure

5.3 Trasarea razelor

5.3.1 Trasare directă

Trasarea directă (forward ray-tracing) presupune urmărirea razelor de la o sursă, la imagine şi apoi la ochi. Apar două probleme:

1. care raze lovesc un pixel?
2. ce culoare produce o asemenea rază (care este culoarea unei raze de lumină)?

Raza de lumină este calea (linie dreaptă) parcursă de o particulă luminoasă (foton) purtătoare de energie. Culoarea unui foton este dependentă de energia fotonului (exprimată prin frecvenţă sau lungime de undă).

Observaţie. Lumina se poate acumula aditiv (în ochi sau pe peliculă) – exemplu: culorile modelului RGB (capitolul 4).

La colorarea unui pixel din planul imaginii contribuie acele raze (fotoni) care, pornind de la o sursă de lumină:

(a) ajung direct în planul imaginii;
(b) ajung în planul imaginii după ce s-au reflectat din unul sau mai multe obiecte (energia scăzând, deoarece o parte este absorbită de obiecte).

5.3.2 Trasare inversă

Dezavantajul trasării directe, de la sursa de lumină înspre imagine, constă în numărul mare de raze de lumină pornite din sursă (surse) care nu contribuie la formarea imaginii pentru că în direcţia lor de propagare nu lovesc planul imaginii şi "ochiul", sau ajung la ochi cu intensitate prea redusă pentru a fi percepute.

O soluţie la această problemă este inversarea sensului de parcurgere a razelor luminoase (backward ray-tracing). În acest context, principalul

obiectiv constă în determinarea acelor fotoni care contribuie efectiv la formarea imaginii.

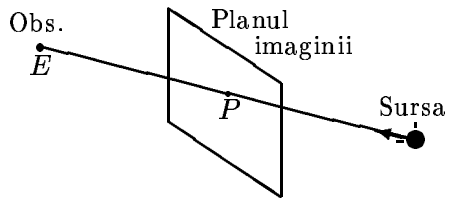


Figura 5.3: Foton ce trece printr-un punct al imaginii

Fotonii căutaţi sunt cei ce lovesc planul imaginii şi trec prin ochi (figura 5.3). Dacă un foton a lovit punctul P din planul imaginii şi a trecut prin ochiul desemnat prin punctul E al spaţiului, el a "călătorit" de-a lunul dreptei EP.

Presupunând că razele ce trec prin imagine au fost depistate, rămâne doar problema determinării culorii obiectelor sau surselor de lumină care se găsesc la celălalt capăt al razelor.

Observaţie. Când se vorbeşte de ray-tracing, în majoritatea cazurilor, se face referire la trasarea inversă, adică de la ochi, prin punctele imaginii, până la primele obiecte întâlnite.

5.3.3 Combinarea razelor

Pentru a determina culoarea unui pixel, trebuie constituită o listă a tuturor razelor de lumină care pleacă din punctul de pe primul obiect intersectat de dreapta de la ochi la pixel, culoarea pixelului fiind o combinaţie a culorilor acestor raze. De exemplu, o rază de lumină roşie şi o rază verde care cad amândouă într-un punct al unui obiect, pot constitui împreună o singură nou rază galbenă ce pleacă din respectivul punct înspre ochi.

Razele de lumină pot fi clasificate astfel:

- raze directe, care duc lumina prin pixel la ochi;
- raze de iluminare sau umbre, care duc lumina de la sursă la un obiect;
- raze reflectate de obiecte;
- raze de transparenţă, care trec printr-un obiect transparent.

Lumina radiată de un punct de pe suprafaţa unui obiect se poate determina cunoscând lumina incidentă (iluminarea) şi modul în care suprafaţa transmite lumina într-o direcţie dată.

Determinarea iluminării unui punct presupune determinarea fotonilor care călătoresc de la fiecare sursă de lumină la respectivul punct. Astfel, se trasează segmentele de dreaptă care unesc punctul cu sursele de lumină. Dacă unul dintre aceste segmente intersectează un obiect opac, sursa de lumină corespunzătoare segmentului produce o umbră a obiectului opac (figura 5.4.a).

Culoarea luminii reflectate este dependentă de culoarea obiectelor şi de culoarea luminii de la sursă (figura 5.4.b).

Razele transmise printr-un obiect transparent sunt rezultatul refracţiei. Raza ce uneşte sursa de lumină şi ochiul poate fi deviată (de la linia dreaptă) la trecerea printr-un asemenea obiect (figura 5.4.c).

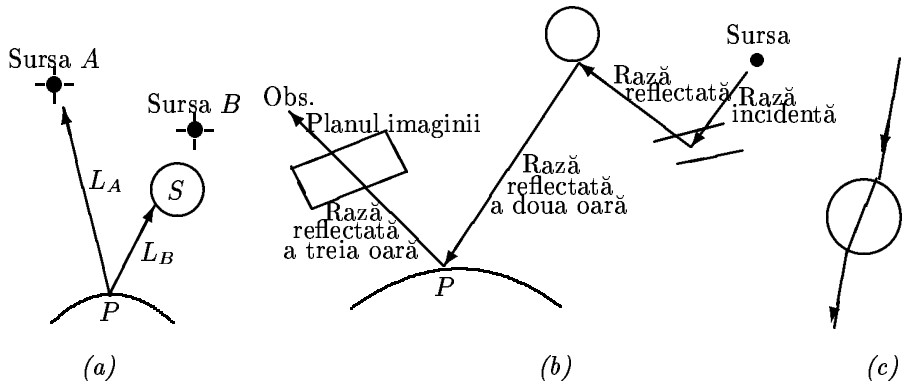


Figura 5.4: (a) Determinarea iluminării unui punct (b) Raze reflectate (c) Raze de transparenţă

5.4 Recursivitatea vizibilităţii

Culoarea luminii radiate de un punct de pe suprafaţa unui obiect (într-o anumită direcţie) este, în esenţă, funcţie de combinaţia luminii ce provine de la:

- surse de lumină,
- alte obiecte ce reflectă lumina,
- lumina transmisă prin obiecte.

Pentru a afla culoarea luminii reflectate şi a celei transmise trebuie determinate obiectele de la care provine această lumină. Pe de altă parte, culoarea luminii care pleacă de la oricare dintre aceste obiecte se determină în acelaşi mod, de unde rezultă caracterul recursiv al algoritmului.

Modul în care se comportă lumina pe o anumită suprafaţă este tratat de ceea ce se numeşte fizica suprafeţelor. Spre exemplificare, se consideră în figura 5.5 (a), patru obiecte, notate O_1, O_2, O_3, O_4: obiectele O_1 şi O_4 sunt opace, iar O_1 şi O_3 parţial transparente şi parţial reflective. L_A şi L_B sunt surse de lumină. Arborele razelor ce contribuie la formarea razei E este dat în figura 5.5 (b).

Se observă că razele R_2, R_3 şi T_3 părăsesc scena ("lovesc" lumea înconjurătoare) şi deci nu participă la formarea culorii luminii care pleacă din O_2 spre O_1 respectiv din O_3 spre O_1.

Urmărirea unei ramuri a arborelui este stopată atunci când:

1. raza intersectează o sursă de lumină;
2. raza părăseşte scena;

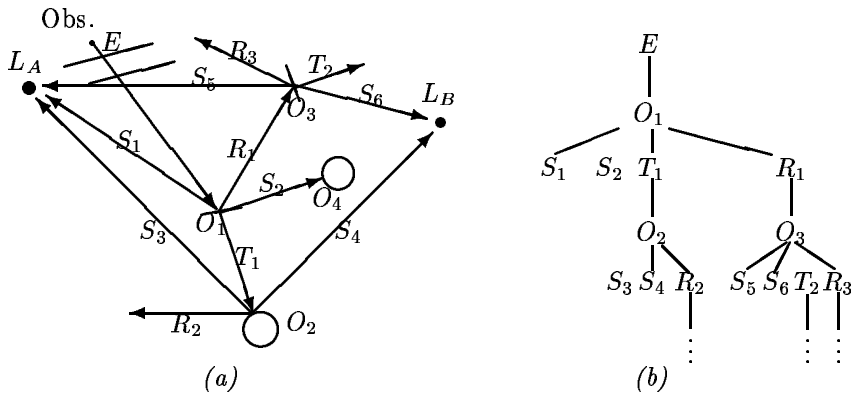


Figura 5.5: (a) Contribuţia razelor la culoarea punctului intersectat de raza E (b) Arborele razelor

3. contribuţia razei este nesemnificativă.

Dacă E ar fi singura rază care trece prin pixelul respectiv, atunci ea ar contribui cu 100% la culoarea lui. În cazul dat, contribuţia razelor R_1 şi T_1 este dată de gradul (procentual) de reflectivitate şi de transparenţă a obiectului O_1. Presupunem că O_1 este 20% reflectiv şi 40% transparent, iar O_2, 30% reflectiv. Atunci contribuţia lui R_2 la T_1 este de 30%, iar contribuţia lui R_2 la E este de 30%·40%=12%. Cu cât adâncimea în arborele de raze este mai mare, cu atât contribuţia razelor la formarea razei finale (rădăcină) este mai mică. În practică, se stabileşte un prag de "participare" care are rolul de a opri recursivitatea. Tehnica rezultată poartă numele de controlul adaptiv al adâncimii arborelui de ray-tracing.

5.5 Atenuarea efectelor datorate discretizării imaginii

Imaginile sintetizate cu calculatorul (imagini discrete) diferă în mod esenţial de imaginile reale (imagini continue). Imaginea preluată prin intermediul unui sistem optic şi electronic (cameră de luat vederi) este transformată într-un semnal electric continuu şi trebuie discretizată pentru a putea fi stocată şi prelucrată în memoria calculatorului.

Această discretizare are un dublu aspect:

- eşantionarea spaţială (spatial sampling);
- cuantizarea culorii (color quantization).

Problemele de eşantionare şi cuantizare sunt tratate de teoria procesării semnalelor (signal procesing).

Problema principală este cea a unei eşantionări şi cuantizări adecvate. O eşantionare insuficient de deasă sau o cuantizare insuficient de precisă provoacă

fenomenul de aliasing (două semnale diferite care apar ca fiind identice). Tehnicile de eliminare a acestui fenomen poartă denumirea de anti-aliasing.

5.5.1 Efecte ale eşantionării spaţiale

Spatial aliasing apare datorită rezoluţiei insuficiente a reţelei de eşantionare. Fie exemplul poligonului din figura 5.6 (a). Mărind rezoluţia reţelei se poate reduce impresia de "scară" a muchiilor. Problema este rezolvabilă în acest fel doar în mod relativ: dacă se măreşte dimensiunea unui ochi al reţelei, deci şi a imaginii, impresia de scară reapare.

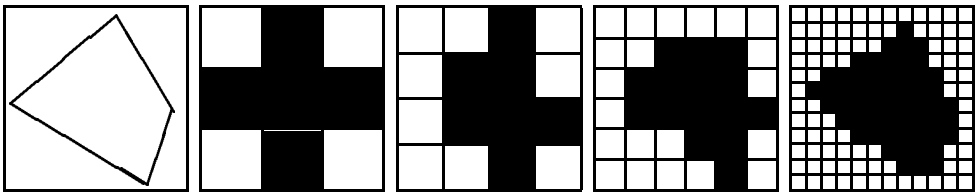


Figura 5.6: (a) Poligonul real (b) Trasare pe o reţea 3×3 (c) Reţea 4×4 (c) Reţea 6×6 (d) Reţea 12×12

Un alt aspect al problemei este acela că obiectele prea mici relative la dimensiunea unui ochi (al reţelei) "scapă" reţelei de eşantionare (figura 5.7). Oricât de multe raze (oricât de dese) de eşantionare se folosesc, ar putea exista obiecte mai mici decât distanţa dintre raze. Pe de altă parte, s-ar putea afirma că, dacă un obiect este atât de mic încât nu poate fi depistat de razele de eşantionare, atunci nu are importanţă dacă apare în imagine sau nu. Un contraexemplu este dat în secţiunea următoare (exemplul 2).

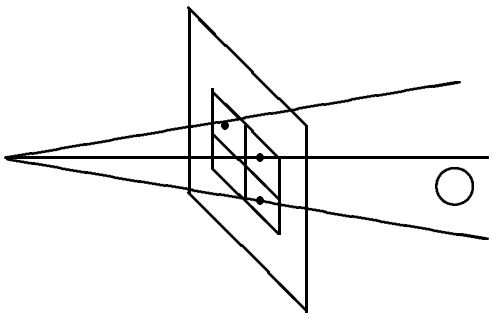


Figura 5.7: Obiect prea mic pentru a fi intersectat de razele de eşantionare

5.5.2 Efecte ale eşantionării temporale

Aparent, pentru a face animaţie pe calculator este suficientă afişarea într-o succesiune destul de rapidă a unor imagini fixe de bună calitate. Această idee este falsă datorită fenomenului de aliasing temporal.

Exemplul 1. Se urmăreşte rotirea spiţei unei roţi (figura 5.8). La o viteză de rotaţie mare, adică la un anumit raport cu rata "expunerii", spiţa pare să se învârtească în sens invers, sau chiar să stea pe loc. Fenomenul apare deoarece mişcarea este prea rapidă pentru a fi înregistrată corect.

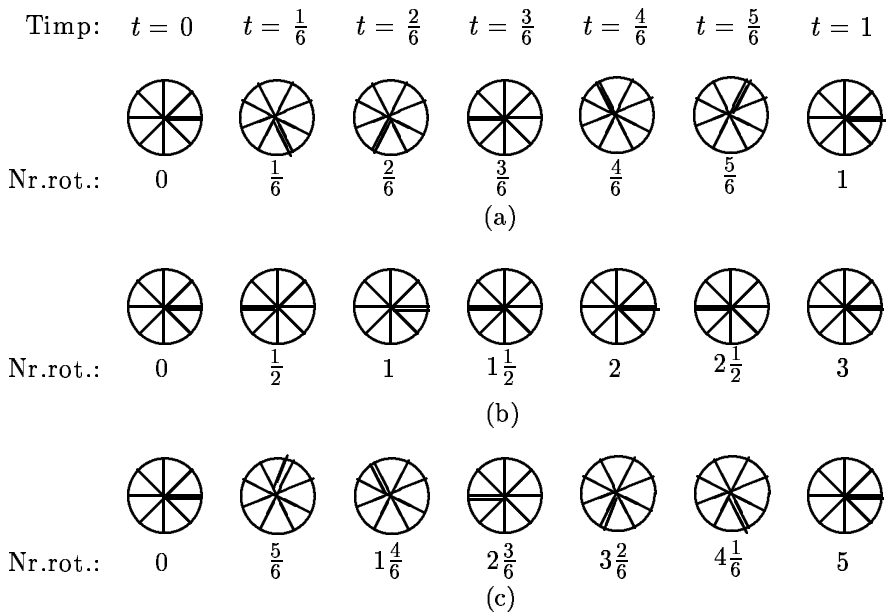


Figura 5.8: Animaţie: (a) Eşantionare corectă (b) Imposibilitate de precizare a direcţiei de înaintare (c) Eşantionare incorectă: impresia deplasării în sens invers

Exemplul 2. Presupunem că într-o scenă există un obiect prea mic (relativ la rezoluţia spaţială) care în secvenţe succesive îşi schimbă poziţia. Se poate întâmpla ca, în anumite secvenţe, obiectul să fie "lovit" de o rază şi să apară în imagine, iar în alte secvenţe, să fie încadrat de raze, deci "omis", ceea ce e cu atât mai deranjant cu cât contrastul cu fondul (background) este mai mare.

