Grafică pe calculator
Note de curs
Facultatea de informatică
Universitatea "Alexandru Ioan Cuza", Iaşi

Lucian Ghirvu

anul universitar 2005-2006

Cuprins

Lista figurilor

Lista tabelelor

Capitolul 1

Introducere în grafica pe calculator

Obiectul graficii pe calculator (GPC) îl constituie crearea, memorarea şi manipularea modelelor şi imaginilor obiectelor. Modelele pot proveni din matematică, fizică, ştiinţe inginereşti, arhitectură, meteorologie, etc. .

Grafica pe calculator este, în mare măsură, *interactivă* : utilizatorul controlează conţinutul, structura şi modul de apariţie al obiectelor şi al imaginilor lor afişate, utilizând dispozitive de intrare (tastatură, mouse, ecrane sensibile la atingere[1]).

Evoluţia GPC :

- Anii '80 : hardware şi software costisitoare, dificil de utilizat,
- PC-urile[2] având ecrane cu grafică rastru încorporată (XEROX STAR, APPLE MACINTOSH, IBM PC) au introdus conceptul de imagini grafice bazate pe *hărţi de biţi*[3].

  Definiţia 1.1

  O *hartă de biţi* este o metodă de reprezentare a unei imagini grafice prin care aceasta este *divizată* într-o matrice de puncte[4] pe ecran. Fiecare pixel este memorat independent de ceilalţi pixeli şi i se poate atribui o culoare şi/sau intensitate (tot în mod independent de ceilalţi pixeli). ▪

- Următorul concept apărut în GPC este cel de *suprafaţă de lucru*[5]. Prin intermediul unui program gestionar de ferestre[6] utilizatorul poate crea, poziţiona, redimensiona, *ferestre*[7]. Ferestrele sunt terminale asociate unor aplicaţii. Acest concept a dus la modificarea interfeţelor text ale sistemelor de operare.

[1] *touch-sensitive panel* în engleză
[2] *Personal Computers* în engleză
[3] *Bitmaps* în engleză
[4] *Pixeli sau pels* în engleză
[5] *Desktop* în engleză
[6] *Window manager* în engleză
[7] suprafeţe dreptunghiulare pe ecran

1.1 Procesarea imaginii

Grafica pe calculator se referă la sinteza unor obiecte reale sau imaginare din modele computaţionale[8].

Procesarea imaginii este un domeniu înrudit dar *diferit* : tratează analiza scenelor sau reconstrucţia unor obiecte 2D/3D din imagini. Procesarea imaginilor este utilă în diverse domenii : recunoaştere aeriană, explorarea spaţiului cosmic, robotică industrială, tomografie, etc. .

Exemplul 1.1

Pe baza unei fotografii aeriene, să se reconstruiască obiectele 3D din zonă. ▪

Subdomeniile procesării imaginii sunt :

1. îmbunătăţirea imaginii[9] : îmbunătăţirea calităţii imaginii prin eliminarea zgomotului[10][11] sau prin îmbunătăţirea contrastului.

2. detectarea şi recunoaşterea modelelor[12] : detectarea şi clasificarea unor şabloane standard precum şi găsirea unor abateri de la aceste şabloane.

   Exemplul 1.2

   Tehnologia OCR[13]. ▪

3. analiza scenelor[14] şi vederea computerizată[15] : permit recunoaşterea şi reconstituirea modelului 3D al unei scene din mai multe imagini 2D.

   Exemplul 1.3

   Un robot industrial. ▪

Deşi GPC şi procesarea imaginii au început ca discipline separate astăzi există mai multe domenii în care se suprapun :

1. procesarea interactivă a imaginii : fotografiile scanate sunt modificate şi combinate cu altele (eventual cu imagini sintetice) înainte de publicare.

2. în GPC anumite operaţii simple de procesare a imaginii sunt utilizate pentru sinteza imaginii unui model.

---

[8] modele ce pot fi memorate în memoria unui calculator
[9] *Image Enhancement* în engleză
[10] *Noise* în engleză
[11] pixeli care lipsesc sau nu au legătură cu imaginea respectivă
[12] *pattern detection* în engleză
[13] *Optical Character Recognition* în engleză
[14] *Scene Analysis* în engleză
[15] *Computer Vision* în engleză

1.2 Avantajele GPC interactive

1. GPC reprezintă un mijloc natural de comunicare om-maşină datorită capacităţii umane de procesare a datelor grafice 2D/3D în mod rapid şi eficient.

2. Vizualizarea ştiinţifică : GPC a devenit un mijloc indispensabil de interpretare a datelor produse în supercomputere.

3. GPC interactivă a devenit cel mai important mod de producere a imaginilor[16] (atât ale unor obiecte concrete dar mai ales ale unor obiecte abstracte sau ale unor date care nu au proprietăţi geometrice, de ex. rezultate ale unor sondaje de opinie). Imaginile nu sunt numai statice ci pot fi şi dinamice, de ex. ale unor obiecte/fenomene care variază în timp : modelarea transformărilor feţei umane din copilărie şi pînă la o vârstă înaintată.

4. Utilizarea imaginilor dinamice este în mod deosebit efectivă atunci când utilizatorul poate controla viteza animaţiei, porţiunea din scenă care este vizualizată, numărul de detalii afişate, etc. Din aceste motive, în cadrul GPC interactive, s-au dezvoltat tehnologii hard/software pentru controlul de către utilizator a dinamicii mişcării[17] cât şi a dinamicii actualizărilor de imagine[18] :

   (a) În cadrul *dinamicii mişcării* obiectele pot fi mutate sau răsturnate[19] în raport cu un observator staţionar. Putem avea şi cazul invers : obiectele pot fi staţionare şi observatorul mobil (de ex. un simulator de zbor).

   (b) *Actualizarea imaginii* se referă la schimbarea vizibilă a formei, culorii sau a altor proprietăţi ale obiectelor vizualizate (de ex. afişarea deformării structurii unui avion în zbor).

5. GPC interactivă dinamică oferă numeroase posibilităţi de codificare şi de comunicare a informaţiei : forma 2D/3D a obiectelor dintr-o imagine, scala de gri/culoare a acestora precum şi variaţia în timp a acestor proprietăţi.

1.3 Utilizări reprezentative ale GPC

1. Interfeţe utilizator.

2. Scheme grafice[20] utilizate în economie, ştiinţă, tehnologie : histograme, hărţi grafice de tip *bar, pie*, ordonanţare a activităţilor, etc.

[16] de la inventarea TV şi a fotografiei
[17] *motion dynamics* în engleză
[18] *update dynamics* în engleză
[19] *tumbled* în engleză
[20] *charts* în engleză

3. Publicaţii electronice, birotică.

4. CAD - design-ul industrial cu ajutorul calculatorului[21] : în industria de automobile, avionică, procesoare, telefonie, etc.

5. Simularea şi animaţia pentru vizualizare ştiinţifică şi divertisment.

   Exemplul 1.4

   Pentru crearea unor desene animate desenatorul creează doar câteva cadre de referinţă, lăsând pe seama calculatorului crearea cadrelor de tranziţie. ▪

6. Artă şi comerţ : utilizarea GPC pentru crearea de reclame sau ghiduri computerizate în muzee.

7. Controloare de proces : GPC intervine în controlul proceselor industriale din rafinării, centrale electrice, etc. prin afişarea valorilor datelor preluate de la senzorii ataşaţi diverselor componente critice ale respectivelor sisteme (de ex. controlul traficului aerian).

8. Cartografie : realizarea de hărţi geografice, hărţi de relief, hărţi oceanografice, hărţi meteorologice, etc.

1.4 Clasificarea aplicaţiilor de GPC

Aplicaţiile de GPC se pot clasifica după mai multe criterii :

1. După *tipul (dimensionalitatea) obiectului* ce trebuie reprezentat şi *tipul de imagine* produsă : vezi tabelul 1.1, pag. 13.

2. După *tipul de interacţiune* ce determină gradul de control al utilizatorului asupra obiectului şi imaginii sale :

   (a) *Offline plotting* : afişarea imaginii pe baza unei baze de date apriorice,

   (b) *Interactive plotting* : utilizatorul controlează afişarea prin furnizarea de parametri. Afişarea interactivă este astfel şi un proces iterativ.

   (c) Afişarea obiectului în *timp real* (de ex. în simulatoarele de zbor).

   (d) *Design interactiv* : în care utilizatorul începe cu un ecran vid, defineşte noi obiecte apoi prelucrează imaginea obţinută.

3. După *rolul imaginii* sau în ce grad imaginea reprezintă un scop în sine sau serveşte unui scop : de ex., în cartografie sau în animaţie imaginea este produsul final, pe când în aplicaţiile gen CAD imaginea este cea a unui obiect construit sau analizat.

[21] *Computer Aided Design* în engleză

4. *Relaţiile temporale* şi *logice* între obiecte şi imaginile lor :

   - Utilizatorul poate lucra cu o singură imagine la un moment dat (cazul plotterelor),
   - Utilizatorul poate lucra cu o secvenţă variind în timp de imagini înrudite (ca în *motion dynamics*),
   - Utilizatorul poate lucra cu o colecţie structurală de obiecte (ca în cazul aplicaţiilor CAD).

tip obiect	reprezentare grafică
2D	desenare de linii
	imagini în nuanţe de gri
	imagini colorate
3D	desenare de linii
	desenare de linii cu diverse efecte
	imagini colorate cu diverse efecte

Tabelul 1.1: Clasificarea după dimensionalitate

1.5 Evoluţia tehnologiilor hardware şi software pentru GPC

1.5.1 Evoluţia tehnologiilor hardware

1. Dispozitive periferice de ieşire similare maşinilor de scris (cca 1950).

2. Apariţia tubului catodic CRT[22] (cca 1955).

3. Apariţia tastaturii, a creioanelor optice (inventate de Ivan Sutherland) în *Sketchpad Drawing System* (1963).

4. Apariţia domeniilor CAD (1964) şi CAM[23] (1981).

5. Apariţia calculatoarelor personale (PCs) cu interfeţe grafice (Apple, IBM PC) a redus dramatic costurile tehnologiilor hard/software pentru GPC.