Exemplul 3. O altă problemă este legată de deplasarea muchiilor. În figura 5.9 se simulează mişcarea lentă a unui poligon cu o latură paralelă cu una din axele de coordonate. Un nou rând de pixeli este aprins doar în momentul în care se acoperă centrele pixelilor. Astfel, latura discretizată a poligonului pare să facă salturi de la o linie la următoarea.

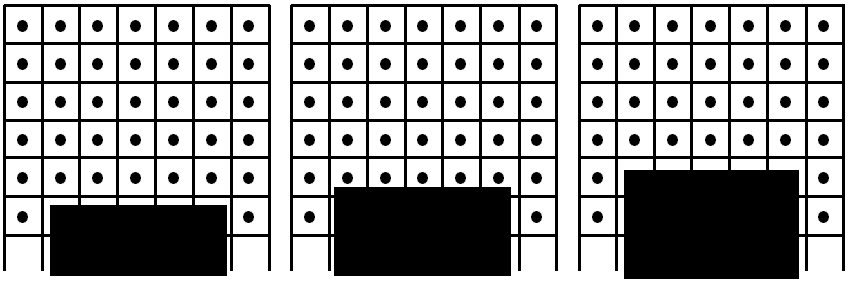


Figura 5.9: Mişcarea lentă a unei laturi a unui poligon

În esenţă problema aliasing-ului provine din natura discretă a reprezentării (în sistemul grafic) a unui fenomen continuu.

Tehnicile care rezolvă aliasing-ul temporal crează imagini statice neclare în zonele în care apar mişcări rapide ale obiectelor, de aceea aceste tehnici sunt considerate ca incluzând neclarităţi datorate mişcării (imagini "mişcate", motion blur).

5.5.3 Supra-eşantionarea

O soluţie posibilă, simplă din punct de vedere conceptual, pentru problemele de mai sus, constă în utilizarea mai multor raze care trec prin acelaşi pixel, culoarea finală fiind o medie a culorilor acestor raze. Se aleg anumite puncte de reprezentare a pixelului prin care trec razele, de exemplu cele din figura 5.10 (a).

Figura 5.10: (a) Puncte fixe (b) Puncte aleatoare — desemnate pentru a reprezenta un pixel, puncte prin care trec razele vizuale

Observaţii:

1. Supra-eşantionarea (supersampling) nu rezolvă problema, ci doar reduce efectele ei.
2. Supra-eşantionarea este extrem de costisitoare: dacă factorul de multiplicare al razelor pentru un pixel este n, atunci timpul de trasare creşte de cel puţin n ori.

Pentru a evita "scăparea" unor obiecte printre razele ce trec prin poziţii fixe, se poate utiliza o distribuţie aleatoare a razelor (dar uniformă) peste regiunea corespunzătoare pixelului, prin supra-eşantionare stohastică (figura 5.10.b).

5.5.4 Supra-eşantionarea adaptivă

În ideea de a determina culoarea predominantă în întreaga regiune care reprezintă un pixel se utilizează un număr variabil de raze ce intersectează pixelul (în locul unui număr fix, ca în cazul din secţiunea anterioară).

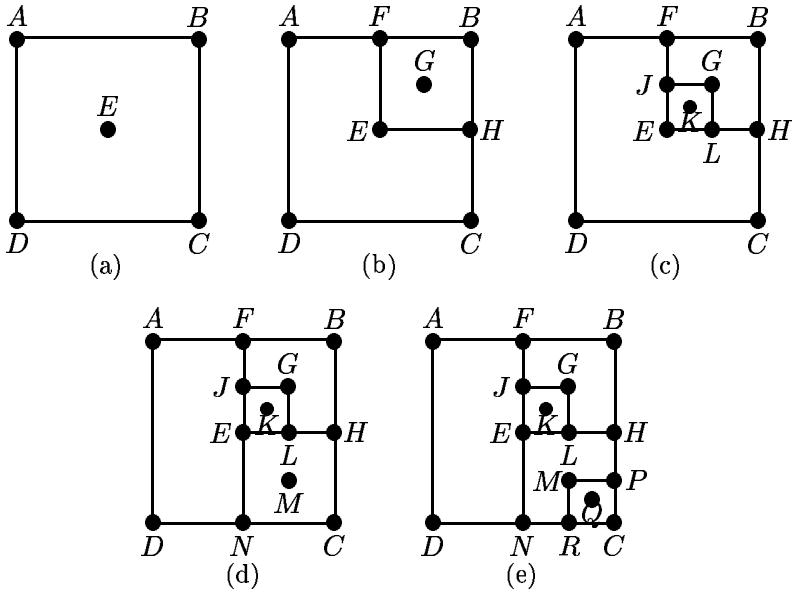


Figura 5.11: Paşi într-un algoritm de super-eşantionare adaptivă

Algoritmul de supra-eşantionare adaptivă (adaptive supersampling) porneşte de la o reprezentare prin cinci raze (figura 5.11.a). Dacă fiecare din razele de start au culori apropiate, se va utiliza media lor. Dacă razele de start diferă în culoare, atunci se subdivide pixelul în regiuni mai mici şi se repetă pasul pentru fiecare nouă regiune. Astfel, multiplicarea razelor nu se face pe regiuni de culoare constantă, ci adaptiv, acolo unde apar variaţii. Se consideră că, în figura 5.11 (a), razele care trec prin A, D şi E au culori similare, ceea ce nu poate fi afirmat şi despre B, C şi E. Regiunea mărginită de B şi E este studiată mai în amănunţime. Se trasează noi raze prin F, G, H (figura 5.11.b). Se compară cele cinci culori. Presupunem că razele ce trec prin F, B, H şi G au culori similare, mai puţin E cu G. Regiunea mărginită de G şi E este studiată mai amănunţit. Se trasează trei noi raze, prin J, K şi L din figura 5.11 (c). Presupunem că toate razele noii regiuni au culori similare. Se revine la regiunea iniţială, pentru a studia în amănunţime perechea de raze ce trec prin C şi E. Se trasează două noi raze, prin M şi N din figura 5.11 (d). Presupunem că razele

specifice noii regiuni au culori similare, mai puţin cele care trec prin C şi M. Regiunea mărginită de aceste două puncte este analizată în amănunţime (figura 5.11.e). Presupunem că în noua subregiune nu există variaţii semnificative de culoare. Culoarea finală este determinată printr-o medie ponderată:

$$\frac{1}{4}\Bigg(\frac{A+E}{2}+\frac{D+E}{2}+\frac{1}{4}\Bigg[\frac{F+G}{2}+\frac{B+G}{2}+\frac{H+G}{2}+$$

$$+\frac{1}{4}\left\{\frac{J+K}{2}+\frac{G+K}{2}+\frac{L+K}{2}+\frac{E+K}{2}\right\}\Bigg]+\frac{1}{4}\Bigg[\frac{E+M}{2}+$$

$$+\frac{H+M}{2}+\frac{N+M}{2}+\frac{1}{4}\left\{\frac{M+Q}{2}+\frac{P+Q}{2}+\frac{C+Q}{2}+\frac{R+Q}{2}\right\}\Bigg]\Bigg)$$

Observaţie. Problema obiectelor mici nu este rezolvată prin supra-eşantionare. Acestea pot "scăpa" printre cele cinci raze iniţiale (datorită faptului că sunt în poziţii fixe).

5.6 Algoritmi de intersecţie

Partea principală a oricărui pachet de programe pentru ray-tracing ("inima") o constituie un set de rutine de intersecţie rază–obiect.

În funcţie de tipul razei, diverse informaţii sunt semnificative:

- pentru o rază trasată de la observator, rutina de intersecţie trebuie să returneze cel mai apropiat punct de intersecţie şi normala la suprafaţă în punctul respectiv;
- pentru o rază de tip senzor de umbră (shadow feeler) important este dacă punctul de intersecţie e mai apropiat decât sursa de lumină (caz în care punctul respectiv este în umbră);
- pentru o rază trasată spre un volum maximal (vezi capitolul 4) este suficientă informaţia de intersecţie/nonintersecţie (astfel de raze se utilizează în tehnici de accelerare).

O altă informaţie utilă este poziţia punctului de intersecţie relativ la o "hartă" ce urmează să fie mapată pe suprafaţa pe care se află punctul (informaţie utilă în cazul suprapunerii unei texturi peste obiect).

În principiu, algoritmii de intersecţie sunt dependenţi de forma obiectului de intersecţie. O clasă de obiecte pentru care rutinele de intersecţie sunt ceva mai simple o reprezintă obiectele a căror ecuaţie este o cuadrică. Exemple particulare de astfel de obiecte sunt sfera, planul, cilindrul, conul, etc. Pentru anumite cazuri particulare (sferă, plan) se pot proiecta rutine mai simple decât rutina generală de intersecţie cu o cuadrică.

Punctele de intersecţie ale unei raze cu un obiect pot fi multiple, dar numărul lor poate fi redus (pot fi luate în considerare doar unele dintre ele) de la bun început. De exemplu, raza fiind o semidreaptă nu intersectează decât suprafeţele care se află pe partea "activă" a razei, deci nu în "spatele" originii razei.

5.6.1 Intersecţia cu o sferă

Soluţia algebrică (brută)

Fie raza definită prin

$$\begin{aligned} \text{origine:} \quad & R_0 \equiv (X_0, Y_0, Z_0), \\ \text{direcţie:} \quad & R_d \equiv (X_d, Y_d, Z_d), \end{aligned}$$

unde $X_d^2 + Y_d^2 + Z_d^2 = 1$ (vector normalizat), astfel încât ecuaţia parametrică (explicită) a razei este

$$R(t) = R_0 + t \cdot R_d, \quad t > 0. \tag{5.1}$$

Observaţii:

1. Punctele pentru care $t < 0$ se află în spatele originii.
2. Punctul obţinut pentru $t = 0$ nu este luat în considerare din motive ce ţin de precizia calculelor.
3. Vectorul direcţional nu trebuie să fie în mod necesar normalizat, dar acest lucru este recomandat pentru ca lungimile de-a lungul razei să fie exprimate în unităţile corespunzătoare sistemului de coordonate al lumii reale).

Sfera se defineşte astfel:

$$\begin{aligned} \text{centrul:} \quad & S_c \equiv (X_c, Y_c, Y_c), \\ \text{raza:} \quad & S_r, \\ \text{suprafaţa:} \quad & \{(X_s, Y_s, Z_s) \mid (X_s - X_c)^2 + (Y_s - Y_c)^2 + (Z_s - Z_c)^2 = S_r^2\} \end{aligned} \tag{5.2}$$

Observaţie: Ecuaţia sferei este implicită, adică prin ecuaţia dată nu se pot genera direct punctele de pe suprafaţa sferei, dar se poate testa dacă un punct este pe sferă, în interior sau în exterior, prin înlocuirea coordonatele sale în ecuaţia sferei.

Pentru a rezolva problema intersecţiei rază–sferă se înlocuiesc în ecuaţia sferei (5.2) coordonatele X_s, Y_s, Z_s cu cele rezultate din ecuaţia parametrică a razei (5.1). Punctele de intersecţie sunt notate generic cu (X, Y, Z). Atunci

$$\begin{cases} X = X_0 + t \cdot X_d, \\ Y = Y_0 + t \cdot Y_d, \\ Z = Z_0 + t \cdot Z_d. \end{cases} \tag{5.3}$$

În plus are loc

$$(X_0 + t \cdot X_d - X_c)^2 + (Y_0 + t \cdot Y_d - Y_c)^2 + (Z_0 + t \cdot Z_d - Z_c)^2 = S_r^2. \tag{5.4}$$

Rezultă o ecuaţie de grad doi în t:

$$At^2 + Bt + C = 0, \tag{5.5}$$

unde

$$\begin{cases} A = X_d^2 + Y_d^2 + Z_d^2 = 1, \\ B = 2[X_d(X_0 - X_c) + Y_d(Y_0 - Y_c) + Z_d(Z_0 - Z_c)], \\ C = (X_0 - X_c)^2 + (Y_0 - Y_c)^2 + (Z_0 - Z_c)^2 - S_r^2. \end{cases}$$

Valorile S_r^2 şi C nu depind de direcţia razei şi pot fi calculate o singură dată pentru obiectul–sferă dat. Ecuaţia în t are soluţiile:

$$t_0 = \frac{-B - \sqrt{B^2 - 4C}}{2}, \qquad t_1 = \frac{-B + \sqrt{B^2 - 4C}}{2}. \tag{5.6}$$

Observaţii:

1. Dacă discriminantul este negativ (ecuaţia nu are soluţii reale), concluzia este că raza nu intersectează sfera.
2. Întrucât condiţia iniţială este $t > 0$, se examinează t_0 şi t_1: rădăcina pozitivă mai mică reprezintă cel mai apropiat punct de intersecţie.
3. Se pot omite anumite calcule: dacă $t_0 > 0$ atunci nu se mai calculează t_1 (pentru că este mai mare decât t_0, deci nu este punctul căutat).

Cunoscută fiind rădăcina t cea mai mică, corespunzătoare celui mai apropiat punct de intersecţie, componentele acestui punct de intersecţie sunt calculate din ecuaţia dreptei:

$$r_i \equiv (x_i, y_i, z_i) = (X_0 + t \cdot X_d, Y_0 + t \cdot Y_d, Z_0 + t \cdot Z_d), \tag{5.7}$$

iar vectorul unitate (versor) normal la suprafaţa sferei în respectivul punct este

$$r_n = \left(\frac{x_i - X_c}{S_r}, \frac{y_i - Y_c}{S_r}, \frac{z_i - Z_c}{S_r} \right). \tag{5.8}$$

Dacă originea razei se află în interiorul sferei, atunci normala se negativează pentru a fi îndreptată în sens contrar razei. Din punct de vedere computaţional, se preferă să se calculeze o singură dată valoarea $1/S_r$, cantitate utilizată în trei multiplicări pentru determinarea normalei (o multiplicare este mai rapidă decât o diviziune).

Exemplu. Fie raza definită de $R_0 = (1, -2, -1)$, $R_d = (1, 2, 4)$ şi sfera $S_r = 3$, $S_c = (3, 0, 5)$. Punctul de intersecţie este $r - I = (1.816, -0.368, 2.269)$, iar normala la suprafaţă în respectivul punct, $r_n = (-0.395, -0.123, -0.910)$.

Considerând că se fac toate precalculele necesare (S_r^2, $1/S_r$), algoritmul de intersecţie decurge astfel:

Pas 1: se calculează A, B, C (8 adunări/scăderi şi 7 înmulţiri);

Pas 2: se calculează discriminantul ecuaţiei în t (1 scădere, 2 înmulţiri şi 1 comparaţie);

Pas 3: se calculează t_0 şi se decide dacă soluţia căutată, adică $t_0 > 0$ (1 scădere, 1 înmulţire, 1 radical şi 1 comparaţie);

Pas 4: dacă $t_0 < 0$, atunci se calculează t_1 şi se decide dacă e soluţia căutată, adică $t_1 > 0$ (1 scădere, 1 înmulţire şi 1 comparaţie);

Pas 5: se calculează componentele punctului de intersecţie (3 adunări, 3 înmulţiri);

Pas 6: se calculează normala la suprafaţă în punctul de intersecţie (3 scăderi, 3 înmulţiri).

Pentru cazul cel mai defavorabil se efectuaeză 17 adunări/scăderi, 17 înmulţiri, 1 radical, 3 comparaţii. Numărul de operaţii se poate reduce prin abordarea soluţiei geometrice.

Soluţia geometrică

O asemenea soluţie este utilizată pentru a accelera rutina de intersecţie.

Observaţii:

1. Pentru reducerea timpului de calcul trebuie evitate operaţiile tip radical, împărţire, înmulţire.
2. De multe ori se pot elimina anumite calcule dacă se fac teste impuse de situaţia concretă.
3. Studiind aspectul geometric se poate stabili, de exemplu, că raza se îndepărtează de sferă, deci, evident, n-o intersectează.

O strategie îmbunătăţită pe baza acestor observaţii este următoarea:

Pas 1: stabileşte dacă originea razei este în afara sferei.

Pas 2: determină punctul de pe rază cel mai apropiat de centrul sferei.

Pas 3: dacă raza este exterioară şi se îndepărtează de sferă, atunci nu intersectează sfera.

Pas 4: altfel, determină pătratul distanţei minime de la punctele razei la suprafaţa sferei.

Pas 5: dacă valoarea este negativă, raza nu intersectează sfera.

Pas 6: altfel se determină distanţa rază – suprafaţă.

Pas 7: calculează punctele de intersecţie.

Pas 8: calculează normala în punctul de intersecţie.

Observaţie. Expresiile lui r_i şi r_n sunt evaluate funcţie de necesităţi.

Se consideră că raza este dată parametric (5.1), iar sfera, implicit (5.2). Dacă $OC = S_c - R_0$ este vectorul care uneşte originea razei (R_0) cu centrul sferei (S_c), atunci se determină dacă originea este interioară pe baza cantităţii

$$L_{2oc} = L_{oc}^2 = (S_c - R_c)^2. \tag{5.9}$$

Dacă $L_{2oc} < S_r^2$, originea razei este în interiorul sferei, iar dacă $L_{2oc} \geq S_r^2$, originea este pe sferă sau exterioară sferei, astfel încât raza poate să nu intersecteze sfera (figura 5.12.a,b). O rază care are originea pe sferă se consideră că nu intersectează sfera (prin convenţie).

Problema determinării distanţei minime de la rază la centrul sferei este echivalentă cu a găsi intersecţia razei cu planul perpendicular pe rază şi care trece prin S_c. Se notează cu t_{ca} produsul scalar

$$t_{ca} = (S_c - R_0) \cdot R_d. \tag{5.10}$$

Se observă că, dacă $t_{ca} < 0$, atunci centrul sferei este "în spatele" lui R_0 (figura 5.12.d). Distanţa minimă este D, unde $D^2 = L_{oc}^2 - t_{ca}^2$. Se determină cantitatea

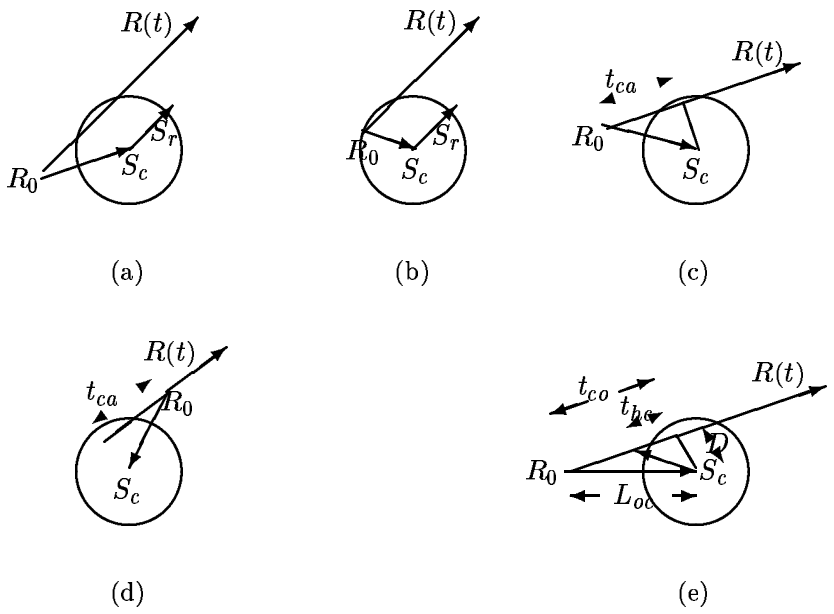


Figura 5.12: (a) Rază cu origine exterioară sferei (b) Rază cu origine interioară sferei (c) Două intersecţii (d) O singură intersecţie (e) Calculul distanţei D

$t_{2hc} = t_{hc}^2 = S_r^2 - D^2$, adică

$$t_{2hc} = S_r^2 - L_{2oc} - t_{ca}^2. \tag{5.11}$$

Dacă $t_{2hc} < 0$, raza nu intersectează sfera (figura 5.12.e). Altfel, valoarea parametrului t corespunzător intersecţiei este

$$t = \begin{cases} t_{ca} - \sqrt{t_{2hc}}, & \text{rază cu origine exterioară sferei,} \\ t_{ca} + \sqrt{t_{2hc}}, & \text{rază cu originea interioară sferei.} \end{cases} \tag{5.12}$$

Aceste relaţii pot fi interpretate astfel: dacă raza "loveşte" sfera şi nu este tangentă, atunci există două puncte de intersecţie. Astfel,

- când R_0 este exterior sferei, se consideră punctul cel mai apropiat de origine, adică cu t cel mai mic, drept prim punct de intersecţie (semn negativ în faţa discriminantului);
- când R_0 este interior sferei, cantitatea t_{ca} este negativă, astfel încât se alege t valoarea cea mai mare (semnul plus în faţa discriminantului pentru un $t > 0$).

Coordonatele punctului de intersecţie şi normala la suprafaţă se calculează ca în paragraful anterior, ecuaţiile (5.7), (5.8).

Algoritmul de intersecţie bazat pe aceste calcule constă în următorii paşi:
Pas 1: determină L_{2oc} (5 adunări/scăderi, 3 înmulţiri);

Pas 2: calculează t_{ca} (2 adunări, 3 înmulţiri);
Pas 3: test de exterioritate şi îndepărtare: $t_{ca} < 0$ şi $L_{2oc} \geq S_r^2$ (2 comparaţii);
Pas 4: determină t_{2hc} (2 adunări/scăderi, 1 înmulţire);
Pas 5: test dacă $t_{2hc} < 0$ (1 comparaţie);
Pas 6: calculează t corespunzător intersecţiei (1 adunare/scădere, 1 radical);
Pas 7: calculează coordonatele punctului de intersecţie (3 adunări, 3 înmulţiri);
Pas 8: calculează normala în punctul de intersecţie (3 adunări, 3 înmulţiri).
În cel mai defavorabil caz se efectuează 16 adunări/scăderi, 13 înmulţiri, 1 radical, 3 comparaţii. Se observă reducerea numărului de operaţii faţă de soluţia algebrică, fapt ce permite o procesare mai rapidă.

Probleme de precizie a calculelor

Efectuarea calculelor în virgulă flotantă poate fi asemuită cu mutarea unor grămezi de nisip – la fiecare mutare se pierd câteva grăunţe de nisip. În mod similar, la fiecare operaţie în virgulă flotantă se pierde ceva din precizia rezultatului.

În algorimul de ray-tracing se întâmplă frecvent ca originea unei raze să se afle chiar pe sferă. Pentru un asemenea punct $t = 0$ şi, deci, originea este ignorată ca punct de intersecţie. Practic, în urma calculelor se găseşte un punct de intersecţie care urmează să fie originea unei noi raze. Datorită impreciziei calculelor (dar şi datorită erorii de reprezentare) în virgulă flotantă, se poate întâmpla ca punctul de intersecţie să fie "sub" suprafaţa sferei (figura 5.13.a). Când se trasează o rază din acest punct spre o sursă de lumină, de exemplu, ea va întâlni suprafaţa sferei şi se poate trage concluzia, lipsită de sens, că punctul de pe sferă este umbrit chiar de sfera în cauză. Efectul practic este apariţia pe suprafaţa sferei a unor puncte în umbră, inexistente în realitate.

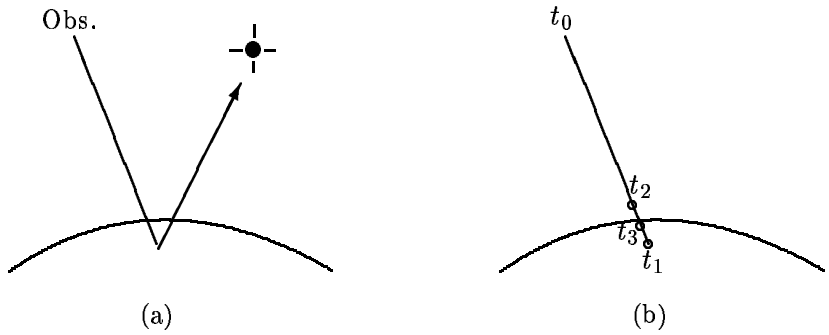


Figura 5.13: (a) Problemă la aproximarea intersecţiilor (b) Procesul iterativ de determinare a intersecţiilor

Această problemă poate interveni la intersecţia cu oricare alt tip de suprafaţă.

Soluţia 1. O soluţie de tratare a problemei cauzate de imprecizia calculelor

constă în asocierea unui indicator (flag) cu originea razei, indicator care să precizeze că originea e chiar pe suprafaţă, lucru posibil întrucât punctul respectiv se ştie că a fost obţinut ca urmare a unui algoritm de intersecţie, deci el trebuie să fie "pe" suprafaţă (chiar dacă coordonatele îl plasează doar "în apropierea" suprafeţei).

Apare însă o altă problemă: dacă corpul pe a cărui suprafaţă se află punctul este un "transmiţător" (raza pornind din acel punct trebuie să-l străbată), atunci trebuie efectuate anumite teste pentru a permite razei de refracţie să lovească cealaltă parte a corpului.

Soluţia 2. O soluţie simplă constă în a testa dacă t corespunzător intersecţiei se încadrează într-o anumită toleranţă: de exemplu, dacă $\text{abs}(t) < 0.00001$, atunci se poate considera că punctul obţinut utilizând t corespunde situaţiei în care originea este pe suprafaţă.

Trebuie avută însă în vedere scalarea toleranţei conform situaţiei concrete: dacă sferele reprezintă atomi, iar razele se exprimă în metri, 0.00001 metri ar fi mult mai mare decât diametrul oricărui atom!

Pe de altă parte, toleranţa poate fi relativă la raza sferei intersectate.

Soluţia 3. Rafinarea iterativă a soluţiilor poate fi o cale de rezolvare a problemei. De exemplu, fie t soluţia ecuaţiei de grad doi corespunzător intersecţiei. Este probabil ca înlocuind coordonatele punctului de intersecţie în ecuaţia implicită a sferei, să se obţină o indicaţie ca aceea de "punct în apropierea suprafeţei". Pornind din acest punct se caută o nouă intersecţie a razei ce porneşte din noul punct cu sfera (aceeaşi direcţie cu prima rază, dacă punctul de intersecţie este în afara sferei, direcţie inversă, altfel). Se determină noul t al intersecţiei. Procesul iterativ poate continua până când t se încadrează într-o toleranţă prestabilită (figura 5.13.b).

Soluţia 4. Se poate muta punctul de intersecţie obţinut din calcule, în afara sau în interiorul sferei, în funcţie de necesităţi. Mai precis, dacă din punctul de intersecţie găsit trebuie trasate noi raze, trebuie să ne asigurăm că el se află de partea "corectă" a suprafeţei (în afara sferei pentru raze de sesizare a umbrei şi pentru raze reflectate, respectiv în interior pentru raze refractate). Dacă punctul nu se află în partea dorită a suprafeţei el poate fi mutat de-a lungul normalei până se constată (prin teste: înlocuind coordonatele punctului în ecuaţia sferei şi testând semnul obţinut) că a ajuns în partea dorită.

Mapare inversă pe sferă

Pentru fiecare punct de intersecţie de pe suprafaţa sferei, textura poate varia. De exemplu, se doreşte suprapunerea unei hărţi peste o sferă. Culoarea fiecărui pixel (punct de intersecţie cu sfera) este preluată din hartă. Problema constă în a determina care puncte din hartă corespund fiecărui pixel în parte (coordonatele punctului de intersecţie se convertesc în longitudine şi latitudine pe hartă).

Intrarea algoritmului de mapare inversă pe sferă (spherical inverse mapping) o constituie normala S_n la suprafaţa sferei în punctul de intersecţie

r_i şi următoarea descriere a axelor sferei:

$$\begin{aligned} \text{axa polului:} \quad & S_p \equiv (X_p, Y_p, Z_p), \\ \text{axa ecuatorului:} \quad & S_e \equiv (X_e, Y_e, Z_e). \end{aligned} \tag{5.13}$$

Prin definiţie, $S_p \cdot S_e = 0$ (S_p este perpendicular pe S_e, S_p şi S_e sunt vectori unitari). Se consideră parametrul u ce variază de-a lungul ecuatorului, de la 0 la 1 (la poli $u = 0$). Parametrul v variază între 0 şi 1, de la polul sud la polul nord (de la $-S_p$ la S_p).

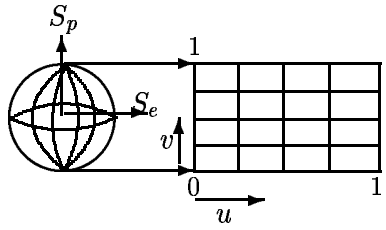


Figura 5.14: Maparea inversă pentru o sferă

Normala la suprafaţă S_n (versor) într-un punct de intersecţie are aceeaşi direcţie ca şi vectorul unitate cu originea în centrul sferei şi capătul în punctul de intersecţie (figura 5.14).

Latitudinea se calculează astfel:

$$\phi = \arccos(-S_n \cdot S_p), \qquad v = \phi/\pi. \tag{5.14}$$

Dacă $v = 0$ sau $v = 1$ longitudinea $u = 0$, altfel

$$\theta = \frac{\arccos((S_e \cdot S_n)/\sin(\phi))}{2\pi}, \quad u = \begin{cases} \theta, & (S_p \times S_e) \cdot S_n > 0, \\ 1 - \theta, & \text{altfel.} \end{cases} \tag{5.15}$$

Produsul vectorial $S_p \times S_e$ poate fi precalculat. Efectul testului este acela de a determina de ce parte a lui S_e (meridianul Greenwich, în cazul hărţii) se află punctul de intersecţie.

5.6.2 Intersecţia cu un plan

Fie raza definită prin

$$\begin{aligned} \text{origine:} \quad & R_0 \equiv (X_0, Y_0, Z_0), \\ \text{direcţie:} \quad & R_d \equiv (X_d, Y_d, Z_d), \end{aligned}$$

unde $X_d^2 + Y_d^2 + Z_d^2 = 1$ (vector normalizat), astfel încât ecuaţia parametrică (explicită) a razei este

$$R(t) = R_0 + t \cdot R_d, \quad t > 0.$$

Planul se defineşte implicit prin

$$Ax + By + Cz + D = 0, \tag{5.16}$$

unde $A^2 + B^2 + C^2 = 1$. Vectorul unitar normal la plan este

$$P_n = (A, B, C), \tag{5.17}$$

iar distanţa de la originea sistemului de coordonate la plan este D. Semnul lui D indică de care parte a planului se află originea sistemului de coordonate.