[22] *Cathode Ray Tube* în engleză
[23] *Computed-Aided Manufacturing* în engleză

Evoluţia tehnologiilor hardware de ieşire

1. *Ecranul vectorial*[24] are arhitectura din figura 1.1, pag. 16. Procesorul de ecran (PE)[25] este similar unui dispozitiv periferic de ieşire. În memoria tampon a ecranului (ME)[26] sunt memorate liste de comenzi[27] de genul :

   - `move x0,y0`, care deplasează cursorul de ecran în poziţia (x_0, y_0),
   - `line x1, y1`, care trasează un segment de dreaptă având extremitatea iniţială poziţia curentă a cursorului iar extremitatea finală poziţia (x_1, y_1),
   - `char ABC`, care afişează şirul "ABC".

   Imaginea pe ecran (CRT) este obţinută prin execuţia de către PE a comenzilor din ME. Acest mod de obţinere a imaginii se numeşte baleiere aleatoare[28].

   Exemplul 1.5

   În figura 1.2, pag. 16, imaginea din partea stângă se obţine, pe un ecran vectorial printr-o succesiune de comenzi `move` şi `line`. ▪

   Limitele acestei tehnologii sunt date de capacitatea de memorare a ME precum şi de viteza PE. În plus trebuie ţinut cont şi de limitările tehnologice care impun reluarea execuţiei tuturor comenzilor din buffer cel puţin de 30 ori pe secundă şi aceasta deoarece fosforul utilizat în CRT îsi pierde luminozitatea în zecimi/sutimi de μs.

2. Ecranele DVST[29] sunt similare celor vectoriale dar nu au ME (şi deci şi procesul de reîmprospătare a imaginii este inutil în acest caz).

3. Apariţia configuraţiei *ecran-minicalculator* a fost următorul pas în dezvoltarea tehnologiilor hardware grafice de ieşire.

4. Apariţia în 1968 a ecranelor cu reîmprospătare a imaginii[30] prevăzute cu procesoare ce puteau efectua (fără a solicita CPU) diverse transformări geometrice : scalare, rotaţie, translarea punctelor şi a segmentelor de dreaptă în timp real, decupări 2D/3D[31] şi proiecţii paralele şi perspective.

[24] *vector, stroke, line drawing, calligraphic display* în engleză
[25] *Display processor* în engleză
[26] *Display buffer memory* în engleză
[27] *Display list, display program* în engleză
[28] *Random scan* în engleză
[29] *Direct view storage tube* în engleză
[30] *Refresh display hardware* în engleză
[31] *Clipping* în engleză

5. Dezvoltarea graficii rastru (cca 1970) a contribuit, mai mult decât orice altă tehnologie, la dezvoltarea domeniului GPC. *Ecranele rastru*[32] memorează *primitivele de ecran* (segmente de dreaptă, caractere, suprafeţe colorate uniform sau cu diverse modele[33]) într-o zonă de memorie tampon de reîmprospătare[34] sub forma *pixelilor* componenţi. Arhitectura unui ecran rastru este prezentată în figura 1.3, pag. 17.

   Imaginea pe un ecran rastru este formată pe baza *rastrului* (alcătuit din linii de rastru[35] ce reprezintă şiruri de pixeli; practic rastrul fiind memorat într-o matrice de pixeli ce reprezintă tot ecranul). Imaginea este baleiată (sau scanată) linie cu linie, secvenţial de către controller-ul video în ordinea sus, jos şi înapoi. Acest mod de baleiere este baleierea rastrului[36].

   În momentul afişării unui pixel, intensitatea fascicolului de electroni din CRT va fi modificată astfel încât să corespundă intensităţii pixelului respectiv. În cazul unui ecran în culori avem trei fascicole de electroni ale căror intensităţi corespund cu intensităţile celor trei componente ale culorii pixelului.

   Exemplul 1.6

   În figura 1.4, pag. 17 este prezentată imaginea pe un ecran rastru a figurii din partea stângă. ▪

   Dacă într-un ecran vectorial buffer-ul de reîmprospătare conţinea codul operaţiilor de trasare şi coordonatele finale, într-un ecran rastru întreaga imagine (de ex. 1024×1024 pixeli) trebuie să fie memorată explicit în buffer-ul de reîmprospătare. Termenul de *bitmap* se referă atât la buffer-ul de reîmprospătare cât şi la matricea valorilor pixelilor.

6. Apariţia memoriilor RAM a contribuit decisiv la impunerea modelului de ecran rastru[37].

   Utilizând memorii RAM au apărut ecrane rastru *monocrom*[38] (i.e., cu 2 culori : alb/negru, negru/verde, portocaliu/verde). Bitmap-ul acestui tip de ecran conţine 1 bit/pixel şi deci, dacă un ecran monocrom are o rezoluţie de 1024×1024 pixeli atunci bitmap-ul corespunzător are 2^{20} biţi.

   Ulterior au apărut ecrane *policrome* cu 8 biţi/pixel (permiţând 256 culori pentru un pixel) sau 24 biţi/pixel (permiţând 16 milioane de culori pentru un pixel).

---

[32] *Raster displays* în engleză
[33] *patterns* în engleză
[34] *Refresh buffer* în engleză
[35] *raster lines* în engleză
[36] *Raster scan* în engleză
[37] în faţa ecranului vectorial
[38] *Bilevel* în engleză

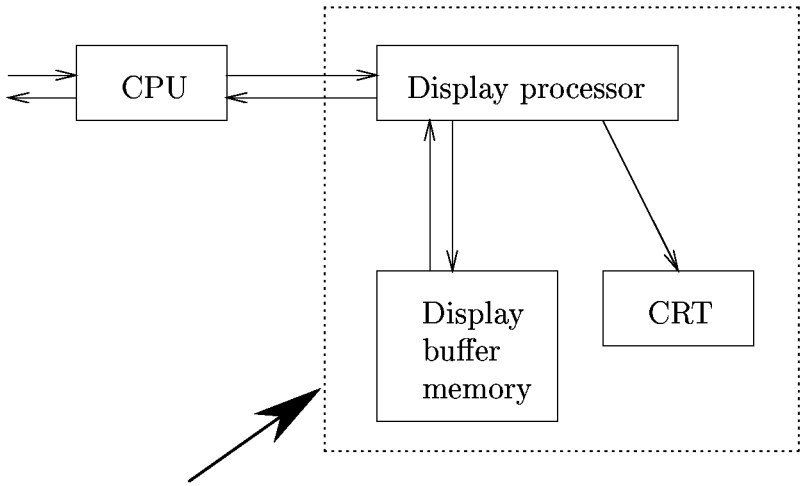


Figura 1.1: Arhitectura unui ecran vectorial

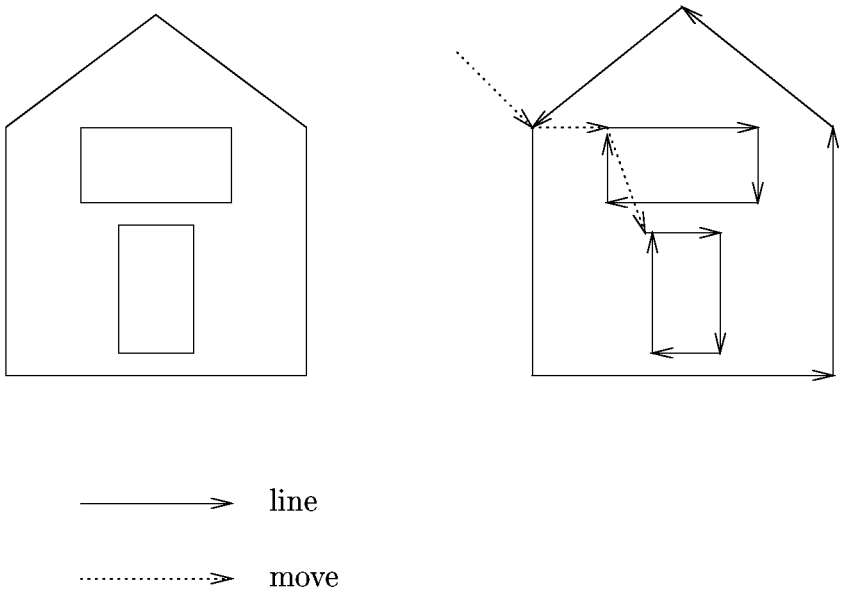


Figura 1.2: Obţinerea unei imagini pe un ecran vectorial

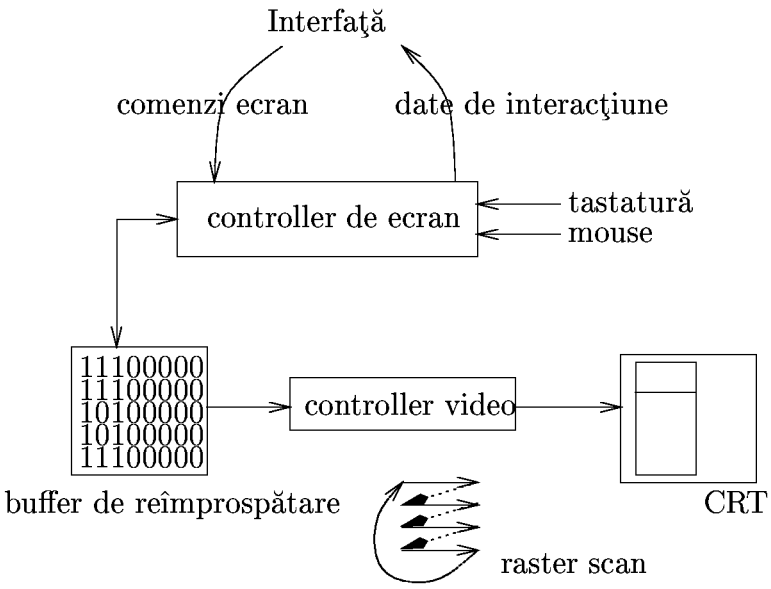


Figura 1.3: Arhitectura unui ecran rastru

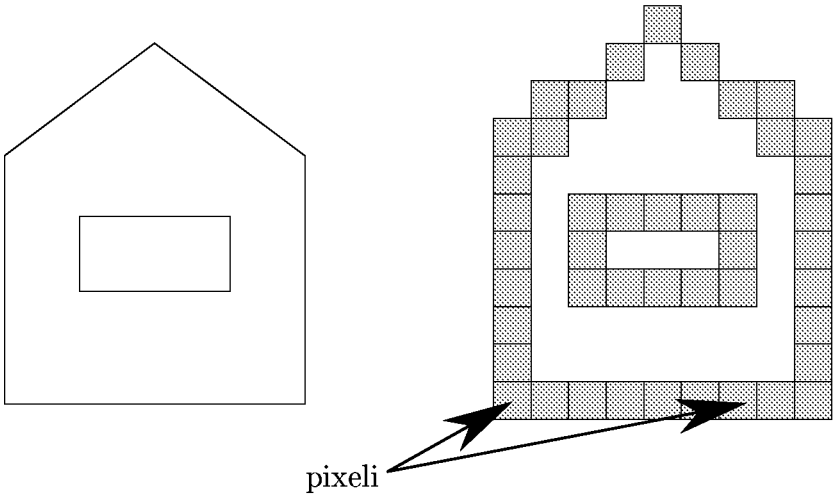


Figura 1.4: Obţinerea unei imagini pe un ecran rastru

Observaţia 1.1

Termenul de *bitmap* se aplică doar ecranelor monocrome, în cazul ecranelor policrome utilizăm termenul de *pixmap*.