În aceste condiţii, parametrul t corespunzător intersecţiei se determină din

$$A(X_0 + t \cdot X_d) + B(Y_0 + t \cdot Y_d) + C(Z_0 + t \cdot Z_d) + D = 0,$$

adică

$$t = -\frac{AX_0 + BY_0 + CZ_0 + D}{AX_d + BY_d + CZ_d}, \tag{5.18}$$

sau în notaţie vectorială

$$t = -\frac{P_n \cdot R_0 + D}{P_n \cdot R_d}. \tag{5.19}$$

Numitorul expresiei, $v_d = P_n \cdot R_d$, este o valoare decisivă pentru intersecţie. Dacă $v_d = 0$, atunci raza este paralelă cu planul (deci nu intersectează planul). Dacă $v_d > 0$, normala la plan se "îndepărtează" de rază şi, dacă planele sunt considerate cu o singură faţă (exterior), atunci algoritmul se termină. Altfel, se calculează numărătorul expresiei lui t, $v_0 = -(P_n \cdot R_0 + D)$ şi apoi $t = v_0/v_d$. Dacă $t < 0$, atunci linia definită de rază intersectează planul în spatele originii razei şi, deci, nu există intersecţia căutată. Altfel, se calculează punctul de intersecţie cu:

$$r_i = (x_i, y_i, z_i) = (X_0 + t \cdot X_d, Y_0 + t \cdot Y_d, Z_0 + t \cdot Z_d). \tag{5.20}$$

În mod obişnuit, normala la plan de care este nevoie în algoritm este cea aflată de aceeaşi parte a planului ca şi raza. De aceea este necesară ajustarea semnului normalei P_n în funcţie de relaţia sa cu vectorul de direcţie R_d al razei. Astfel, P_n este înlocuit în ecuaţii cu r_n, unde

$$r_n = \begin{cases} P_n, & P_n \cdot R_d < 0 \ (v_d > 0), \\ -P_n, & \text{altfel.} \end{cases} \tag{5.21}$$

Algoritmul de intersecţie constă în următorii paşi:

Pas 1: se calculează v_d şi se compară cu 0 (2 adunări, 3 înmulţiri, 1 comparare);

Pas 2: se calculează v_0 şi t şi se compară t cu 0 (3 adunări, 3 înmulţiri, 1 comparare);

Pas 3: se calculează coordonatele punctului de intersecţie (3 adunări, 3 înmulţiri);

Pas 4: se compară v_d cu 0 şi se inversează normala (1 comparare).

În total, în cel mai defavorabil caz, se efectuează 8 adunări/scăderi, 9 înmulţiri, 3 comparări.

5.6.3 Intersecţia cu un poligon

Odată stabilită intersecţia cu un plan, e necesar uneori (atunci când în planul respectiv se află o faţă a unui obiect), să se determine dacă punctul de intersecţie este interior sau exterior unui poligon. Metoda clasică constă în aplicarea testului par-impar (capitolul 4).

5.6.4 Intersecţia cu un paralelipiped

Acest tip de intersecţie este util atât în cazul obiectelor paralelipipedice, cât şi în cazul când obiecte complicate (complexe) sunt încadrate într-un paralelipiped (volume de mărginire, capitolul 4) cu care se testează intersecţia şi, în caz afirmativ, se încearcă determinarea intersecţiei cu obiectul în cauză, altfel nu.

Ideea de bază constă în a considera un paralelipiped ca volumul obţinut prin intersecţia a trei perechi de plane paralele.

Raza intersectează fiecare pereche de plane paralele într-un punct mai apropiat şi într-un punct mai îndepărtat. Dacă cea mai mare dintre valorile apropiate este mai mare decât cea mai mică dintre valorile îndepărtate, atunci raza nu intersectează paralelipipedul. Astfel, îl intersectează.

Fie un paralelipiped cu laturi paralele cu axele de coordonate, definit prin:

$$\text{punctul cel mai apropiat de origine: } B_1 = (X_1, Y_1, Z_1),$$
$$\text{punctul cel mai îndepărtat de origine: } B_h = (X_h, Y_h, Z_h).$$

Raza este descrisă parametric.

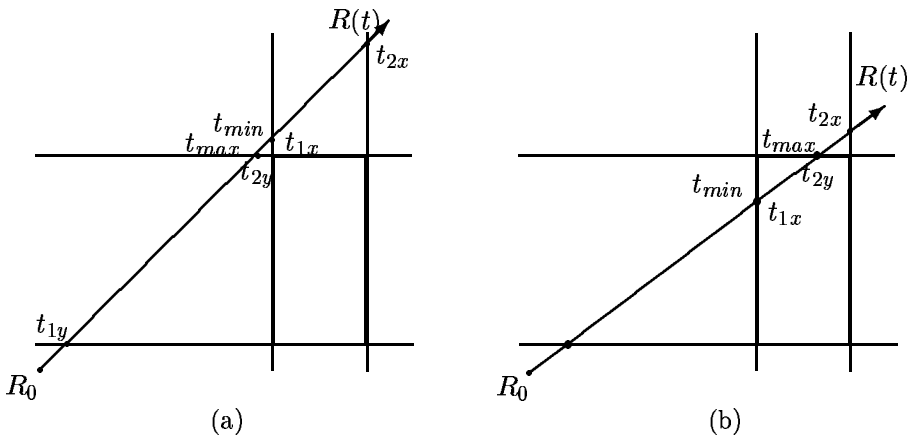


Figura 5.15: Test de intersecţie paralelipiped – rază (a) $t_{min} > t_{max}$ şi raza nu intersectează paralelipipedul (b) $0 < t_{min} < t_{max}$ şi t_{min} caracterizează punctul de intersecţie

Algoritmul de test al intersecţiei este următorul:

(a) fie $t_{min} = -\infty$, $t_{max} = \infty$ (arbitrar de mari);

(b) pentru fiecare pereche de plane paralele delimitatoare ale paralelipipedului (paralele cu axele x, y, z) se efectuează următoarele (exemplificare pentru cazul planelor paralele cu axa x):

- dacă $X_d = 0$, raza este paralelă cu planele şi dacă X_0 nu se află între planele parelele ($X_0 < X_1$ sau $X_0 > X_h$), atunci nu există intersecţie;
- altfel, raza nu este paralelă cu planele şi
  - calculează distanţele până la intersecţiile cu planele:

$$t_1 = (X_1 - X_0)/X_d, \; t_2 = (X_h - X_0)/X_d; \tag{5.22}$$

  - dacă $t_1 > t_2$, interschimbă t_1 cu t_2;
  - dacă $t_1 > t_{min}$, atunci $t_1 = t_{min}$;
  - dacă $t_2 < t_{max}$, atunci $t_2 = t_{max}$;
  - dacă $t_{min} > t_{max}$, atunci paralelipipedul nu este intersectat (figura 5.15.a);
  - dacă $t_{max} < 0$, atunci paralelipipedul este în spatele razei;
- continuă cu alte perechi de plane

(c) dacă testele de respingere de mai sus nu au succes, raza intersectează paralelipipedul în două puncte corespunzătoare lui t_{min} şi t_{max} (figura 5.15.b).

C a p i t o l u l 6

I n t e r f e ţ e u t i l i z a t o r

Dacă în perioada de început a prelucrării datelor cu ajutorul calculatorului electronic, ponderea principală o aveau calculele propriu-zise, odată cu creşterea complexităţii aplicaţiilor şi datorită necesităţii interacţiunii dintre utilizator şi program, a crescut ponderea codului destinat acestei interacţiuni.

Deşi nu există o definiţie unanim acceptată şi general valabilă pentru interfaţa utilizator, se poate spune că aceasta reprezintă ansamblul dispozitivelor logice (de interacţiune între utilizator şi aplicaţie) şi a sarcinilor (task-uri) elementare de interacţiune (implementate cu diverse tehnici de interacţiune) care asigură introducerea interactivă a datelor de intrare, controlul interactiv al execuţiei programului şi obţinerea rezultatelor intermediare sau finale.

6.1 Elemente de bază ale interfetelor utilizator

În contextul general al graficii interactive — şi în particular, al interfeţelor grafice — avantajele şi dezavantajele dispozitivelor de interacţiune sunt abordate la unul dintre următoarele trei nivele:

- dispozitiv (caracteristici hard);
- task (tehnici de interacţiune comparate în implementări utilizând dispozitive diferite);
- dialog (secvenţe de interacţiuni considerate sub aspectul uşurinţei desfăşurării unui dialog).

6.2 Dispozitive de intrare

Dispozitivele de intrare pot fi analizate din două puncte de vedere: fizic şi logic. Dispozitivele fizice de intrare se referă la acele dispozitive hardware prin care utilizatorul introduce informaţii într-un sistem de calcul. Dispozitivele logice de intrare reprezintă tipurile de interacţiune posibile într-o aplicaţie.

Dispozitive logice de bază sunt:

1. locatorul,
2. tastatura,
3. valuatorul,
4. selectorul.

Dezvoltarea unor aplicaţii de avangardă a exercitat presiuni asupra proiectanţilor de dispozitive care au oferit noi tipuri de dispozitive. Pe de altă parte (feedback!), noile dispozitive fizice disponibile reprezintă o provocare pentru proiectanţii de aplicaţii pentru a dezvolta noi tehnici de interacţiune.

Dispozitive specializate de intrare mai puţin comune sunt:

(a) dispozitive pentru recunoaşterea vocii;
(b) diverse tipuri de tablete grafice, sensibile la presiune sau orientare;
(c) dispozitive de interacţiune tridimensională.

Astfel, unele dintre dispozitivele de interacţiune 2D au fost extinse pentru a permite interacţiuni 3D (de exemplu: joystick-ul, trackball-ul, spacebal-l-ul , data glove, etc. Astfel de dispozitive se utilizeaza frecvent în aplicaţii din domeniul realităţii virtuale (Virtual Reality).

6.2.1 Locatorul

Locatorul este un dispozitiv utilizat pentru a indica o poziţie şi/sau o orientare în spaţiul de lucru.

Dispozitivele fizice de tip locator pot fi clasificate funcţie de trei caracteristici independente:

1. după sistemul de coordonate propriu:
   (a) dispozitivele absolute (de exemplu, digitizoarele) au o origine şi precizează poziţiile în raport cu acea origine;
   (b) dispozitivele relative (de exemplu mouse, trackball, joystick) nu au o origine absolută, ele precizând doar modificări faţă de poziţia anterioară;
2. după modul de interacţiune:
   (a) dispozitivele directe (de exemplu, lightpen sau touch screen): poziţia curentă este indicată prin poziţionarea dispozitivului pe ecran;
   (b) dispozitivele indirecte (de exemplu, mouse sau joystick): mutarea cursorului (poziţiei curente) pe ecran cu ajutorul dispozitivului care nu atinge ecranul. Astfel de dispozitive presupun realizarea unei coordonări între ochiul şi mâna utilizatorului.
3. după tipul poziţionării:
   (a) dispozitivele continue (de exemplu, joystick sau mouse) sunt dispozitive care transformă o deplasare lină şi continuă a mâini într-o deplasare similară a cursorului pe ecran.
   (b) dispozitivele discrete (de exemplu, săgeţile de pe tastaură) permit deplasarea cursorului doar pe direcţii prestabilite, cu incrementi de deplasare prestabilită, eventual în poziţii prestabilite.

Observaţie. Viteza de deplasare a cursorului la dispozitivele continue este afectată de rata C/D (Control - to - Display), adică de viteza de deplasare

a mâini raportată la deplasarea cursorului.

6.2.2 Tastatura

Reprezintă unul dintre dispozitivele de intrare cele mai comune. În principal, este utilizat pentru introducerea şirurilor de caractere. Poate fi utilizat însă şi pentru a suplini locatorul (poziţionare), valuatorul (precizarea unei valori) sau selectorul (alegerea unei variante din mai multe posibile).

Cel mai des întâlnit tip de tastatură este tastatura Qwerty. Există însă şi alte tipuri de tastaturi, cum ar fi tastatura Dvorak, la care literele frecvent utilizate sunt plasate în poziţii de bază ale degetelor, sau tastatura pentru stenodactilografie.

6.2.3 Valuatorul

Valuatorul este un dispozitiv logic care permite introducerea unui număr ce reprezintă o valoare numerică a unui parametru.

Din punct de vedere fizic, valuatoarele pot fi de tipul potenţiometrelor de radio rotative sau liniare. Poziţia potenţiometrului faţă de origine precizează valoarea numerică.

Valuatorul poate fi cu plajă fixă de valori, valorile obţinute de la un asemenea dispozitiv fiind absolute (de exemplu, butoane gen potenţiometru de volum), sau cu plajă nemărginită de valori, valorile obţinute fiind relative (potenţiometrii fără poziţie maximă şi minimă).

6.2.4 Selectorul

Selectorul sau întrerupătorul este un dispozitiv logic care permite selecţia unei variante din mai multe alternative posibile (listă de stări sau acţiuni posibile).

Dispozitivele fizice tipice pentru selectare sunt tastele funcţionale dar şi dispozitive specializate de tip întrerupător (de exemplu, butoanele mouse-ului sau a trackball-ului).

6.3 Sarcini de interacţiune

O sarcină de interacţiune (tip de interacţiune, unitate de interacţiune, interaction task) reprezintă unitatea de informaţie semnificativă în contextul aplicaţiei (interactive) pe care utilizatorul o poate produce cu ajutorul unui dispozitiv de interacţiune.

Observaţie. O sarcină de interacţiune diferă de un dispozitiv logic de intrare prin aceea că defineşte ce realizează utilizatorul, în timp ce dispozitivele logice precizează cum se realizează o anumită interacţiune. Sarcinile de interactiune sunt centrate utilizator, în timp ce dispozitivele logice de intrare reprezintă un concept orientat către programator şi pachetele grafice.

Se pune întrebarea: cât de mică sau cât de mare este o astfel de unitate de informaţie? De exemplu, deplasarea unui dispozitiv de poziţionare pe o anumită distanţă, determină o unitate de informaţie dacă noua poziţie semnifică repoziţionarea unui obiect sau specificarea unui capăt de segment. În schimb mutarea cursorului peste un element de meniu nu reprezintă o unitate de informaţie.

Sarcinile de interacţiune pot fi clasificate astfel:

1. sarcini de interacţiune de bază (Basic Interaction Task, prescurtate BIT), care sunt unităţi de informaţie indivizibile (pe baza acestei definiţii şi analogiei cu chimia, le putem numi atomi de interacţiune);
2. sarcini de interacţiune compuse (Composite Interaction Task, prescurtate CIT), care se obţin prin agregarea (compunerea) mai multor BIT (molecule de interacţiune).

O tehnică de interacţiune reprezintă modul în care se realizează o sarcină de interacţiune. De exemplu, o selecţie poate fi realizată printr-un "clic" cu mouse-ul pe un element de meniu, prin introducerea numelui (identificatorului) elementului de selectat de la tastatură, prin apăsarea unei taste funcţionale asociată cu varianta, sau utilizând un dispozitiv de recunoaştere a vocii. Un alt exemplu este cel al unui dispozitiv de intrare folosit în mai multe tipuri de interacţiune.

Un set complet de BIT-uri pentru grafica interactivă este constituit din:

(a) poziţionare,
(b) selecţie,
(c) introducere de text,
(d) introducere de valori numerice.

Fiecare dintre aceste sarcini de interacţiune de bază poate fi realizat prin mai multe tehnici de interacţiune.

6.4 Sarcini de interacţiune de bază

6.4.1 Poziţionarea

Poziţionarea necesită specificarea unui set de coordonate bidimensionale sau tridimensionale. Tehnica folosită de obicei pentru poziţionare implică fie mutarea cursorului pe ecran în poziţia dorită şi apoi apăsarea unui buton, fie precizarea coordonatelor prin intermediul unei tastaturi reale sau virtuale.

Indiferent de tehnica utilizată, apar o serie de probleme generale pentru toate aceste tehnici:

1. sistemele de coordonate,
2. rezoluţia,
3. grila,
4. feedback-ul,
5. timpul de acomodare.

Sisteme de coordonate

Sistemul de coordonate devine o problemă importantă atunci când, în cursul poziţionării, se primeşte reacţia de la sistemul asupra căruia se acţionează.

De exemplu, dacă un dispozitiv de tip locator este mutat spre dreapta pentru a "trage" un obiect, apare problema direcţiei de mutare a obiectului. Există cel puţin trei posibilităţi: obiectul se mută

(a) în sensul pozitiv al axei x din sistemul de coordonate al ecranului (DCS);
(b) în sensul pozitiv al axei x din sistemul de coordonate ale lumii înconjurător (WCS);
(c) în sensul pozitiv al axei x din sistemul de coordonate propriu al obiectului.

Prima soluţie este cea mai viabilă întrucât respectă principiul compatibilităţii stimul – reacţie, care afirmă că răspunsul sau reacţia sistemului la acţiunea utilizatorului trebuie să fie în aceeaşi direcţie şi/sau cu aceeaşi orientare, iar magnitudinea răspunsului trebuie să fie proporţională cu acţiunea.

Rezoluţia

Rezoluţia necesară pentru poziţionare poate varia de la aplicaţie la aplicaţie. Dacă poziţia se introduce numeric (de la tastatură) rezoluţia poate fi oricât de mare. Dacă poziţionarea se face prin tehnici de mutare a cursorului (utilizând un locator) apar limitări de ordin fizic. În mod obişnuit, rezoluţia dispozitivelor de tip mouse, tabletă grafică etc, este de la a 500-a până la a 2000-a parte din unitatea de rezoluţie a dispozitivului de afişare.

Prin utilizarea transformărilor de tip window - to - viewport, de exemplu, pentru a mări o regiune din WCS, este posibil să se realizeze o corespondenţă între unitatea de rezoluţie a ecranului şi o unitate de rezoluţie a sistemului de coordonate ale lumii înconjurătoare, arbitrar de mică, corespondenţă care, de fapt, corespund unei măriri de rezoluţie.

Grila

Un ajutor important în realizarea sarcinii de poziţionare îl poate oferi o grilă (grid, reţea rectangulară) suprapusă peste aria de lucru pentru a uşura alinierea poziţiilor sau obiectelor.

De asemenea, grila poate fi utilă pentru a forţa ca punctele de capăt ale primitivelor grafice să cadă pe grilă (prin rotunjirea coordonatelor locatorului la coordonatele celui mai apropiat nod al grilei).

Feedback

Există două tipuri de poziţionare: spaţială şi lingvistică. În poziţionarea spaţială, utilizatorul cunoaşte poziţia dorită prin raport cu elementele spaţiale din apropiere. În poziţionarea lingvistică, utilizatorul cunoaşte valoarea numerică a coordonatelor poziţiei dorite. Feedback-ul reprezintă marcarea poziţiei curente prin cursor sau prin coordonatele numerice.

Timpul de acomodare

Din punct de vedere al factorului uman se pune problema reducerii timpului în care se ajunge la o bună coordonare ochi – mână în procesul de poziţionare. Un caz specific de poziţionare, care presupune un timp de acomodare mai lung, îl reprezintă poziţionarea continuă (o succesiune de poziţii care definesc o curbă).

6.4.2 Selecţia

Selecţia, ca tip de interacţiune, are scopul de a alege un element dintr-un set de alternative. În mod tipic, seturile de alternative constau din

1. comenzi
2. seturi de valori ale unor atribute
3. clase de obiecte
4. instanţe de obiecte

De exemplu meniul line style (dintr-un program de desenat) constă dintr-un set de valori pentru atributul line style, iar meniul object type (linie, cerc, dreptunghi, text, etc.) este un set de clase de obiecte. Pe de altă parte, mulţimea tuturor obiectelor din suprafaţa de desen constituie un set de instanţe de obiecte.

Anumite tehnici de interacţiune pot fi folosite pentru a selecta din oricare din aceste seturi de alternative. Alte tehnici, însă, sunt utile numai pentru selecţii din anumite clase. De exemplu, prin poziţionarea pe reprezentarea vizuală a unui element dintr-un set de alternative se poate face o selecţie indiferent de tipul setului, pe când selecţia dintr-un set de instanţe de obiecte nu se poate face prin utilizarea tastelor funcţionale.

Ceea ce diferenţiază tehnicile de interacţiune care se pot utiliza pentru selecţie este caracterul fix sau variabil al numărului de elemente ale setului de alternative.

Din acest punct de vedere, un set de alternative poate fi clasificat ca fiind

- cu dimensiunea (relativ) fixă
- cu dimensiunea variabilă

În prima clasă intră seturile de comenzi, atribute, clase de obiecte, care în mod obişnuit au un număr fix de elemente, fără a se exclude însă posibilitatea lărgirii sau reducerii acestuia (nu foarte des şi nu foarte mult!). În a doua clasă, setul de alternative poate fi relativ mare şi se poate schimba frecvent (la fiecare nouă actualizare).

Selecţia din seturi de alternative cu dimensiune variabilă

Două tehnici sunt considerate ca fiind potrivite pentru acest tip de interacţiune: numirea (naming) şi indicarea (pointing).

Selecţia obiectelor prin numire. În cazul acestei tehnici, utilizatorul tastează numele obiectului pe care vrea să-l selecteze. Evident, există situaţii când această tehnică nu este eficientă (de exemplu, când utilizatorul nu cunoaşte numele obiectelor din diverse motive: sunt prea multe, nu apar explicit pe

suprafaţa grafică, etc.). Dacă însă utilizatorul cunoaşte numele obiectelor din setul de alternative, selecţia prin numire poate fi mai eficientă decât selecţia prin indicare (de exemplu, această tehnică permite selecţii multiple prin utilizarea de wildcards).

Dacă se apelează la selecţia prin numire este indicat ca după fiecare tastă apăsată să se afişeze, ca feedback, lista obiectelor al căror nume se potriveşte cu numele parţial introdus. În momentul în care această listă conţine un singur element, numele parţial introdus poate fi completat de către program în mod automat (auto completion).

Selecţia prin indicare. Această tehnică presupune poziţionarea cursorului pe obiectul dorit şi apoi selecţia propriu zisă, fie cu ajutorul unei taste, fie cu un clic pe un buton de mouse. Şi în cazul acestei tehnici pot apare probleme atunci când obiectele sunt dispuse pe mai multe nivele (se acoperă unele pe altele). Soluţia constă de regulă în precizarea nivelului (layer) pe care se face selecţia. Exemple: AutoCAD, Meniuri ierarhice, etc (figura 6.1).

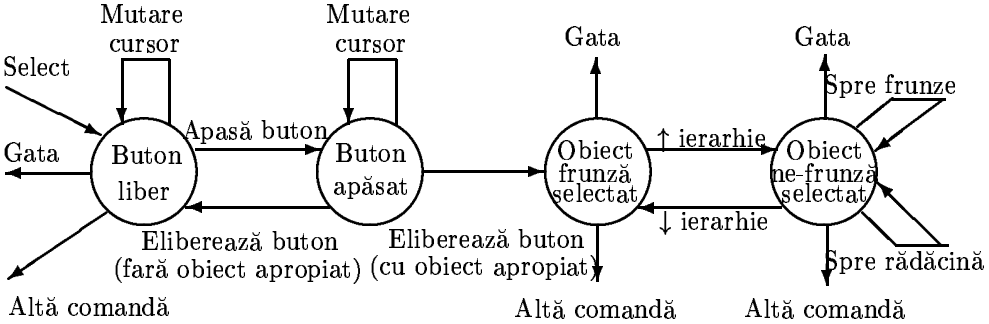


Figura 6.1: Diagrama de stare a tehnicii de selecţie prin indicare a unui obiect dintr-o ierarhie de obiecte

Selecţia din seturi de alternative cu dimensiune (relativ) fixă.

În cazul acestui tip de interacţiune, tehnica cea mai frecvent utilizată o constituie selecţia din meniuri. Problematica proiectării meniurilor include următoarele:

- ordinea elementelor dintr-un meniu;
- meniuri cu un singur nivel sau meniuri multinivel;
- plasarea meniurilor;
- reprezentare vizuală;
- dimensiunea şi forma meniurilor;
- taste funcţionale.

Ordinea elementelor dintr-un meniu. Elementele de meniu pot fi organizate după diverse criterii de ordonare:

1. alfabetică,
2. grupare logică după scopul funcţional,
3. după frecvenţa utilizării (cele mai frecvent utilizate la început),
4. în ordinea importanţei (cele mai importante la început).

Modul în care se ordonează elementele meniurilor trebuie să fie consistent în toate meniurile aplicaţiei.

Meniuri cu un singur nivel sau meniuri multinivel. În cazul în care numărul elementelor de meniu este prea mare pentru a fi afişate simultan, meniul respectiv poate fi divizat într-o ierarhie structurată logic sau într-o secvenţă liniară de elemente care se afişează succesiv (utilizând scroll bars).

În cazul meniurilor ierarhice utilizatorul face o selecţie din setul de alternative de la nivelul cel mai de sus, ceea ce îi pune la dispoziţie un alt set de alternative de pe nivelul imediat inferior. Ierarhiile de meniuri pot fi prezentate în diverse moduri:

1. cu reacoperire
2. în cascadă
3. panouri de ierarhie

Plasarea meniurilor. Meniurile pot fi statice şi vizibile permanent, sau pot apare dinamic, la cerere (pop-up, pull-down, etc.).

Pentru anumite aplicaţii, mai ales cele care presupun existenţa unui display specializat pentru "obiectul" propriu-zis, se poate plasa meniul pe un al doilea monitor.

Meniurile statice sunt indicate în cazul utilizării unui display (fereastră) doar pentru afişarea meniurilor şi precizarea comenzilor.

Meniurile pop-up apar pe ecran pentru a oferi posibilitatea unei selecţii fie:

(a) ca urmare a unei acţiuni explicite a utilizatorului (apăsarea unei taste funcţionale, selecţia unei icoane, etc);
(b) automat, când desfăşurarea dialogului (utilizator–aplicaţie) presupune o selecţie.

Observaţii:

1. Meniul pop-up apare de obicei în poziţia curentă a cursorului (spre care e focalizată atenţia utilizatorului).
2. La apariţia meniului poate fi prezentată ca selecţie curentă (care trebuie confirmată):
   (a) alternativa implicită (cea mai frecvent folosită, cea mai "generală", etc);
   (b) ultima alternativă utilizată.
3. Meniurile pop-up pot fi dependente de context (sensibile la context).

Meniurile pull-down (pulled-out) sunt ancorate într-o bază de meniu (care conţine "titlul" meniurilor pull-down). Activarea acestor meniuri poate fi:

- explicită (selecţie făcută cu cursorul pe titlu);
- implicită (meniul "apare" când cursorul ajunge deasupra titlului).

Reprezentarea vizuală. Problema principală este dacă meniul va conţine nume ale elementelor din setul de alternative sau reprezentări grafice (icoane) ale acestora. Reprezentarea textuală (prin nume) a alternativelor are avantajul

clarității dar conferă un aspect "încărcat" meniului, în timp ce reprezentarea grafică (prin icoane) poate mări randamentul utilizării meniului dacă icoanele sunt sugetive. Uneori se dublează alternativele textuale cu icoane care vor fi din ce în ce mai des folosite, pe măsură ce utilizatorul se acomodează cu ele (exemplu: meniul din MS Word).

Selecția curentă. În cele mai multe situații este de preferat ca elementul curent selectat din meniu să fie marcat pentru a se deosebi într-un fel sau altul de celelalte elemente. Selecția curentă poate fi marcată în diferite moduri:

1. afișare în video invers sau cu altă culoare,
2. marcare cu diverse semne grafice,
3. afișare sub titlul meniului.

Dimensiunea și forma meniurilor. La proiectarea meniurilor trebuie realizat un compromis între dimensiunea mai mare a elementelor de meniu — care permite o selecție mai ușoară — și dimensiunea mai mică — care are avantajul că ocupă mai puțin spațiu din suprafața utilă a ecranului.

Taste funcționale. În cazul în care numărul elementelor de meniu este relativ mic, se pot asocia unora sau tuturor elementelor câte o tastă funcțională ceea ce permite o selecție mult mai rapidă (în cazul unui utilizator versat).

6.4.3 Interacțiunea de tip text

Sarcina de interacțiune text permite introducerea de șiruri de caractere cărora aplicația nu le asociază o semnificație aparte. De exemplu introducerea numelui unei comenzi nu intră în această categorie de interacțiune, spre deosebire de etichetarea obiectelor unui desen sau introducerea de text în cadrul unui editor de texte, care sunt interacțiuni de tip text. În mod obișnuit tehnica de interacțiune pentru introducerea de text presupune utilizarea tastaturii.

O alternativă la folosirea tastaturii pentru introducerea textului o reprezintă utilizarea unui scanner. În principiu, această tehnică presupune utilizarea ulterioară (după scanarea textului) a unei componente de recunoaștere optică a caracterelor (Optical Character Recognition - OCR) care are sarcina de a transforma bitmap-ul obținut după scanare în succesiunea de coduri ASCII corespunzătoare textului din imagine. Tehnica OCR apelează intens la algoritmi de recunoaștere a formelor pentru identificarea caracterelor din imaginea cu text. Problemele care apar la utilizarea acestei tehnici de introducerea textului includ:

1. îmbunătățirea imaginii scanate;
2. eliminarea fondului;
3. decuparea unităților de recunoscut (litere);
4. scalare;
5. orientare;
6. poziționare.

O altă alternativă, relativ recent apărută, o reprezintă utilizarea dispozitivelor de recunoașterea vocii, prin intermediul cărora textul vorbit este transformat în succesiune de coduri ASCII.

6.4.4 Cuantificarea

Acest tip de interacţiune presupune specificarea unei valori numerice, de regulă între o valoare minimă şi una maximă.

Tehnicile de interacţiune utilizate includ:

1. tastarea valorii;
2. poziţionarea unui "potenţiometru";
3. modificarea unui contor pentru incrementare/decrementare.

6.4.5 Sarcini de interacţiune compuse

Tipurile de interacţiune compozite (CIT) sunt combinaţii de tipuri de interacţiune de bază. Există trei tipuri de bază de CIT:

1. celule de dialog (dialog boxes),
2. tehnici de construcţie,
3. manipulare dinamică.