În cazul în care dorim să distingem între cele 2 semnificaţii ale termenului bitmap (pixmap) : conţinutul bufferului de reîmprospătare şi bufferul de reîmprospătare vom utiliza pentru bufferul de reîmprospătare termenul de *buffer cadru*[39]. ▪

Observaţia 1.2

Avantajele GPC utilizând ecrane rastru faţă de GPC utilizând ecrane vectoriale sunt :

- Costuri scăzute (datorate costurilor scăzute ale memoriilor RAM).
- Posibilitatea de a colora uniform sau cu diferite modele zone compacte pe ecran.
- Procesul de reîmprospătare al imaginilor este independent de complexitatea imaginii (i.e., de numărul de segmente de dreaptă, de numărul de poligoane, etc.) iar tehnologiile hardware actuale permit citirea cel puţin o dată a fiecărui pixel din bufferul cadru într-un ciclu de reîmprospătare, lucru care nu este posibil întotdeauna în cazul unor imagini complicate afişate pe un ecran vectorial.

Totuşi GPC utilizând ecrane rastru are şi o serie de *dezavantaje* faţă de GPC utilizând ecrane vectoriale :

- Reprezentarea segmentelor de dreaptă sau a poligoanelor se face, în general, prin vârfurile lor şi în concluzie pentru a fi afişate pe un ecran rastru trebuie să existe software specializat care să calculeze pixelii corespunzători unui segment de dreaptă sau poligon. În general această conversie se efectuează utilizând un procesor dedicat RIP[40].
- Deoarece reprezentarea unei primitive grafice pe un ecran rastru necesită o conversie, reprezentarea grafică a unor fenomene dinamice în timp real este costisitoare pentru ecranele rastru.
- Reprezentarea unor curbe continue este aproximată prin pixeli pentru ecranele rastru. De aici rezultă aspectul zimţat, colţuros al acestor reprezentări.

▪

Evoluţia tehnologiilor hardware de intrare

Primele dispozitive hardware grafice de intrare au fost *creioanele optice* pentru ecranele vectoriale. Ulterior au apărut : *mouse-ul* (Doug Engelbart 1968), *tabletele grafice*[41], *ecranele sensibile la atingere*[42].

[39] *frame buffer* în engleză
[40] *Raster image processor* în engleză
[41] *Tablet data* în engleză
[42] *Touch Sensitive Panels* în engleză

1.5.2 Evoluţia tehnologiilor software grafice (şi al standardelor grafice)

Tehnologiile software grafice s-au dezvoltat plecând de la software scris pentru un singur tip de ecran şi într-un limbaj de programare de nivel scăzut (chiar limbaj de asamblare sau cod maşină) şi ajungând la software independent de dispozitivul grafic de ieşire şi dezvoltat în limbaje de programare de nivel înalt (C, Pascal, Fortran, etc.).

Standardele grafice au apărut începând cu anul 1975 (cca) :

- 1977, 1979 : 3D Core Graphics System.
- GKS (Graphical Kernel System) : este standard ANSI (1985).
- GKS 3D (1988).
- PHIGS (Programmer's Hierarchical Interactive Graphic System) : 1988.
- SRGP (Simple Raster Graphics Package).
- SPHIGS (Simple PHIGS).

1.6 Conceptele GPC interactive

Majoritatea sistemelor grafice interactive au structura din figura 1.5, pag. 20 :

1. *Modelul aplicaţiei*[43] reprezintă datele sau obiectele ce vor fi reprezentate pe ecran.
2. *Programul aplicaţiei*[44] creează, memorează în sau regăseşte din modelul aplicaţiei.
3. Programul aplicaţiei gestionează intrările de la utilizator şi produce *privelişti*[45] ale modelului trimiţând *sistemului grafic* o serie de comenzi grafice de ieşire ce conţin o descriere geometrică detaliată a imaginii ce va fi afişată cât şi atributele ce descriu modul în care va apărea respectiva imagine.

   Sistemul grafic poate fi văzut realizând o *transformare de ieşire* ale obiectelor din modelul aplicaţiei într-o priveliştе a respectivului model sau realizând o *transformare de intrare* a acţiunilor utilizatorului în intrări pentru programul aplicaţiei.

Designer-ul unui sistem grafic interactiv trebuie să specifice :

- Clasele de obiecte ce vor fi generate şi reprezentate grafic.
- Modalităţile prin care utilizatorul şi programul aplicaţiei vor interacţiona pentru a crea şi modifica modelul şi reprezentarea sa vizuală.

[43] *Application model* în engleză
[44] *Application program* în engleză
[45] *Views* în engleză

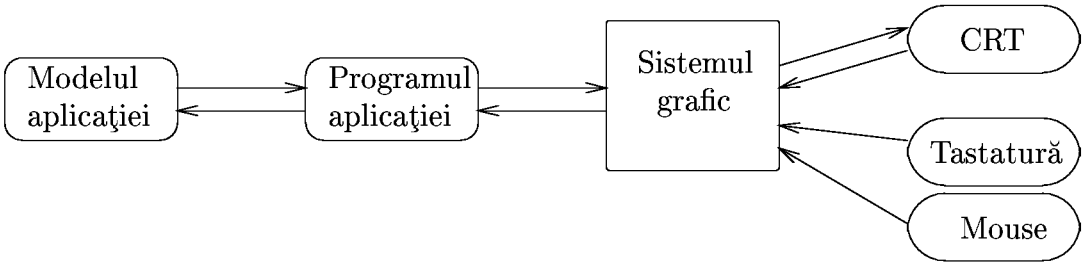


Figura 1.5: Structura unui sistem grafic interactiv

1.6.1 Modelarea aplicaţiei

Definiţia 1.2

Modelul aplicaţiei cuprinde datele, obiectele şi relaţiile dintre ele relevante pentru partea de afişare şi de interacţiune a programului aplicaţiei şi pentru orice modul de postprocesare negrafic (de ex. module care realizează simularea unui model populaţional). ▪

Exemplul 1.7

Programul PAINT (WINDOWS^{TM}) este un program interactiv de realizare de imagini. Utilizatorul poate modifica în mod direct pixelii din bitmap/pixmap. ▪

Modelul aplicaţiei poate fi de mai multe feluri :

1. Model *inexistent* (de ex. în programul PAINT).

2. Model alcătuit din *combinaţii* de descrieri de date şi de proceduri. Modelul datelor poate fi un tablou sau o listă înlănţuită sau chiar o bază de date relaţională.

Modelul aplicaţiei *memorează* de obicei :

- *Primitive* (puncte, segmente de dreaptă, poligoane în 2D/3D, poliedre, suprafeţe 3D) care definesc *forma* componentelor obiectului.

- *Atribute* ale obiectului : culoare, stil (pentru segmente de dreaptă), textură (pentru suprafeţe).

- *Relaţii de conexiune* şi *date de poziţionare* care descriu modul de ajustare al obiectelor.

Putem efectua o *clasificare* a modelelor aplicaţiilor în funcţie de *gradul de modelare geometrică* al obiectelor :

1. Modele în care obiectele sunt descrise în întregime în termeni geometrici (de ex. poliedrele componente).

2. Modele în care descrierea geometrică a obiectelor nu este necesară (de ex. foi de calcul[46]).

3. Modele pentru care descrierea geometrică poate fi obţinută *on-the-fly* (de ex. grafurile orientate se memorează prin liste de adiacenţă şi doar când e nevoie se poate genera descrierea geometrică).

1.6.2 Programarea aplicaţiei

Programul aplicaţiei *creează modelul aplicaţiei* fie *aprioric* (ca rezultat al unor calcule anterioare) sau *interactiv* (când utilizatorul ghidează procesul de construcţie al modelului aplicaţiei pas cu pas; în acest caz programul aplicaţiei trebuie să poată furniza privelişti ale modelului parţial).

Deoarece modelele sunt specifice diverselor aplicaţii şi sunt create independent de un anumit tip de ecran, programul aplicaţiei trebuie să *convertească* o porţiune din modelul ce trebuie vizualizat din reprezentarea internă geometrică (fie explicit memorată în model, fie derivată on-the-fly) în apeluri de proceduri sau comenzi pe care sistemul grafic le utilizează pentru a crea o imagine.

Procesul de *conversie* are două etape :

- Programul aplicaţiei extrage porţiunile ce trebuie vizualizate utilizând criterii de interogare sau selecţie pentru baza de date în care este memorat modelul aplicaţiei.

- Porţiunile extrase sunt convertite în formatul de intrare al sistemului grafic. Datele extrase în prima etapă trebuie să fie geometrice sau să fie convertite într-un format geometric înţeles de către sistemul grafic. Dacă modelul conţine primitive geometrice care nu sunt suportate de către sistemul grafic atunci programul aplicaţiei trebuie să le convertească în primitivele suportate de către sistemul grafic.