Celule de dialog

De multă ori este necesară o selecţie multiplă dintr-un set de alternative. De exemplu, atribute de text, ca italic, bold, underline, etc. nu se exclud reciproc şi utilizatorul ar dori să selecteze mai multe simultan. În plus pot să existe mai multe seturi de alternative care trebuie precizate simultan. De exemplu, tipul obiectului de desenat, grosimea liniei, aspectul liniei fac parte din seturi de alternative diferite care se pot preciza într-un pas de selecţie, anterior acţiunii de desenare a obiectului respectiv.

Tipurile de meniuri din care se poate face o singură selecţie la un moment dat (pull-down, pop-up) nu sunt satisfăcătoare pentru selecţii multiple fiindcă dispar după o selecţie făcută, fiind necesară activarea lor pentru o a doua selecţie. Această problemă poate fi depăşită prin utilizarea celulelor de dialog care reprezintă o formă de meniu ce rămâne vizibilă până când este părăsită în mod explicit (cu acceptarea selecţiilor făcute sau cu anularea lor). Selecţiile făcute într-o celulă de dialog pot fi corectate pe loc. De asemenea, atributele sau valorile specificate într-o celulă de dialog pot fi aplicate imediat, pentru ca utilizatorul să vadă efectul selecţiei făcute înainte de a părăsi celula de dialog.

Tehnici de construcţie

Un mod de a desena o linie impune utilizatorului să precizeze punctul de start şi punctul de sosire, după care linia este desenată. Cu această tehnică însă, utilizatorul nu are posibilitatea de a vedea rezultatul înainte ca acţiunea să rămână definitivă.

O tehnică superioară constă în utilizarea benzilor elastice (rubberband). După precizarea poziţiei de start, orice nouă poziţie a cursorului este interpretată ca punct de sosire temporar şi linia este desenată corespunzător. De

abia în momentul selectării punctului de sosire definitiv linia devine permanentă. Starea de bandă elastică este activă atâta timp cât un buton al dispozitivului de poziţionare şi de selecţie este apăsat. În această poziţie deplasarea cursorului provoacă deplasarea liniei. Pornind de la această tehnică de desenare a liniilor se derivează tehnica desenării dreptunghiului elastic, cercului elastic, elipsei elastice.

O altă tehnică de construcţie priveşte desenarea de linii frânte care presupune o succesiune de selecţii (ale capetelor segmentelor care alcătuiesc linia poligonală) intercalate cu poziţionări (care precizează poziţia capetelor).

În oricare dintre aceste tehnici pot fi aplicate asupra poziţiei cursorului anumite constrângeri. De exemplu, cursorul poate fi obligat să se deplaseze pe anumite direcţii sau la distanţe limitate de poziţia de start.

Un alt tip de constrângere îl reprezintă utilizarea unui, aşa numit, câmp de gravitate, pentru a uşura suprapunerea a doua poziţii: în momentul în care cursorul intră în câmpul de gravitate al unui punct de capăt al unui obiect existent pe suprafaţa de desen, poziţia care va fi selectată va coincide cu (va fi atrasă spre) acest punct.

Manipulare dinamică

În multe situaţii nu este suficientă desenarea de obiecte geometrice, ci este necesară şi modificarea acestora. Modificările posibile includ schimbarea poziţiei, rotirea şi scalarea.

Schimbarea poziţiei se realizează prin tehnica de click-and-drag, care implementează succesiunea buton apăsat-deplasare-buton eliberat, corespunzând selecţiei obiectului, deplasării în noua poziţie şi selecţiei noii poziţii.

Tehnici similare se utilizează pentru rotirea sau scalarea unui obiect şi care presupun selecţia obiectului, precizarea noii orientări (prin definirea unghiului de rotaţie) sau a noii dimensiuni (prin definirea factorul de scală) şi, apoi, selecţia noii situaţii.

Deplasarea, rotirea sau scalarea afectează un obiect în întregime. Uneori, este de dorit însă mutarea unui punct individual dintr-un obiect. Această operaţie se poate realiza de asemenea prin tehnica click-and-drag.

6.5 Principii în proiectarea interfeţelor utilizator

Pentru a asigura calitatea unei interfeţe din punct de vedere utilizator trebuie avute în vedere o serie de principii de proiectare:

1. consistenţă,
2. asigurarea feedback-ului,
3. minimizarea posibilităţilor de eroare (la utilizare),
4. asigurarea posibilităţii de revenire din eroare,
5. posibilitatea utilizării pe nivele de îndemânare,
6. reducerea necesităţii de memorizare.

6.5.1 Consistenţa

Un sistem este consistent dacă modelul conceptual, funcţionalitatea, secvenţialitatea şi legătura cu dispozitivele hard sunt uniforme şi respectă câteva reguli simple şi deci nu prezintă excepţii şi condiţii speciale.

Principalul scop urmărit prin consistenţa unui sistem este de a permite utilizatorului să generalizeze cunoştinţe despre un aspect al sistemului la alte aspecte.

De asemenea consistenţa permite evitarea frustărilor care apar când un sistem nu se comportă într-un mod logic.

Metoda principală de a asigura consistenţa este proiectarea top-down a sistemului.

Exemple de consistenţă:

La ieşiri,

(a) se vor utiliza întotdeauna aceleaşi codificări (de exemplu roşu → stop, verde → porneşte);
(b) mesajele despre starea sistemului apar întotdeauna într-o poziţie logică fixată;
(c) elementele de meniu îşi vor păstra întotdeauna aceiaşi poziţie relativă în meniu.

La intrări,

(a) tastele (CR, TAB, BS, etc) au întotdeauna aceleaşi funcţiuni;
(b) comenzile globale (Help, Status, Cancel) pot fi inversate oricând;
(c) comenzile generice (Move, Copy, Delete) sunt disponibile şi pot fi aplicate, cu rezultate previzibile, asupra oricărui tip de obiect din sistem.

6.5.2 Asigurarea feedbackului

Are ca scop permanenta informare a utilizatorului asupra stării sistemului, asupra efectelor acţiunilor întreprinse şi asupra acţiunilor posibile ce urmează a fi întreprinse.

Reacţia sistemului se poate asigura pe trei nivele, corespunzătoare nivelelor de proiectare:

- semantic: funcţională,
- sintactic: a secvenţializării,
- lexical: a legăturii cu dispozitivele (hard) fizice.

Nivelul cel mai de jos este cel hardware: fiecare acţiune a utilizatorului cu un dispozitiv de intrare ar trebui să provoace o reacţie imediată şi evidentă (de ex. caracterele tastate vor fi afişate pe ecran, mutarea mouse-ului este fidel urmată de mişcarea cursorului).

La nivel sintactic acceptarea/neacceptarea fiecărei unităţi (cuvânt) a limbajului de intrare (comandă, poziţie, obiect, etc), trebuie semnalată de sistem. De exemplu, un obiect selectat sau un element de meniu selectat va fi "iluminat", astfel încât utilizatorul să ştie că acţiunea sa a fost acceptată.

La nivel funcţional, o formă de feedback o constituie notificarea faptului că acţiunea comandată este în curs de efectuare (dacă durează mai mult de câteva

secunde). Altă formă de feedback la nivel funcţional presupune anunţarea terminării operaţiei fie printr-un mesaj, fie prin afişarea rezultatelor.

Trebuie făcută o distincţie între feedback-ul din domeniul problemei şi respectiv cel din domeniul controlului. Feedback-ul în domeniul problemei se referă la obiectele de manipulat: existenţa, poziţia, apariţia. Feedback-ul în domeniul controlului se referă la mecanisme de control ale sistemului interactiv: stare, valori curente şi implicite, meniuri, etc.

6.5.3 Minimizarea posibilităţilor de eroare

Ideea este aceea de a respecta următoarea regulă: "Nu pregăti capcane pentru utilizator!"

Exemple:

1. Nu oferi posibilitatea alegerii unui element de meniu care va genera mesaje de genul: selecţie ilegală, comandă invalidă, etc.!
2. Nu lăsa utilizatorul să selecteze Move, Copy, Delete când nimic nu este selectat!
3. Nu lăsa utilizatorul să selecteze Paste dacă clipboard-ul este gol!

În general, "oferta" care se face utilizatorului pentru a selecta din ea trebuie să fie senzitivă la context: sistemul ghidează utilizatorul în funcţie de contextul respectiv, făcând imposibilă selectarea comenzilor nepermise.

6.5.4 Asigurarea posibilităţii de revenire din eroare

De regulă patru tipuri de corectare a erorilor sunt avute în vedere la proiectarea interfeţelor utilizator: Undo, Abort, Cancel, Correct.

Dacă s-a lansat o operaţie din greşeală, este recomandat Undo. Acesta se poate asigna pe un nivel sau pe mai multe nivele (caz în care se păstrează comenzile şi starea sau parametri într-o stivă). În acest context (Undo multinivel) este utilă o comandă Redo.

În cursul execuţiei operaţiei în cauză, dacă utilizatorul îşi dă seama că a comis o eroare, trebuie să i se asigure posibilitatea întreruperii operaţiei şi revenirea la starea anterioară selecţiei respectivei operaţii printr-o comandă Abort.

Undo şi Abort pot fi comasate într-o singură comandă.

Dacă utilizatorul decide, în cursul specificării parametrilor pentru o operaţie, că a greşit, el poate renunţa prin comanda Cancel la terminarea specificării şi lansării operaţiei.

De asemenea, dacă nu operaţia, ci doar anumiţi parametri sunt eronaţi, utilizatorul poate dispune de facilitatea de a corecta ceea ce a greşit.

6.5.5 Posibilitatea operării pe mai multe nivele

Diverse aplicaţii trebuie proiectate astfel încât să permită utilizarea lor pe diverse nivele de complexitate: de la nivelul unui utilizator fără experienţă, care de abia învaţă comenzile de bază şi are nevoie de sprijin din partea sistemului,

până la nivelul unui utilizator cu experienţă, care este dispus să renunţe la anumite informaţii ajutătoare în schimbul unei viteze sporite de lucru. Utilizatorul cu experienţă trebuie să poată face uz de tehnici de interacţiune mai rapide.

Metodele utilizate pentru a permite utilizarea interfeţei pe mai multe nivele de îndemânare sunt:

1. acceleratori (taste funcţionale în loc de meniu),
2. prompteri (ghidarea acţiunii următoare),
3. help,
4. extensibilitate (posibilitatea combinării unor comenzi elementare într-una mai complexă),
5. ascunderea complexităţii (set minimal de comenzi).

6.5.6 Minimizarea memorizării

Reducerea necesităţii de memorizare a comenzilor disponibile în sistem se realizează de obicei prin apelul la capacitatea utilizatorului de a recunoaşte (intui) semnificaţia comenzilor reprezentate fie prin numele lor fie printr-o icoană. Pe de altă parte, necesitatea memorizării comenzilor poate fi redusă prin asigurarea unor servicii de tip Help ca cele menţionate mai sus.

6.6 Software pentru interfeţe utilizator

Software pentru interfeţe (grafice) utilizator există pe diverse nivele (figura 6.6).

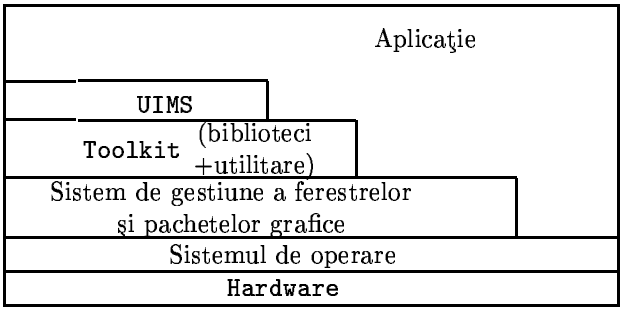


Figura 6.2: Software pentru interfeţe utilizator

Dispozitivele logice de intrare (locator, valuator, pick, choice, string, stroke - GKS, PHIGS) pot opera într-unul din modurile:

1. request;
2. sample;
3. event.

Fiecărui dispozitiv îi este asociată o măsură (tipul informaţiei returnate de dispozitiv).

Modurile de operare indică modalitatea prin care programul de aplicaţie recepţionează o schimbare a măsurii dispozitivelor lor.

În modul request (la cerere) programul de aplicaţie solicită o intrare de la dispozitiv care întoarce odată cu controlul şi măsura asociată acţiunii utilizatorului (de exemplu, deplasare) asupra dispozitivului. Acţiunea se numeşte trigger. Dezavantajele sunt următoarele:

(a) intrarea se face în mod sincron;

(b) nu există posibilitatea de feedback dinamic pentru că aplicaţia nu recapătă controlul decât după terminarea acţiunii.

În modul sample (polling) dispozitivele sunt cercetate (baleiate) unul după altul în speranţa surprinderii unei modificări a măsurii. Dezavantajul constă în faptul că, în timp ce se procesează intrarea de la un dispozitiv, se pot pierde intrările de la alte dispozitive.

În modul event intrările (acţionările asupra dispozitivelor) sunt asincrone în sensul că orice eveniment asociat cu un dispozitiv generează o intrare într-o coadă de evenimente (figura 6.3) de regulă FIFO (se poate avea însă în vedere şi asocierea de priorităţi diferitelor clase de dispozitive).

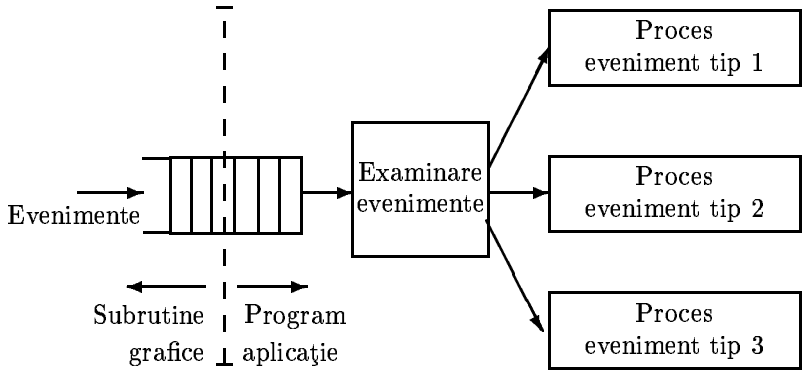


Figura 6.3: Tratarea evenimentele din coadă

6.6.1 Sisteme de gestiune a ferestrelor

Un sistem de gestiune a ferestrelor (window-management system) asigură majoritatea caracteristicilor importante ale interfeţelor utilizator moderne: rularea aplicaţiilor în ferestre distincte pe ecran, abilitatea de a redimensiona şi muta ferestrele, meniuri pop-up şi pull-down, ferestre de dialog (dialog boxes).

Un sistem de gestiune a ferestrelor este, în principal, un gestionar de resurse. El alocă spaţiu pe ecran diverselor programe care încearcă să folosească ecranul şi apoi gestionează utilizarea zonelor de ecran alocate, astfel încât programele să nu interfereze (pe ecran) unele cu altele. De asemenea, sistemul de gestiune a ferestrelor alocă programelor care solicită intrări resursele dispozitivelor de

interacţiune şi, apoi, direcţionează informaţia de intrare de la dispozitiv către coada de evenimente a programului destinaţie.