Sistemul grafic constă într-un set de proceduri corespunzând diverselor primitive şi atribute. Acestea sunt colectate în biblioteci grafice (apelabile din limbaje de nivel înalt : C, Pascal). Sistemul grafic permite deasemenea şi abstractizarea diverselor detalii de implementare a sistemului grafic prin crearea *dispozitivelor logice de ecran*[47]. Pot fi abstractizate detalii de implementare cum ar fi :

- care parte anume din generarea de imagini se efectuează cu ajutorul tehnologiilor hardware şi care se efectuează de către sistemul grafic,

- mouse-ul, tabela de date, ecranul senzitiv, joystick-ul pot fi tratate ca dispozitive de intrare logice de localizare ce returnează o adresă (x, y) de pe ecran. În acest caz programul aplicaţiei poate cere sistemului grafic să testeze dispozitivele de intrare sau să

[46] *Spreadsheets* în engleză
[47] *Logical Display Device* în engleză

aştepte până când un eveniment este generat în momentul activării de către utilizator a unui dispozitiv de intrare. Cu aceste valori de intrare (obţinute prin testare[48] sau prin aşteptarea unui eveniment) programul aplicaţiei poate gestiona interacţiuni cu utilizatorul care modifică modelul sau ecranul sau care îi schimbă modul de operare.

1.6.3 Tratarea interacţiunilor

Tratarea interacţiunilor în programul aplicaţiei se face utilizând o buclă de tratare a evenimentelor[49] (reprezentată în tabelul 1.2, pag. 22). Aceasta este un automat finit determinist cu o singură stare (de aşteptare) şi tranziţii generate de evenimentele cauzate de intrările utilizator.

```
begin
  1. generarea ecranului iniţial (derivată din modelul aplicaţiei)
  while(!quit)
    2. trecerea în modul de selectare a comenzilor
    3. aşteptare selecţie utilizator
    switch(selecţie)
      4. selectarea unui proces pentru completarea (finalizarea)
         comenzii sau procesarea completă de către acest proces a comenzii
      5. actualizarea modelului şi a ecranului
    endswitch
  endwhile
end
```

Tabelul 1.2: Buclă de tratare a evenimentelor

Observaţia 1.3

În bucla de tratare a evenimentelor trebuie reţinut că *nu* pot exista modificări pe ecran fără a exista în prealabil în modelul aplicaţiei (coerenţa dintre modelul aplicaţiei şi ecran trebuie menţinută).

Această observaţie nu se referă şi la cazul în care modelul aplicaţiei şi imaginea de pe ecran sunt identice (de ex. în cazul programul PAINT sau în aplicaţiile de îmbunătăţire a imaginilor). ▪

Observaţia 1.4

Sistemul grafic nu are nici un fel de responsabilitate în construirea sau modificarea modelului (fie iniţial, fie ca răspuns la interacţiunea cu utilizatorul). Singura sa responsabilitate constă în crearea imaginilor din descrieri geometrice şi de a transmite datele de intrare de la utilizator. ▪

[48] *sampling* în engleză
[49] *event-driven loop* în engleză

1.7 Sumar

- Interfeţele grafice au substituit pe cele în mod text ca standarde în interacţiunea utilizator-calculator.
- Exprimarea *grafică* a ideilor, datelor în cele mai diverse domenii este realizată prin intermediul GPC.
- În anii '80 au apărut calculatoarele personale (PCs) cu grafică bitmap şi cca un deceniu mai târziu procesoarele care realizează în timp real animaţii 3D cu imagini policrome.

Capitolul 2

Spaţii vectoriale

În cele ce urmează reamintim câteva noţiuni legate de spaţiile vectoriale şi de aplicaţiile liniare.

Definiţia 2.1

Se numeşte *spaţiu vectorial* tuplul $(K, V, +, \cdot)$, unde $+$ şi $\cdot$ sunt două funcţii : $+ : V \times V \to V$ şi $\cdot : K \times V \to V$.

Elementele mulţimii V se numesc *vectori* iar elementele mulţimii K se numesc *scalari*. Operaţiile de adunare a vectorilor precum şi de înmulţire a vectorilor cu scalari au următoarele proprietăţi :

SV_1 adunarea vectorilor este asociativă : $(\forall \vec{u}, \vec{v}, \vec{w} \in V)((\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w}))$,

SV_2 adunarea vectorilor este comutativă : $(\forall \vec{u}, \vec{v} \in V)(\vec{u} + \vec{v} = \vec{v} + \vec{u})$,

SV_3 în V există vectorul nul : $(\exists \vec{0} \in V)(\forall \vec{v} \in V)(\vec{v} + \vec{0} = \vec{v})$,

SV_4 existenţa vectorului opus : $(\forall \vec{v} \in V)(\exists - \vec{v} \in V)(\vec{v} + (-\vec{v}) = \vec{0})$,

SV_5 înmulţirea vectorilor cu scalari este asociativă : $(\forall a, b \in K)(\forall \vec{v} \in V)(a \cdot (b \cdot \vec{v}) = (a \cdot b) \cdot \vec{v})$,

SV_6 în K există scalarul unitate : $(\exists 1 \in K)(\forall \vec{v} \in V)(1 \cdot \vec{v} = \vec{v})$,

SV_7 adunarea vectorilor este distributivă : $(\forall a \in K)(\forall \vec{u}, \vec{v} \in V)(a \cdot (\vec{u} + \vec{v}) = a \cdot \vec{u} + a \cdot \vec{v})$,

SV_8 înmulţirea vectorilor cu scalari este distributivă : $(\forall a, b \in K)(\forall \vec{v} \in V)((a + b) \cdot \vec{v} = a \cdot \vec{v} + b \cdot \vec{v})$.

▪

Exemplul 2.1

Un exemplu de spaţiu vectorial este $(\mathbf{R}, \mathbf{R}^n, +, \cdot)$, unde $(a_1, a_2, \ldots, a_n) + (b_1, b_2, \ldots, b_n) = (a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n)$ şi $a \cdot (a_1, a_2, \ldots, a_n) = (a \cdot a_1, a \cdot a_2, \ldots, a \cdot a_n)$. ▪

În cele ce urmează definim un spaţiu vectorial izomorf[1] cu $(\mathbf{R}, \mathbf{R}^3, +, \cdot)$: spaţiul vectorial tridimensional $(\mathbf{R}, V_3)$.

Elementele mulţimii V_3 se definesc prin intermediul unei relaţii de echivalenţă R, definită pe mulţimea segmentelor orientate din plan (spaţiu).

Un *segment orientat* este perechea (A, B), unde A, B sunt puncte din plan (spaţiu). Îl mai notăm şi $\overrightarrow{AB}$. Dreapta AB este dreapta suport, A este originea iar B extremitatea segmentului orientat $\overrightarrow{AB}$.

Fie $\overrightarrow{AB}$, $\overrightarrow{CD}$ două segmente orientate. Atunci $(\overrightarrow{AB}, \overrightarrow{CD}) \in R$ dacă au aceeaşi direcţie, sens şi lungime, unde :

1. $\overrightarrow{AB}$ şi $\overrightarrow{CD}$ au *aceeaşi direcţie* dacă dreptele suport AB şi CD coincid sau sunt paralele,
2. segmentele orientate $\overrightarrow{AB}$ şi $\overrightarrow{CD}$ (care au aceeaşi direcţie) au *acelaşi sens* dacă B şi D sunt situate în acelaşi semiplan (al planului ABC) determinat de dreapta AC,
3. *lungimea* (modulul, norma) segmentului orientat $\overrightarrow{AB}$ (notată $|\overrightarrow{AB}|$) este distanţa ($\in \mathbf{R}$) între punctele A şi B.

Un *vector* $\vec{v}$ este un reprezentant al unei clase de echivalenţă $\bar{v}$ determinate de relaţia R.
Se pot menţiona câţiva vectori particulari :

1. vectorul nul $\vec{0}$ este reprezentantul clasei de echivalenţă $\{\overrightarrow{AA} |\ A$ punct din plan (spaţiu)$\}$.
2. vectorul unitate (versor) este un vector $\vec{v}$ cu proprietatea că $|\ \vec{v}\ | = 1$.

Definim în plan (spaţiu) un punct O numit *origine*. Atunci, fiecare punct A din plan (spaţiu) determină *vectorul de poziţie* $\overrightarrow{OA}$. Se poate observa că putem defini o bijecţie între mulţimea punctelor din plan (spaţiu) şi mulţimea vectorilor din V_3.

Definim aplicaţiile $+ : V_3 \times V_3 \to V_3$ şi $\cdot : \mathbf{R} \times V_3 \to V_3$ după cum urmează :

1. Fie vectorii $\overrightarrow{PQ} \in \overline{AB}$ şi $\overrightarrow{QR} \in \overline{CD}$. Atunci definim adunarea vectorilor (vezi şi figura 2.1, pag. 27) astfel : $\overrightarrow{AB} + \overrightarrow{CD} \in \overline{PR}$ (regula triunghiului). Fără a restrânge generalitatea spunem că $\overrightarrow{AB} + \overrightarrow{CD} = \overrightarrow{PR}$.

Proprietatea 2.1

Adunarea vectorilor are proprietăţile $SV_{1,2,3,4}$ din definiţia 2.1, pag. 25. ▪

Fie $\overrightarrow{AB} \in \overline{AB}$. Inversul său (vectorul opus) este $\overrightarrow{BA} \in \overline{BA}$.

[1] există o bijecţie între cele 2 spaţii vectoriale care conservă operaţiile

2. Fie $k \in \mathbf{R}$ şi $\vec{v} \in V_3$. Atunci $k \cdot \vec{v}$ este :

   (a) vectorul $\vec{0}$ dacă $k = 0$ sau $\vec{v} = \vec{0}$,

   (b) vectorul care are aceeaşi direcţie cu $\vec{v}$ şi acelaşi sens (dacă $k > 0$) sau sens diferit (dacă $k < 0$) şi lungime $k \cdot |\vec{v}|$.

Proprietatea 2.2

Înmulţirea vectorilor cu scalari are proprietăţile $SV_{5,6,7,8}$ din definiţia 2.1, pag. 25. ▪

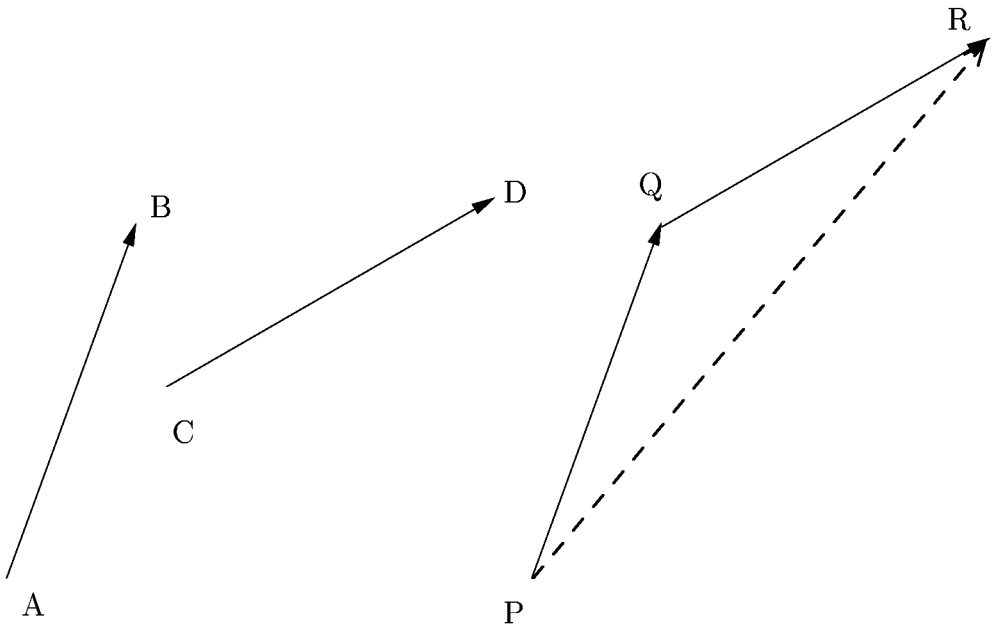


Figura 2.1: Adunarea vectorilor

Definiţia 2.2

Doi vectori $\vec{u}$ şi $\vec{v}$ sunt *coliniari* dacă au aceeaşi direcţie. ▪

Proprietatea 2.3

Vectorii $\vec{u}$ şi $\vec{v}$ sunt coliniari dacă $(\exists \lambda \in \mathbf{R}^*)(\vec{u} = \lambda \cdot \vec{v})$. ▪

Pentru evidenţierea izomorfismului între spaţiile vectoriale $(\mathbf{R}, \mathbf{R}^3, +, \cdot)$ şi $(\mathbf{R}, V_3, +, \cdot)$ vom introduce un sistem de coordonate carteziene $Oxyz$ (astfel încât axele Ox, Oy şi Oz sunt perpendiculare două câte două).

Proprietatea 2.4

$(\forall \vec{v} \in V_3)(\exists \vec{v}_x, \vec{v}_y, \vec{v}_z \in V_3)(\vec{v} = \vec{v}_x + \vec{v}_y + \vec{v}_z)$ şi mai mult, $\vec{v}_x$, $\vec{v}_y$ şi $\vec{v}_z$ sunt unici cu această proprietate. Direcţiile vectorilor $\vec{v}_x$, $\vec{v}_y$ şi $\vec{v}_z$ sunt axele Ox, Oy şi Oz. Dacă $\vec{v} = \overrightarrow{OA}$ atunci $\vec{v}_x = \overrightarrow{OA_x}$, $\vec{v}_y = \overrightarrow{OA_y}$ şi $\vec{v}_z = \overrightarrow{OA_z}$, unde $AA_x \perp OA_x$, $AA_y \perp OA_y$ şi $AA_z \perp OA_z$ iar $A_x \in Ox$, $A_y \in Oy$, $A_z \in Oz$. ▪

Fie $\vec{i}, \vec{j}, \vec{k}$ versorii sistemului de coordonate (vectorii unitate ai direcţiilor Ox, Oy, Oz). Atunci $(\exists v_x, v_y, v_z \in \mathbf{R})(\vec{v}_x = v_x \cdot \vec{i} \wedge \vec{v}_y = v_y \cdot \vec{j} \wedge \vec{v}_z = v_z \cdot \vec{k})$. Numim $\vec{v}_x, \vec{v}_y, \vec{v}_z$ *componentele* vectorului $\vec{v}$ în raport cu sistemul de coordonate $Oxyz$ iar v_x, v_y, v_z *coordonatele* vectorului $\vec{v}$ în raport cu sistemul de coordonate $Oxyz$.

Fie $A_i(a_i, b_i, c_i)$, $1 \leq i \leq 2$ două puncte astfel încât $\vec{v} = \overrightarrow{A_1A_2}$. *Forma hipercomplexă* a vectorului $\overrightarrow{A_1A_2}$ este $\overrightarrow{A_1A_2} = (a_2 - a_1) \cdot \vec{i} + (b_2 - b_1) \cdot \vec{j} + (c_2 - c_1) \cdot \vec{k}$ iar $|\overrightarrow{A_1A_2}| = \sqrt{(a_2 - a_1)^2 + (b_2 - b_1)^2 + (c_2 - c_1)^2}$.

Deci există o corespondenţă bijectivă ϕ între spaţiile vectoriale $(\mathbf{R}, \mathbf{R}^3, +, \cdot)$ şi $(\mathbf{R}, V_3, +, \cdot)$ dată de faptul că oricărui vector $\vec{v} \in V_3$ îi corespunde în mod unic tripletul coordonatelor sale (v_x, v_y, v_z) şi viceversa, oricărui triplet (a, b, c) din $\mathbf{R}^3$ îi corespunde în mod unic vectorul $a \cdot \vec{i} + b \cdot \vec{j} + c \cdot \vec{k}$ din V_3.

Se poate arăta deasemeni şi că $\phi(\vec{u} + \vec{v}) = \phi(\vec{u}) + \phi(\vec{v})$ şi $\phi(a \cdot \vec{u}) = a \cdot \phi(\vec{u})$, $\forall \vec{u}, \vec{v} \in V_3$, $\forall a \in \mathbf{R}$.

Deci, dacă $\vec{a} = a_1 \cdot \vec{i} + a_2 \cdot \vec{j} + a_3 \cdot \vec{k}$ şi $\vec{b} = b_1 \cdot \vec{i} + b_2 \cdot \vec{j} + b_3 \cdot \vec{k}$ şi $p \in \mathbf{R}$ atunci $\vec{a} + \vec{b} = (a_1 + b_1) \cdot \vec{i} + (a_2 + b_2) \cdot \vec{j} + (a_3 + b_3) \cdot \vec{k}$ şi $p \cdot \vec{a} = (p \cdot a_1) \cdot \vec{i} + (p \cdot a_2) \cdot \vec{j} + (p \cdot a_3) \cdot \vec{k}$.

Exerciţiul 2.1

Fie $\vec{a} = 2\vec{i} + (\frac{1}{2})\vec{j} - \vec{k}$ şi $\vec{b} = -3\vec{i} + 2\vec{j} + 5\vec{k}$. Să se calculeze $\vec{a} + \vec{b}$, $-\vec{b}$, $\vec{a} - \vec{b}$, $d\vec{a}$. ▪

În concluzie vom identifica un vector cu coordonatele sale : $\vec{i}$ cu $(1, 0, 0)$, $\vec{j}$ cu $(0, 1, 0)$ şi $\vec{k}$ cu $(0, 0, 1)$ iar $\vec{v}$ cu (v_x, v_y, v_z).

Definiţia 2.3 (Produs scalar)

Produsul scalar este aplicaţia $\cdot : V_3 \times V_3 \to \mathbf{R}$ definită astfel :

$$\vec{u} \cdot \vec{v} = \begin{cases} 0 & , \quad \text{dacă } \vec{u} = \vec{0} \vee \vec{v} = \vec{0}, \\ |\vec{u}||\vec{v}| \cos \widehat{\vec{u}, \vec{v}} & , \quad \text{dacă } \vec{u} \neq \vec{0} \wedge \vec{v} \neq \vec{0} \end{cases}$$

, unde $\vec{u}, \vec{v} \in V_3$ şi $\widehat{\vec{u}, \vec{v}}$ este cel mai mic dintre unghiurile determinate de dreptele suport ale vectorilor $\vec{u}$ şi $\vec{v}$. ▪

Proprietatea 2.5

Produsul scalar are următoarele proprietăţi :

1. este comutativ,

2. $\vec{i}\cdot\vec{i}=\vec{j}\cdot\vec{j}=\vec{k}\cdot\vec{k}=1$ (deoarece unghiul dintre vectori este 0°),

3. $\vec{i}\cdot\vec{j}=\vec{i}\cdot\vec{k}=\vec{j}\cdot\vec{k}=0$ (deoarece unghiul dintre vectori este 90°),

4. dacă $\vec{a}=a_1\cdot\vec{i}+a_2\cdot\vec{j}+a_3\cdot\vec{k}$ şi $\vec{b}=b_1\cdot\vec{i}+b_2\cdot\vec{j}+b_3\cdot\vec{k}$ atunci $\vec{a}\cdot\vec{b}=\sum_{i=1}^{3}a_ib_i$,

5. $\cos\widehat{\vec{u},\vec{v}}=\frac{\vec{u}\cdot\vec{v}}{|\vec{u}||\vec{v}|}=\frac{\sum_{i=1}^{3}a_ib_i}{\sqrt{(\sum_{i=1}^{3}a_i^2)(\sum_{i=1}^{3}b_i^2)}}$.

■

Exerciţiul 2.2

Fie $\vec{a}=3\vec{i}-4\vec{j}$ şi $\vec{b}=\vec{i}+2\vec{j}-2\vec{k}$. Care este unghiul $\widehat{\vec{u},\vec{v}}$? ■

Definiţia 2.4 (Produs vectorial)

Produsul vectorial este aplicaţia $\times : V_3\times V_3\to V_3$ definită astfel : $\vec{u}\times\vec{v}$ este $\vec{0}$ dacă $(\vec{u}=\vec{0}\vee\vec{v}=\vec{0})$ şi, în cazul în care $(\vec{u}\neq\vec{0}\wedge\vec{v}\neq\vec{0})$, este vectorul $\vec{w}$ definit astfel încât :

1. $|\vec{w}|=|\vec{u}||\vec{v}|\sin\widehat{\vec{u},\vec{v}}$,

2. $\vec{w}\perp\vec{u}$ şi $\vec{w}\perp\vec{v}$,

3. $(\vec{u},\vec{v},\vec{w})$ constituie un sistem orientat drept : dacă efectuăm cea mai scurtă rotaţie astfel încât să ducem $\vec{u}$ peste $\vec{v}$ atunci degetul mare arată sensul vectorului $\vec{w}$ (sau dacă degetul mare al mâinii drepte arată sensul $\vec{u}$ iar degetul arătător al mâinii drepte arată sensul $\vec{v}$ atunci palma arată sensul $\vec{w}$).