Un sistem de gestiune a ferestrelor are două părţi:

1. gestionarul de ferestre (window manager) cu care interacţionează utilizatorul atunci când solicită crearea, redimensionarea, mutarea, deschiderea, închiderea, etc. unei ferestre;
2. sistemul de ferestre (window system), care reprezintă componenta funcţională responsabilă cu crearea, redimensionarea, mutarea, deschiderea, închiderea, etc. efectivă a unei ferestre.

Gestionarul de ferestre este construit deasupra sistemului de ferestre şi utilizează serviciile pe care acesta i le pune la dispoziţie, pentru a satisface cererile utilizatorului.

Programele construite deasupra sistemului de ferestre sunt denumite uneori programe client, iar sistemul de ferestre este denumit program server.

În anumite sisteme de gestiune a ferestrelor construite pe o arhitectură client-server, precum X-Windows, gestionarul de ferestre este el însuşi un client al sistemului de ferestre (figura 6.4).

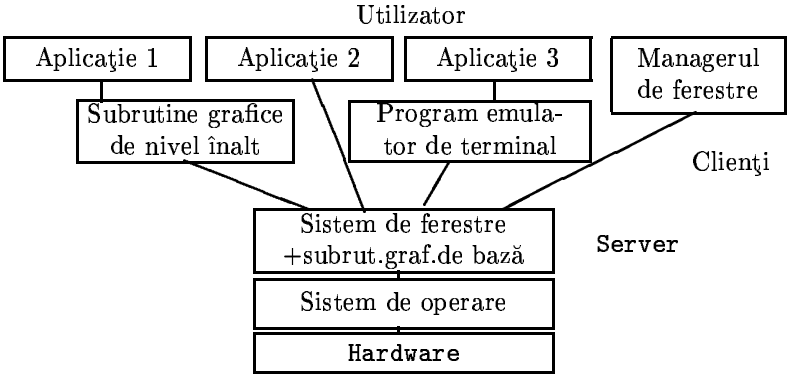


Figura 6.4: Relaţia dintre sistemul de ferestre, sistemul de operare şi programul-aplicaţie

Anumite sisteme de ferestre sunt astfel proiectate încât să suporte mai mulţi manageri de ferestre, fiecare cu aspect şi reacţie (look and feel) distincte (X-Windows).

Observaţie. Managerul de ferestre, şi nu sistemul de ferestre, este cel care determină aspectul unei ferestre şi modul în care utilizatorul interacţionează cu aceasta.

De obicei se integrează un pachet de subrutine grafice în sistemul de ferestre care asigură o parte din funcţionalitatea sistemului de ferestre, dar care pot fi apelate şi direct.

6.6.2 Tratarea ieşirilor în sistemele de ferestre

Resursa de ieşire alocată de sistemul de ferestre unui program client este spaţiul ecran, care trebuie astfel gestionat încât clienţii să nu interfereze unii cu alţii în utilizarea spaţiului ecran.

Strategiile de alocare a spaţiului ecran pot să varieze în mod considerabil de la un sistem de ferestre la altul, dar, în principiu, se încadrează în trei categorii mari. Principala diferenţă constă în modul de afişare a zonei din fereastră devenită vizibilă în urma măririi ferestrei, în urma descoperirii ferestrei sau în timpul defilării (figura 6.5).

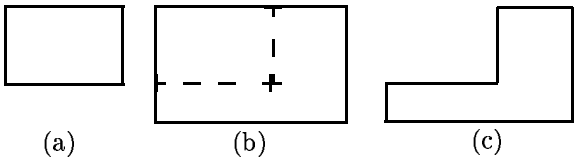


Figura 6.5: Lărgirea ferestrei (a) la dimensiunea ferestrei (b) necesită decuparea primitivelor faţă de regiunea (c)

Strategii:

- Un sistem de ferestre minimal nu-şi asumă nici o responsabilitate în afişarea suprafeţei nou expuse a ferestrei, ci transmite un eveniment de tipul fereastră expusă clientului asociat ferestrei . Un astfel de sistem de ferestre nu salvează partea acoperită a ferestrei. Când un client afişează o informaţie în fereastră, primitivele de ieşire sunt decupate relativ la partea vizibilă a ferestrei. La mărirea suprafeţei vizibile a ferestrei, clientul asociat ferestrei recepţionează un mesaj de tip fereastră expusă şi este responsabil cu afişarea primitivelor corespunzătoare zonei proaspăt expuse a ferestrei.
- Sistemele de ferestre care dispun de mai multă memorie salvează partea acoperită a ferestrei, astfel încât clientul este degrevat de sarcina afişării porţiunii proaspăt expuse. Problema care se pune este cât din fereastra acoperită trebuie salvat. În mod tipic, se salvează dimensiunea maximă posibilă a ferestrei (egală cu dimensiunea ecranului). Anumite sisteme de ferestre salvează chiar porţiuni mai mari decât ecranul, soluţie care, deşi costisitoare, devine din ceea ce mai atractivă datorită tendinţei de scădere a preţului memoriei calculatoarelor. Clientul este implicat în reafişare doar în cazul în care se face o defilare în fereastră, dincolo de porţiunea salvată, sau în cazul unei scalări.
  O stategie puţin diferită constă în păstrarea de către managerul de ferestre a unei copii complete pentru fiecare fereastră. De câte ori o parte a unei ferestre este descoperită, porţiunea corespunzătoare din copia ferestrei este afişată pe ecran. Actualizarea ferestrei conform acestei strategii este lentă întrucât clientul asociat poate să scrie în porţiunea acoperită a ferestrei şi este necesară refacerea copiei. Actualizarea unei ferestre com-

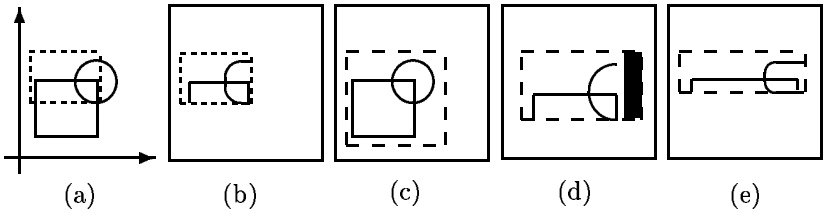


Figura 6.6: Relaţiile dintre fereastra WCS şi fereastra managerului (de ferestre) (a) Imagine în WCS (b) Imaginea printr-o fereastră (c) Lărgirea ferestrei în WCS (d) Lărgirea ferestrei managerului cu scalare uniformă (e) Lărgirea ferestrei managerului cu scalare neuniformă

plet descoperite se face mai rapid întrucât copia ei trebuie actualizată doar în momentul dinaintea acoperirii ferestrei. O alternativă a acestei strategii constă în partiţionarea fiecărei ferestre în regiuni rectangulare, doar acelea care nu sunt vizibile fiind salvate.

- O strategie diferită este utilizată de sistemele de ferestre care menţin o listă a primitivelor (vizibile sau nu) afişate în fiecare fereastră. La fiecare reafişare a ferestrei, lista asociată este parcursă şi se fac decupările necesare. Această strategie presupune utilizarea unui hardware pentru conversie de scanare rapidă.

O altă problemă care apare frecvent este efectul operaţiei de redimensionare a unei ferestre asupra informaţiei vizibile în fereastră. Există două posibilităţi şi programul client ar trebui să fie capabil să le trateze pe amândouă.

În primul caz, în momentul redimensionării ferestrei de către utilizator, fereastra WCS îşi modifică dimensiunea în mod corespunzător, utilizatorul "văzând" mai mult sau mai puţin din "lume", în funcţie de operaţia de redimensionare (mărire sau micşorare) efectuată (figura 6.6.c).

În al doilea caz (figura 6.6.d), dimensiunea ferestrei WCS rămâne fixă astfel încât prin mărirea ferestrei se vede aceeaşi scenă, dar la o scară mai mare. În acest caz se mai pune şi problema scalării neuniforme pe cele două axe (figura 6.6.e) .

Anumite sisteme de ferestre permit crearea de ferestre ierarhice, adică ferestre care conţin subferestre (figura 6.7). Acestea pot fi utilizate de exemplu la implementarea ferestrelor de dialog: întreaga fereastră de dialog este definită ca o fereastră şi fiecare câmp, buton, scroll-bar, este definit ca o subfereastră şi evenimente de tipul buton - mouse apăsat sunt disponibile pentru fiecare subfereastră. Aceasta inseamnă că fiecare eveniment este însoţit de numele subferestrei în care a apărut.

În mod tipic un sistem de ferestre ar trebui să dispună de următoarele funcţiuni:

1. crează o fereastră nouă (care devine fereastră curentă);
2. setează poziţia ferestrei curente;

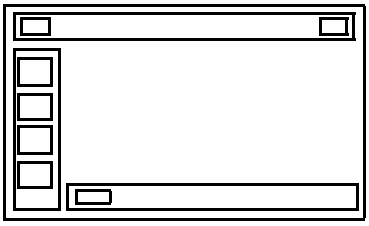


Figura 6.7: Divizarea unei ferestre în ferestre ierarhice

3. setează dimensiunea ferestrei curente;
4. selectează o fereastră (care devine fereastră curentă);
5. plasează o fereastră invizibilă deasupra tuturor celorlalte (aceasta este noua fereastră curentă);
6. ascunde fereastra curentă şi expune toate ferestrele acoperite de aceasta;
7. setează titlul ferestrei curente;
8. citeşte poziţia ferestrei curente;
9. citeşte dimensiunea ferestrei curente;
10. plasează fereastra curentă deasupra tuturor celorlalte;
11. plasează fereastra curentă sub toate celelalte;
12. şterge fereastra.

Pe lângă spaţiul ecran, sistemul de ferestre este responsabil şi de alocarea unei alte resurse: intrări ale tabelei de culori (look-up table). În principiu, două strategii de alocare a intrărilor în tabela de culori sunt utilizate:

- se partajează intrările între toţi clienţii, alocându-se fiecăruia un număr fix. Dezavantajele metodei sunt:
  (a) în cazul în care sunt puţini clienţi, rămân intrări neutilizate în tabele;
  (b) pot exista clienţi cărora li se vor aloca mai puţine intrări decât au nevoie, ceea ce implică degradarea calităţii modului de afişare al ferestrei (din punct de vedere al culorilor).
- se alocă toate intrările necesare clientului a cărui fereastră conţine cursorul. Dezavantajul constă în faptul că aspectul coloristic al ferestrelor se poate modifica dramatic (chiar dacă pentru scurt timp!), în momentul trecerii cu cursorul dintr-o fereastră într-alta.

O altă soluţie posibilă constă în alocarea de culori în loc de intrări în look-up table. Toţi clienţii care utilizează aceleaşi culori vor lucra cu aceeaşi indecşi în tabela de culori. În cazul în care un client solicită o culoare pentru care nu există intrare în tabelă, şi aceasta nu este plină, se creează o nouă intrare. Dacă tabela este plină, atunci se va folosi indexul culorii celei mai apropiate de cea solicitată. Dezavantajul evident este acela că cea mai apropiată culoare ar putea fi inacceptabil de departe de cea solicitată.

6.6.3 Tratarea intrărilor în sistemele de ferestre

Resursa de intrare alocată şi gestionată de către sistemul de ferestre pentru clienţii săi reprezintă setul de dispozitive de intrare şi evenimentele generate de acestea. Sistemul de ferestre trebuie să dispună de informaţiile necesare pentru a dirija evenimentele spre clientul căruia îi sunt destinate.

Procesul de dirijare a evenimentelor către clientul adecvat poartă denumirea de multiplexare.

Pe lângă evenimentele generate de diverse dispozitive, un sistem de ferestre poate lua în considerare şi generarea unor evenimente speciale, ca, de exemplu, intrarea într-o nouă fereastră (window enter), respectiv ieşirea din fereastra curentă (window leave), care permit evidenţierea (highlighting) ferestrei care conţine cursorul.

Un alt tip de eveniment special – window damage – ar putea fi utilizat pentru a preciza sistemului de ferestre că fereastra clientului căruia îi este destinat evenimentul trebuie reafişată.

În cazul în care există ferestre ierarhice, evenimentele pot fi dirijate şi către subferestre.

Două modalităţi sunt utilizate, în principal, de către sistemele de ferestre pentru dirijarea evenimentelor către clienţi:

1. evenimentul este dirijat către clientul a cărui fereastră conţine cursorul (real-estate-based event routing);
2. evenimentul este dirijat către un client precizat de către un alt client, posibil, dar nu obligatoriu, acelaşi (listener event routing). De exemplu, managerul de ferestre poate avea o comandă prin care utilizatorul poate dirija intrările de la tastatură către clientul care posedă o anumită fereastră.

6.7 Utilitare pentru tehnici de interacţiune

Utilitarele pentru tehnici de interacţiune (interaction-technique toolkits) sunt subrutine care implementează legătura cu hardware-ul în cadrul unei interfeţe utilizator. Acestea determină caracteristica interfeţelor utilizator descrisă de termenul look and feel" (aspect şi reacţie).

Tehnicile de interacţiune pot fi implementate direct în aplicaţie, dar această soluţie are două dezavantaje majore:

1. este consumatoare de timp;
2. nu răspunde necesităţii utilizatorului de a întâlni interfeţe similare la aplicaţii distincte.

Utilitarele pentru tehnici de interacţiune, de fapt, colecţii de subrutine de bibliotecă accesibile programelor de aplicaţie, înlătură ambele dezavantaje menţionate.

Utilizarea aceleiaşi colecţii în toate aplicaţiile şi chiar în sistemul de ferestre reprezintă o metodă des folosită în scopul unificării aspectului interfeţelor.

Modalităţile de implementare sunt următoarele: peste sistemul de gestiune a ferestrelor, sau, în lipsa lui, peste pachetul de subrutine grafice.

Exemple: colecţia de subrutine care însoţeşte sistemul de gestiune a ferestrelor Andrew, OSF/Motil, InterViews, implementare OpenLock.

În sistemul de gestiune a ferestrelor X-Windows tehnicile de interacţiune sunt denumite widgets (WIndows gaDGETS) şi o listă tipică de astfel de widgets include:

- fereastră de dialog,
- fereastră de selecţie fişier,
- fereastră de avertizare,
- fereastră de ajutor (help),
- fereastră de mesaje,
- butoane radio,
- banc de butoane radio,
- butoane de selecţie (marcare),
- bancuri de selecţie,
- butoane basculante,
- bancuri de butoane basculante,
- meniu fix,
- meniu pop-up,
- bară de defilare,
- fereastră de introducere text,
- fereastră de aplicaţie.

Fiecare widget este implementat ca o fereastră care poate conţine la rândul ei subferestre.

Utilitarele pentru tehnici de interacţiune dispun, în mod tipic, de notificatori (pentru intrare/ieşire în/din ferestre) pentru a permite apelul de proceduri de reîntoarcere atunci când anumite evenimente se produc în subferestrele lor.

În lista de widgets de mai sus apar atât elemente de bază cât şi altele compuse. De obicei, colecţiile de subrutine dispun de mijloace de compunere a widget-urilor. De exemplu, fereastra de dialog conţine atât butoane radio, cât şi butoane de selecţie binară.

Crearea de ierarhii de ferestre este o activitate migăloasă pentru orice programator (chiar dacă se dispune de o colecţie de widgets-uri). De aceea s-au realizat editoare interactive pentru proiectarea (crearea şi modificarea) aspectului ferestrelor de dialog (care, de regulă, conţin celelalte tipuri de widgets). Ieşirea din astfel de editoare este o reprezentare a ferestrei compuse, fie ca structură de date care poate fi translatată în cod sursă, fie sub formă de cod compilat. În toate situaţiile sunt însă prevăzute mecanisme pentru editarea legăturilor cu aplicaţia.