■

Proprietatea 2.6

Produsul vectorial are următoarele proprietăţi :

1. $(\forall\vec{u},\vec{v}\in V_3)(\vec{u}\times\vec{v}=-\vec{v}\times\vec{u})$,

2. $\vec{i}\times\vec{i}=\vec{j}\times\vec{j}=\vec{k}\times\vec{k}=0$ (deoarece unghiul dintre vectori este 0°),

3. $\vec{i}\times\vec{j}=\vec{k}$, $\vec{j}\times\vec{k}=\vec{i}$, $\vec{k}\times\vec{i}=\vec{j}$ (deoarece unghiul dintre vectori este 90°),

4. $\vec{u}\times\vec{v}=\begin{vmatrix}\vec{i}&\vec{j}&\vec{k}\\u_x&u_y&u_z\\v_x&v_y&v_z\end{vmatrix}$,

■

Exerciţiul 2.3
Fie $\vec{a} = 5\vec{\imath} - 3\vec{\jmath} + \vec{k}$ şi $\vec{b} = -\vec{\imath} - \vec{\jmath} - 2\vec{k}$. Calculaţi $\vec{u} \times \vec{v}$. ▪

Proprietatea 2.7
Doi vectori $\vec{u}$ şi $\vec{v}$ sunt ortogonali (direcţiile lor sunt drepte perpendiculare) dacă produsul lor scalar este 0 : $\vec{u} \cdot \vec{v} = 0$. ▪

Definiţia 2.5 (Dependenţă liniară)
Fie $(\vec{v}_i)_{1 \leq i \leq n}$ vectori din spaţiul vectorial (K, V) şi $\vec{v} \in V$. Atunci vectorul $\vec{v}$ este *liniar dependent* de vectorii $(\vec{v}_i)_{1 \leq i \leq n}$ dacă $(\exists (a_i \in K)_{1 \leq i \leq n})(\vec{v} = \sum_{i=1}^{n} a_i \vec{v}_i)$. Mai spunem că $\vec{v}$ este o combinaţie liniară de vectorii $(\vec{v}_i)_{1 \leq i \leq n}$. ▪

Observaţia 2.1
În spaţiul vectorial $(\mathbf{R}, V_3)$ orice vector $\vec{v}$ depinde liniar de vectorii $\vec{\imath}$, $\vec{\jmath}$, $\vec{k}$. ▪

Definiţia 2.6 ((In)dependenţă liniară)
Un sistem $(\vec{v}_i)_{1 \leq i \leq n}$ de vectori din spaţiul vectorial (K, V) este *liniar dependent* dacă $(\exists (a_i \in K)_{1 \leq i \leq n})(\sum_{i=1}^{n} a_i \vec{v}_i = \vec{0})$ şi este *liniar independent* dacă $(\forall (a_i \in K)_{1 \leq i \leq n})(\sum_{i=1}^{n} a_i \vec{v}_i = \vec{0} \iff (\forall 1 \leq i \leq n)(a_i = 0))$. ▪

Proprietatea 2.8
Vectorii $\vec{\imath}$, $\vec{\jmath}$, $\vec{k}$ sunt liniar independenţi. ▪
Demonstraţie
Fie $(a_i \in \mathbf{R})_{1 \leq i \leq 3}$ astfel încât $a_1\vec{\imath} + a_2\vec{\jmath} + a_3\vec{k} = \vec{0}$. Atunci, efectuând înmulţirea scalară la dreapta cu $\vec{\imath}$, rezultă $a_1\vec{\imath}\vec{\imath} + a_2\vec{\jmath}\vec{\imath} + a_3\vec{k}\vec{\imath} = \vec{0}\vec{\imath} = 0$ şi deci $a_1 = 0$. Analog se arată $a_2 = a_3 = 0$. □

Proprietatea 2.9
$(\forall \vec{v} \in V_3)(\exists (a_i \in \mathbf{R})_{1 \leq i \leq 3})(\vec{v} = a_1\vec{\imath} + a_2\vec{\jmath} + a_3\vec{k})$. ▪
Demonstraţie
$(a_i \in \mathbf{R})_{1 \leq i \leq 3}$ există şi sunt unici cu proprietatea din enunţ : $a_1 = \vec{v} \cdot \vec{\imath}$, $a_2 = \vec{v} \cdot \vec{\jmath}$, $a_3 = \vec{v} \cdot \vec{k}$. □

Definiţia 2.7 (Bază)
O *bază* a unui spaţiu vectorial (K, V) este un sistem $B \subseteq V$ de vectori astfel încât orice vector din V se exprimă în mod unic ca o combinaţie liniară de elemente din B. ▪

Observaţia 2.2
Sistemul $\{\vec{\imath}, \vec{\jmath}, \vec{k}\}$ este o bază a spaţiului vectorial $(\mathbf{R}, V_3)$. ▪

Definiţia 2.8 (Coordonate)

Fie $B = \{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n\}$ o bază în spaţiul vectorial (K, V). Atunci $\forall \vec{v} \in V$, $\exists (a_i \in K)_{1 \leq i \leq n}$ unice astfel încât $\vec{v} = \sum_{i=1}^{n} a_i \vec{v}_i$. Numim $(a_i)_{1 \leq i \leq n}$ *coordonatele* lui $\vec{v}$ în raport cu baza B. ▪

Definiţia 2.9 (Baze ortogonale, ortonormale)

1. O bază $B = \{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n\}$ în spaţiul vectorial (K, V) este *ortogonală* dacă vectorii bazei sunt ortogonali doi câte doi : $(\forall 1 \leq i < j \leq n)(\vec{v}_i \cdot \vec{v}_j = 0)$.
2. Baza B este *ortonormală* dacă este ortogonală şi vectorii bazei sunt vectori unitate : $(\forall 1 \leq i \leq n)(\mid \vec{v}_i \mid = 1)$.

▪

Observaţia 2.3

Sistemul $\{\vec{i}, \vec{j}, \vec{k}\}$ este o bază ortonormală a spaţiului vectorial $(\mathbf{R}, V_3)$. ▪

Capitolul 3

Aplicaţii liniare

În cele ce urmează reamintim câteva noţiuni legate de aplicaţiile liniare.

Definiţia 3.1
Se numeşte *aplicaţie liniară* o aplicaţie $A : V \to V'$ astfel încât $(\forall a \in \mathbf{R})(\forall \vec{u}, \vec{v} \in V)(A(\vec{u}+\vec{v}) = A(\vec{u}) + A(\vec{v}) \wedge A(a\vec{u}) = aA(\vec{u}))$. ▪

Exemplul 3.1
Aplicaţia identică $I : V \to V, I(\vec{v}) = \vec{v}, \forall \vec{v} \in V$ este o aplicaţie liniară : $I(\vec{u}+\vec{v}) = \vec{u}+\vec{v} = I(\vec{u}) + I(\vec{v})$ şi $I(a\vec{u}) = a\vec{u} = aI(\vec{u})$. ▪

Definiţia 3.2 (Operaţii definite pentru aplicaţii liniare)
Fie V, V' două spaţii vectoriale[1].
Fie $AL(V,V') = \{A \mid A : V \to V'$ şi A este aplicaţie liniară$\}$. Atunci $AL(V,V')$ este un spaţiu vectorial :

1. Fie $A, B \in AL(V,V')$, $(A+B)(\vec{v}) = A(\vec{v}) + B(\vec{v})$, $(aA)(\vec{v}) = aA(\vec{v})$. Se poate arăta că $A+B$ şi aA sunt aplicaţii liniare $(\in AL(V,V'))$.

2. Produsul a două aplicaţii liniare se defineşte astfel : fie $A \in AL(V,V')$ şi $B \in AL(V',V'')$, unde V, V', V'' sunt spaţii vectoriale. Definim $B \cdot A \in AL(V,V'')$ prin $(B \cdot A)(\vec{v}) = B(A(\vec{v}))$.

▪

Definiţia 3.3
O aplicaţie liniară $A \in AL(V,V)$ este *ortogonală* dacă $(\forall \vec{u}, \vec{v} \in V)(\vec{u} \cdot \vec{v} = A(\vec{u}) \cdot A(\vec{v}))$ i.e., lasă neschimbat produsul scalar al vectorilor. ▪

Proprietatea 3.1
Fie $A \in AL(V,V)$ o aplicaţie liniară ortogonală. Atunci avem :

[1] dacă mulţimea scalarilor este subînţeleasă atunci o putem omite

1. $(\forall \vec{u} \in V)(\mid \vec{u} \mid = \mid A(\vec{u}) \mid)$ i.e., lasă neschimbată lungimea vectorilor. Această proprietate rezultă din faptul că $1 = \cos \widehat{\vec{u}, \vec{u}} = \frac{\vec{u} \cdot \vec{u}}{|\vec{u}||\vec{u}|} \iff \vec{u} \cdot \vec{u} = \mid \vec{u} \mid\mid \vec{u} \mid \iff \mid \vec{u} \mid = \sqrt{\vec{u} \cdot \vec{u}}$ şi deci $\mid A(\vec{u}) \mid = \sqrt{A(\vec{u}) \cdot A(\vec{u})} = \sqrt{\vec{u} \cdot \vec{u}} = \mid \vec{u} \mid$.

2. $(\forall \vec{u}, \vec{v} \in V)(\widehat{\vec{u}, \vec{v}} = \widehat{A(\vec{u}), A(\vec{v})})$. Se arată ca mai sus.

▪

Exemplul 3.2

Se poate arăta că rotaţiile sunt aplicaţii liniare ortogonale. ▪

Definiţia 3.4

Fie $A \in AL(V, V)$ o aplicaţie liniară. Dacă A este şi izomorfism atunci A se numeşte *aplicaţie liniară regulată* :

ALR_1 $(\forall \vec{u}, \vec{v} \in V)(A(\vec{u} + \vec{v}) = A(\vec{u}) + A(\vec{v}))$,

ALR_2 $(\forall a \in K)(\forall \vec{v} \in V)(A(a\vec{v}) = aA(\vec{v}))$ şi

ALR_3 $A : V \to V$ este bijectivă.