O soluţie alternativă pentru crearea de meniuri şi ferestre de dialog o constituie utilizarea unui limbaj de nivel înalt pentru descrierea acestora.

În fine, proiectantul de interfeţe poate crea un widget sau poate compune mai multe widgets în mod interaciv, prin exemple (sistemul Peridot).

6.7.1 Sisteme de gestiune a interfeţelor utilizator

Sistemele de gestiune a interfeţelor utilizator (User-Interface Management System - UIMS) asistă utilizatorul în proiectarea şi implementarea aspectului interfeţei dar şi, în anumite cazuri, a semanticii acesteia. Orice UIMS oferă mijloace pentru definirea secvenţelor de acţiuni utilizator admisibile, pentru specificarea aspectului interfeţei şi a conţinutului mesajelor de help si eroare. Un UIMS poate mări considerabil productivitatea programatorului, facilitând în acelaşi timp rafinarea iterativă a interfeţei pe măsură ce se câştigă experienţă.

Aplicaţiile dezvoltate peste un UIMS constau, în mod tipic, dintr-un set de subrutine, denumite uneori rutine de acţiune sau rutine de acţiune semantică. UIMS-ul apelează rutina de acţiune adecvată ca răspuns la o intrare (input) produsă de utilizatorul aplicaţiei. În schimb, rutina de acţiune influenţează dialogul – de exemplu, modificând acţiunile utilizator posibile în continuare. Se poate spune că UIMS şi rutinele de acţiune partajează controlul dialogului. Acest model este cunoscut sub denumirea de model cu control partajat (shared-control model). Prin opoziţie cu acest model, există UIMS-uri în care rutinele de acţiune nu au nici o influenţă asupra dialogului. Aceste UIMS-uri se încadrează în modelul cu control extern (external-control model).

Serviciile specifice pe care diverse UIMS-uri le oferă proiectantului de interfeţe utilizator pot varia dar componenta esenţială, care nu lipseşte din nici un UIMS, o reprezintă specificarea secvenţei dialogului. Această componentă controlează ordinea în care tehnicile de interacţiune sunt puse la dispoziţia utilizatorului aplicaţiei.

Secvenţializarea dialogului

Secvenţele admisibile de acţiuni ale utilizatorului pot fi definite într-o varietate de moduri: via reţele de tranziţie (numite şi diagrame de stări), reţele de tranziţie recursive, limbaje bazate pe evenimente sau prin exemple, (unde proiectantul demonstrează sistemului succesiunile de acţiuni permise şi sistemul "învaţă" care secvenţe sunt posibile). Comun tuturor acestor metode este conceptul unei stări a interfeţei utilizator şi acţiuni ale utilizatorului asociate care pot fi executate din această stare. Fiecare din metodele de specificare codifică starea interfeţei utilizator într-un mod specific,care generalizează folosirea uneia sau a mai multor variabile de stare. Dacă se crează o interfaţă utilizator senzitivă la context, răspunsul sistemului la acţiunile utilizatorului trebuie să depindă de starea curentă a interfeţei.

Cea mai simplă şi cea mai puţin puternică, dar totuşi utilă, metodă de specificare a succesiunii dialogului o reprezintă reţeaua de tranziţie sau diagrama de stări. Reţelele de tranziţie au o singură variabilă de stare, un întreg ce indică starea curentă. Acţiunile utilizatorului cauzează tranziţii de la o stare la alta; fiecare tranziţie are asociată cu ea zero sau mai multe rutine de acţiune care sunt apelate când tranziţia se întâmplă. Pe lângă aceasta, stările pot avea rutine de acţiune asociate care sunt executate oricând se intră în această stare. Reţelele de tranziţie sunt utile, în special, pentru găsirea incosistenţelor în secvenţializarea

dialogului, şi pot fi folosite fără probleme pentru a determina numărul necesar de paşi pentru a completa o secvenţă de sarcini de interacţiune.

Reţelele de tranziţie au totuşi dezavantaje. În primul rând, starea interfeţei utilizator este, în mod tipic, bazată pe un număr de variabile de stare, şi necesitatea de a găsi toate combinaţiile posibile de valori ale acestor variabile specifice unei stări este dificilă şi non-intuitivă pentru proiectantul interfeţei utilizator. Reţeaua de tranziţie din figura 6.8) descrie o aplicaţie cu următoarele comenzi:

- selectează obiect (stabileşte obiectul curent - CSO);
- deselectează obiect (nu mai există CSO!);
- crează obiect (stabileşte un CSO);
- şterge CSO (nu mai există CSO!);
- copiază CSO în clipboard (trebuie să existe un CSO; umple clipboard-ul);
- preia (paste) din clipboard (trebuie sa existe ceva în clipboard; crează un CSO);
- goleşte clipboard-ul (clipboard-ul trebuie să conţină ceva şi va rămâne gol).

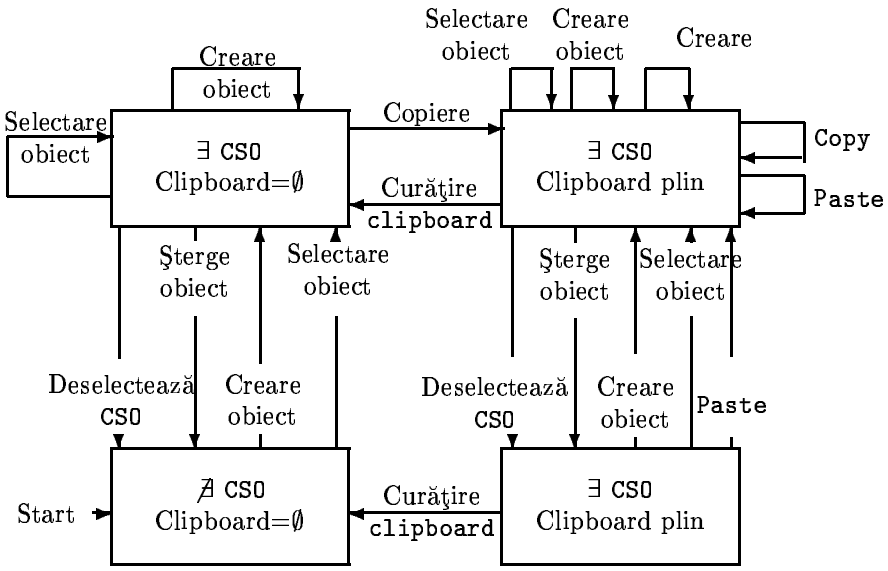


Figura 6.8: Reţea de tranziţie cu patru stări

Diferite construcţii specializate au fost dezvoltate pentru a simplifica reţelele de tranziţie. De exemplu, putem alinia problema de help prin folosirea subreţelelor într-un mod analog subrutinelor, pentru a ascunde un detaliu repetitiv localizat. Reţelele de tranziţie care pot apela recursiv subreţele sunt numite

reţele de tranziţie recursive. Variabilele de stare în acest caz sunt întreaga stivă de stări salvate plus starea reţelei de tranziţie curent active.

Notaţia Backus - Naur (BNF) poate de asemenea fi folosită pentru a defini secvenţializarea, şi este echivalentă ca putere de reprezentare cu reţelele de tranziţie recursive (ambele sunt echivalente cu automatele push-down). Mai multe UIMS mai vechi au fost bazate pe specificaţii BNF.

Când reţelele de tranziţie devin mai complicate, cu expresii logice la tranziţii şi apeluri de subrutine, se ajunge la specificaţii asemănătoare unor programe (program-like). În definitiv, limbajele de programare reprezintă cel mai puternic mod de a specifica secvenţializările şi condiţiile multiple asociate, de multe ori, cu tranziţiile.

Mai multe limbaje bazate pe evenimente au fost dezvoltate special pentru specificarea interfeţelor utilizator. Este de remarcat că limbajele bazate pe evenimente, spre deosebire de limbajele de programare tradiţionale, nu au un flux explicit de control. În schimb, oricând o condiţie if devine adevărată, acţiunile asociate sunt executate. Aşadar, limbajul bazat pe evenimente este un sistem de reguli de producţie. Limbajele bazate pe evenimente sunt mai puternice decât reţelele de tranziţie, reţelele de tranziţie recursive şi gramaticile, dar prea complicate pentru cazurile simple.

Un mod diferit de a defini sintaxa (secvenţializarea) dialogului este prin exemplu. În acest caz, UIMS-ul se plasează într-un mod "de învăţare", şi apoi se trece prin toate secvenţele acceptabile de acţiuni. Proiectantul poate începe cu un meniu principal, selectează un câmp din acest meniu şi merge printr-un director pentru a localiza submeniul, celula de dialog sau obiecte specifice aplicaţiei, pentru a fi prezentate utilizatorului ca răspuns la selectarea meniului principal. Obiectul apare pe ecran şi proiectantul poate să indice poziţia, mărimea sau alte atribute pe care obiectul trebuie să le aibă atunci când aplicaţia este într-adevăr executată. Proiectantul continuă să execute câteva operaţii asupra obiectului vizualizat şi apoi arată care obiect urmează, sau cum răspunde obiectul vizualizat la operatie; proiectantul repetă acest proces până când toate acţiunile asupra tuturor obiectelor sunt definite. Această tehnică funcţionează pentru secvenţializări prin item-uri (elemente) care au fost deja definite de proiectant.

DANA PETCU • LUCIAN CUCU

PRINCIPII ALE GRAFICII PE CALCULATOR

Coperta: Lucian Cucu

Consilier editor: Corina Bădulescu
Tehnoredactare computerizată: Dana Petcu
Bun de tipar: 20.09.1995
Coli de tipar:
Tiparul a fost executat la S.C. „HELICON" BANAT S.A.
Comanda nr.
ISBN 973 - 9015 - 51 - 4

D A N A P E T C U L U C I A N C U C U

P R I N C I P I I A L E G R A F I C I I
P E C A L C U L A T O R

E D I T U R A E X C E L S I O R

T i m i ş o a r a , 1 9 9 5

P r e f a ţ ă

Evoluţia omenirii, începând cu a doua jumătate a acestui secol, este profund marcată de impactul informaticii asupra cvasi-totalităţii domeniilor de activitate umană. Acest fenomen are o caracteristică foarte interesantă: se autoîntreţine. Într-adevăr, utilizarea calculatoarelor generează informaţie — din ce în ce mai multă şi mai diversă — care conduce la presiuni în scopul dezvoltării calitative a calculatoarelor, dar şi la înmulţirea numărului de utilizatori şi beneficiari ai acestora, care generează ... informaţie, şi ciclul se reia pe un nivel superior. În acest context, ultimii 10-15 ani au marcat o dinamică deosebită a aplicaţiilor informatice care fac uz (cel puţin sub forma interfeţelor utilizator) de grafică pe calculator. De la jocuri la programe de desenat, de la proiectarea asistată de calculator la vizualizarea datelor ştiinţifice, de la tehnoredactarea computerizată la animaţia pe calculator (şi lista ar putea continua...) grafica pe calculator este din ce în ce mai des utilizată.

Apare ca absolut firească şi necesară însuşirea (temeinică) a principiilor de proiectare şi realizare a aplicaţiilor grafice de către cei chemaţi să realizeze astfel de aplicaţii precum şi iniţierea (măcar şi sumară) în domeniu a potenţialilor utilizatori ai aplicaţiilor grafice.

Această carte îşi propune să umple un gol existent în literatura în limba română, gol populat în anii scurşi de la apariţia ultimei lucrări (relativ) similare (D. Dogaru, Elemente de grafică 3D, Ed. Ştiinţifică şi Enciclopedică, Bucureşti, 1988) de lucrări dedicate, mai ales, prezentării unor aplicaţii (editoare, programe pentru tehnoredactare, pachete de programe pentru proiectarea asistată de calculator) şi mai puţin prezentării principiilor generale.

Lucrarea se bazează în mare măsură pe rezultate recente (1990-1995) prezentate atât în cărţi de referinţă în domeniu (Foley şi alţii, Computer Graphics. Principles and Practice, 1992) cât şi în diverse rapoarte tehnice şi grupuri de discuţie din reţeaua Internet.

Cartea este structurată astfel:

Capitolul 1, intitulat sugestiv Concepte generale, prezintă pe scurt noţiunile de bază în grafica pe calculator, componentele hardware ale sistemelor grafice şi o clasificare a aplicaţiilor grafice.

Capitolul 2, Transformări geometrice, introduce noţiunile geometrice fundamentale pentru grafica pe calculator, precum transformările în spaţiul bidimensional sau tridimensional, sistemele carteziene de referinţă, proiecţiile paralele şi perspective.

Capitolul 3, sub titlul Reprezentări simple ale imaginii, tratează probleme elementare ale graficii pe calculator, precum trasarea primitivelor grafice (linii, cercuri, poligoane, etc), atributele primitivelor şi reprezentarea corpurilor tridimensionale pe suprafeţe bidimensionale.

Capitolul 4, Prelucrarea reprezentărilor simple, prezintă probleme complexe în construirea imaginilor: realism vizual, determinarea liniilor şi suprafeţelor ascunse, culoare, texturi, iluminare, umbre, fractali, animaţie.

Capitolul 5, se ocupă de una din metodele cele mai des utilizate în sintetizarea unor imagini de calitate deosebită, Metoda drumului optic, de la principiile de bază la detalii privind algoritmul.

Capitolul 6, Interfeţe utilizator, se ocupă de o problemă mai puţin tratată în literatura în limba română: principii de proiectare şi realizare a interfeţelor utilizator grafice.

Lucrarea de faţă se adresează în primul rând studenţilor de la facultăţile cu profil de informatică, dar şi tuturor utilizatorilor şi proiectanţilor de aplicaţii grafice.

Timişoara, 28.06.95 AUTORII

Domnului prof. dr. Ştefan Măruşteru

B i b l i o g r a fi e

[1] E. Angel, Computer Graphics, Addison-Wesley Publishing Company, New York, 1990.

[2] V. Baltac, D. Roman, N. Zegheru, D. Costăchescu, Calculatoare electronice, grafică interactivă şi prelucrarea imaginilor, Editura tehnică, Bucureşti, 1985.

[3] D. Dogaru, Metode noi în proiectare. Elemente de grafică 3-D., Editura Şiinţifică şi Pedagogică, Bucureşti 1988.

[4] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, Computer Graphics, Addison-Wesley Publishing Company, New York, 1992.

[5] A.S. Glassner (ed.), An Introduction to Ray Tracing, Academic Press, London, 1989.

[6] D. Hearn, M. Pauline Baker, Computer Graphics, Addison-Wesley Publishing Company, New York 1986.

[7] R.T. Stevens, C.D. Watkins, Advanced Graphics Programming in C and C++, M&T Books, New York, 1991.

[8] A. Tănăsescu, R. Constantinescu, I.D. Marinescu, L. Busuioc, Grafică asistată. Programe FORTRAN pentru reprezentări geometrice, Editura Tehnică, Bucureşti, 1989.

[9] M. Vlada, A. Posea, I. Nistor, C. Constantinescu, Grafică pe calculator în limbajele Pascal şi C, Editura Tehnică, Bucureşti, 1992.

C u p r i n s

INDEX