▪

Proprietatea 3.2

Fie $B \subseteq V$ o bază a spaţiului vectorial V. Fie $A \in AL(V, V)$ o aplicaţie liniară regulată. Atunci $A(B) \subseteq V$ este tot o bază în V. ▪

Demonstraţie

Fie $B = \{\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n\}$. Fie $\vec{v} \in V$. Atunci $(\exists v_i \in K)_{1 \leq i \leq n}$ unice astfel încât $\vec{v} = \sum_{i=1}^{n} v_i \vec{e}_i$. Deci $A(\vec{v}) = A(\sum_{i=1}^{n} v_i \vec{e}_i) = \sum_{i=1}^{n} A(v_i \vec{e}_i) = \sum_{i=1}^{n} v_i A(\vec{e}_i)$. Dar $A(V) = V$ (avem $A(V) \subseteq V$ şi $V \subseteq A(V)$ deoarece A este surjectivă). Deci $\forall \vec{w} \in A(V)(= V)$, $(\exists w_i \in K)_{1 \leq i \leq n}$ unice $(\vec{w}_i = \vec{v}_i, \forall 1 \leq i \leq n)$ astfel încât $\vec{w} = \sum_{i=1}^{n} w_i \vec{A}(e_i)$. Avem $\mid A(B) \mid = \mid B \mid$ deoarece A este bijectivă. Deci $A(B)$ este o bază în $A(V)$ şi deci în V.

□

Proprietatea 3.3

Fie $A \in AL(V, V)$ o aplicaţie liniară ortogonală. Fie $B \subseteq V$ o bază ortonormală. Atunci $A(B)$ este tot o bază ortonormală. ▪

Demonstraţie

Fie $\vec{v}_1, \vec{v}_2 \in B$. Atunci avem $\vec{v}_1 \cdot \vec{v}_2 = 0$ şi $\mid \vec{v}_1 \mid = \mid \vec{v}_2 \mid = 1$. Deoarece A este ortogonală avem $A(\vec{v}_1) \cdot A(\vec{v}_2) = \vec{v}_1 \cdot \vec{v}_2 = 0$. Deasemeni mai avem : $1 = \mid \vec{v}_1 \mid = \mid A(\vec{v}_1) \mid$ şi $1 = \mid \vec{v}_2 \mid =$

$| A(\vec{v}_2) |$. Deci avem : $(\forall \vec{w}_1, \vec{w}_2 \in A(B))(| \vec{w}_1 | = | \vec{w}_2 | = 1 \wedge \vec{w}_1 \cdot \vec{w}_2 = 0)$. Evident avem şi $| A(B) | = | B |$ deoarece vectorii din $A(B)$ sunt distincţi (sunt ortogonali 2 câte 2).

Deci $A(B)$ este o bază ortonormală.

□

Definiţia 3.5

Fie matricea $A \in \mathcal{M}_{m \times n}(\mathbf{R})$. Numim A o matrice *ortogonală* dacă ${}^tA = A^{-1}$ i.e., $A \cdot {}^tA = I_m$. ▪

Proprietatea 3.4

Fie matricea ortogonală $A \in \mathcal{M}_{m \times n}(\mathbf{R})$. Dacă interpretăm liniile (coloanele) matricii A ca vectori din $\mathbf{R}^n$ (respectiv $\mathbf{R}^m$) atunci avem : $(\forall \vec{u}, \vec{v}$ linii(coloane) din $A)(\vec{u} \neq \vec{v} \Rightarrow \vec{u} \cdot \vec{v} = 0 \wedge \vec{u} = \vec{v} \Rightarrow \vec{u} \cdot \vec{v} = 1)$ i.e., liniile (coloanele) matricii A sunt o bază ortonormală în $\mathbf{R}^n$ (respectiv $\mathbf{R}^m$). ▪

Definiţia 3.6

Fie $A \in \mathcal{M}_{m \times n}(\mathbf{R})$ o matrice ortogonală. Dacă $\det A = 1$ atunci A se numeşte *matrice ortogonală proprie.* ▪

Exemplul 3.3

Se poate arăta că matricile rotaţiilor 3D sunt matrici ortogonale proprii. ▪

Capitolul 4

Lumină (a)cromatică

Culoarea unui obiect depinde nu numai de obiectul în sine ci şi de sursa de lumină care îl luminează, de culoarea zonelor adiacente şi de sistemul vizual uman.

Unele obiecte reflectă lumina (de ex. un perete, o foaie de hârtie) iar altele o transmit (de ex. o folie de celofan, sticla).

4.1 Lumină acromatică

Senzaţiile vizuale acromatice pot fi descrise utilizând doar culorile negru, gri şi alb.

Definiţia 4.1

Lumină acromatică este ceea ce se poate observa pe un ecran TV sau monitor A/N[1]. ▪

Lumină acromatică are un singur atribut : *cantitatea.*

Definiţia 4.2

Cantitatea de lumină poate fi descrisă prin conceptul fizic de *energie* (caz în care utilizăm termenii *intensitate* şi *luminanţă*) sau prin conceptul psihologic de *intensitate percepută* (caz în care utilizăm termenul de *strălucire*[2]). ▪

Diverselor niveluri de intensitate li se asociază scalari : 0 (reprezentând culoarea neagră) şi 1 (reprezentând culoarea albă). Nivelurile între 0 şi 1 reprezintă nuanţe de gri.

4.1.1 Selectarea intensităţilor

În lucrul cu intensităţile luminii acromatice se pun 2 probleme :

[1] alb/negru

[2] *brightness* în engleză

Problema 4.1

Dacă dorim să avem $n+1$ niveluri de intensitate ($n \in \mathbf{N}$), cum le plasăm în intervalul $[0,1]$? ▪

Problema 4.2

În mod practic, de câte niveluri de intensitate este nevoie ? Câte niveluri de intensitate sunt suficiente ? ▪

În cazul problemei 4.1, pag. 38 să presupunem că dorim să avem 256 ($n = 255$) niveluri de intensitate :

- Dacă am plasa 128 niveluri în $[0, 0.1]$ şi alte 128 niveluri în $[0.9, 1]$ atunci trecerea de la intensitatea 0.1 la 0.9 ar apărea discontinuă (spre deosebire de celelalte tranziţii).
- Dacă le-am plasa uniform în intervalul $[0,1]$ (la distanţă de $\frac{1}{255}$) atunci această alegere nu ar fi conformă cu o anumită caracteristică a ochiului uman :

  Fapt 4.1 *Ochiul uman percepe nu atât valori absolute ale intensităţii cât rapoarte între intensităţi.*

  Astfel trecerea de la intensităţile 0.1 şi 0.5 la 0.11 şi 0.55 este resimţită ca identică. În concluzie vom amplasa nivelurile de intensitate la distanţe logaritmice (şi nu liniare) pentru a obţine diferenţe egale în strălucire, aşa cum rezultă din figura 4.1, pag. 39.

Pentru obţinerea celor $n+1$ niveluri de intensitate procedăm astfel : fie I_0 nivelul de intensitate cel mai de jos posibil (din motive practice, pentru un CRT avem $I_0 \in [0.005, 0.025]$).

Alegem $I_1 = r \cdot I_0$, $I_2 = r \cdot I_1 = r^2 \cdot I_0$, ..., $I_j = r^j \cdot I_0$ $(0 \leq j \leq n)$, ..., $I_n = r^n \cdot I_0 = 1$. Deci $r = \left(\frac{1}{I_0}\right)^{\frac{1}{n}}$ şi $I_j = r^j \cdot I_0 = I_0^{1-\frac{j}{n}} = I_0^{\frac{n-j}{n}}$, $0 \leq j \leq n$.

Exemplul 4.1

Dacă $n = 3$, $I_0 = \frac{1}{8}$ atunci $r = 2$ şi avem nivelurile de intensitate : $\{\frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1\}$. ▪

Definiţia 4.3

Se numeşte *domeniu dinamic*[3] valoarea $\frac{1}{I_0}$ (raportul dintre intensitatea maximă şi minimă). ▪

Observaţia 4.1

Din punct de vedere practic valoarea unui pixel şi valoarea intensităţii sale calculată conform ecuaţiei $I_j = r^j \cdot I_0$, $0 \leq j \leq n$ diferă. Se procedează astfel :

1. Intensitatea luminii pe ecranul unui CRT depinde de energia unei unde de electroni ce stimulează nişte particule de fosfor. Practic avem $I = k_1 \cdot N^\gamma$, k_1, γ constante şi $\gamma \in [2.2, 2.5]$.

[3] *dynamic range* în engleză

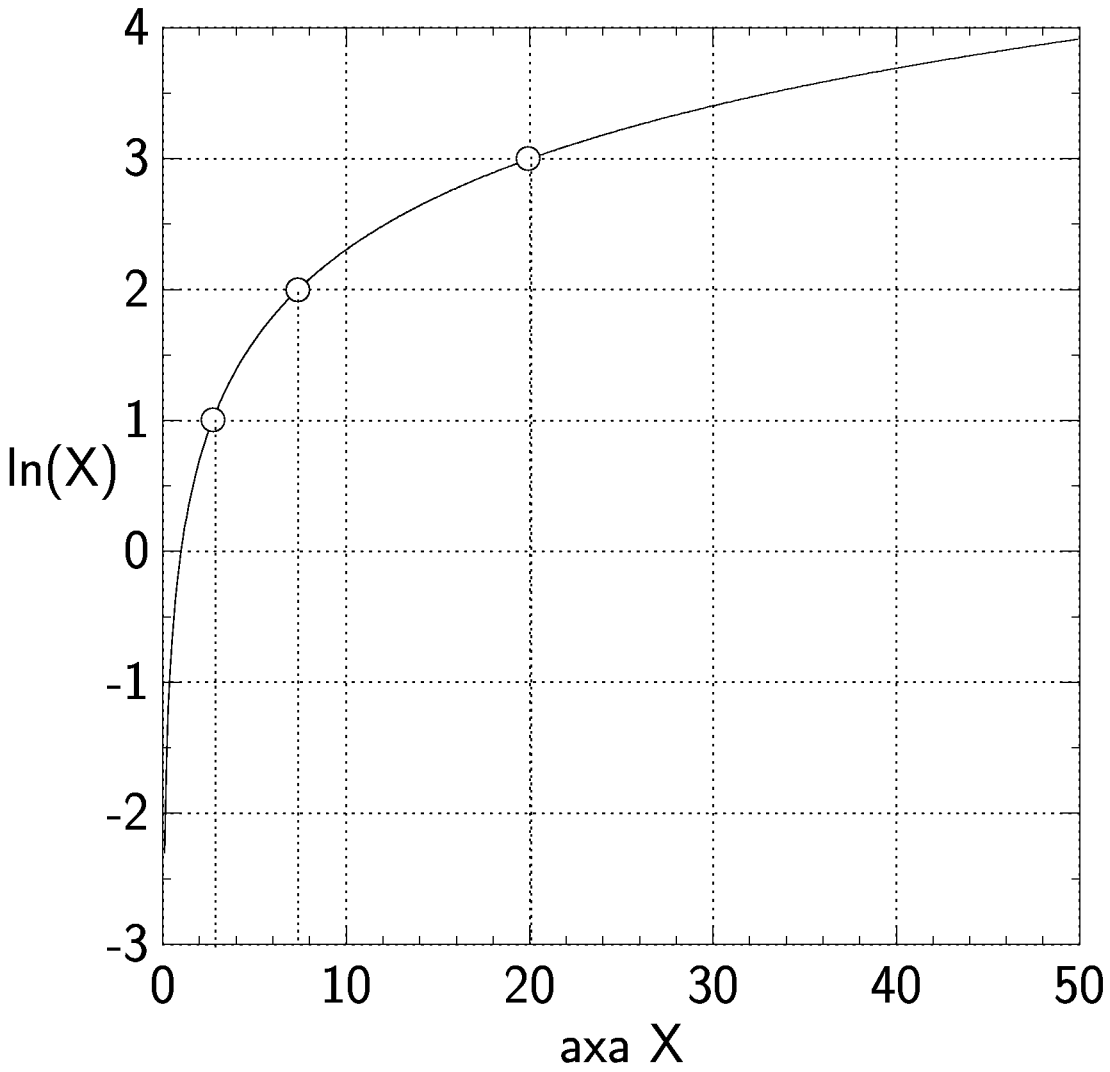


Figura 4.1: Amplasarea nivelurilor de intensitate

2. N este proporţională cu valoarea V a unui pixel şi deci avem $I = K \cdot V^{\gamma} \Leftrightarrow V = \left(\frac{I}{K}\right)^{\frac{1}{\gamma}}$.

3. Să presupunem că dorim să determinăm valoarea V a unui pixel astfel încât aceasta să creeze intensitatea I. Se procedează astfel :

   - Determinăm j astfel încât intensitatea I_j este cea mai apropiată de I : $I = I_j = r^j \cdot I_0 \Rightarrow j = \left[log_r\left(\frac{I}{I_0}\right)\right]$.
   - Deoarece $V = \left(\frac{I}{K}\right)^{\frac{1}{\gamma}}$ vom avea $V_j = \left[\left(\frac{I_j}{K}\right)^{\frac{1}{\gamma}}\right]$.

▪

În cazul problemei 4.2, pag. 38 prin termenul de *suficient* ne gândim la numărul de intensităţi necesar pentru ca reproducerea unei imagini A/N (de ex. o fotografie A/N) să aibă un aspect *continuu* (i.e., între 2 pixeli vecini să nu existe diferenţe mari de intensitate). Aspectul continuu este posibil doar pentru $r = 1.01$. S-a putut observa că dacă $r < 1.01$ ochiul nu poate distinge între intensităţile I_j şi I_{j+1}. Deci, din ecuaţia $r = \left(\frac{1}{I_0}\right)^{\frac{1}{n}}$ obţinem $r^n = \frac{1}{I_0} \Rightarrow n = log_r\left(\frac{1}{I_0}\right)$ şi deci $n = log_{1.01}\left(\frac{1}{I_0}\right)$.

Câteva din valorile $\frac{1}{I_0}$ şi n se regăsesc în tabelul 4.1, pag. 40.

	$\frac{1}{I_0}$	n
CRT	$50 - 200$	$400 - 530$
hârtie fotografică	100	465
hârtie A/N	100	465
hârtie culori	50	400
ziar A/N	10	234

Tabelul 4.1: Valori pentru $\frac{1}{I_0}$ şi n

4.1.2 Aproximarea intensităţilor prin autotipie

Problema 4.3

În cazul ecranelor (sau al dispozitivelor de tipărire) nu putem avea decât un număr limitat de intensităţi (2 pentru ecrane monocrome sau 4, 8 dacă avem ecrane rastru cu 2, 3 biţi/pixel). Cum putem obţine, în aceste cazuri, niveluri de intensitate suplimentare ? ▪

Pentru rezolvarea problemei 4.3, pag. 40 ne folosim de faptul că *ochiul realizează o integrare spaţială* : dacă privim o zonă foarte mică de la o distanţă suficient de mare atunci ochiul percepe o intensitate medie (şi nu intensitatea fiecărui pixel în parte).

Tehnica de *autotipie*[4] constă în simularea intensităţii prin cercuri negre de arie direct proporţională cu $1-I$, unde I reprezintă valoarea intensităţii din fotografia originală. Aceasta tehnică se foloseşte, în general, pentru tipărirea fotografiilor A/N.

Dispozitivele de ieşire grafice pot aproxima cercurile de arie variabilă folosite în autotipie astfel : o zonă de 2×2 pixeli monocromi poate simula 5 niveluri de intensităţi diferite, cu preţul reducerii la jumătate a rezoluţiei spaţiale[5] pe fiecare axă (vezi figura 4.2, pag. 41).

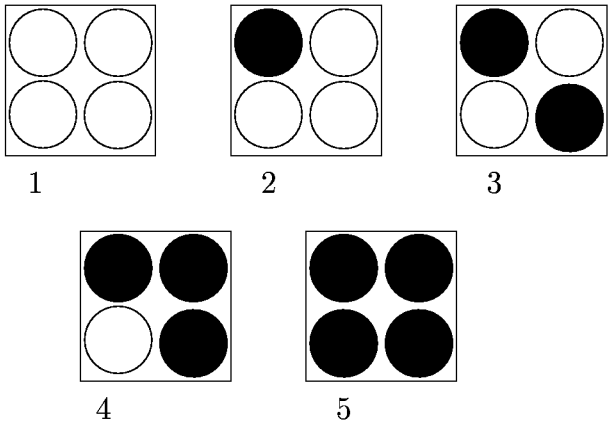


Figura 4.2: Simulare intensităţi

În general avem următoarea proprietate :

Proprietatea 4.1

Un grup de $n \times n$ pixeli monocromi poate furniza $n^2 + 1$ niveluri de intensitate. ▪

Exemplul 4.2

O zonă de 3×3 pixeli monocromi poate simula 10 intensităţi : vezi figura 4.3, pag. 42. Putem sintetiza aceste niveluri de intensitate într-o matrice de oscilaţie[6] : $\begin{pmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{pmatrix}$, unde nivelul de intensitate I se obţine setând toţi pixelii cu valori mai mici decât I. ▪

Putem avea mai multe modele pentru zonele de $n \times n$ pixeli utilizate în autotipie, dar aceste modele trebuie să satisfacă următoarele reguli :

- Un anumit model nu trebuie să introducă noi obiecte vizuale în zone de intensităţi egale. De exemplu, dacă în figura 4.3, pag. 42 în loc de modelul dat pentru 3 am fi utilizat modelul $3'$ am produce apariţia unei linii orizontale în orice zonă a imaginii iniţiale colorată uniform cu pixeli de intensitate 3.

[4] *halftoning* în engleză

[5] i.e., dacă o imagine era reprezentată pe $n \times n$ pixeli, fiecare pixel putând avea 5 intensităţi, în cazul când avem un ecran monocrom, pentru a avea o imagine la fel de detaliată avem nevoie de $2n \times 2n$ pixeli

[6] *Dither matrix* în engleză

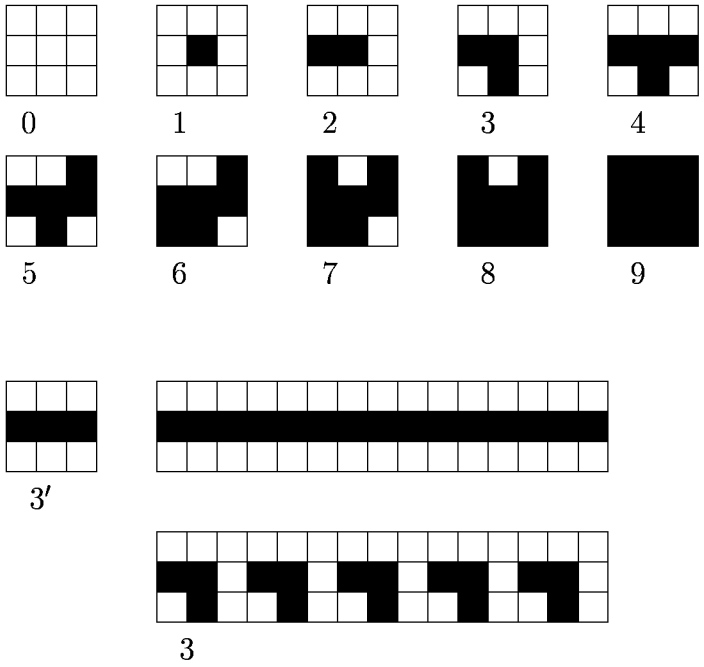


Figura 4.3: 10 niveluri de intensitate

- Modelele trebuie să alcătuiască o secvenţă crescătoare astfel încât dacă un pixel era intensificat pentru simularea nivelului de intensitate j atunci va rămâne intensificat şi pentru toate nivelurile de intensitate $k > j$.
- Modelele trebuie să *crească* dinspre centru înspre exterior pentru a simula efectul de creştere a mărimii punctului.
- Pixelii setaţi (1) trebuie să fie grupaţi. Nu se admit pixeli 1 răsfiraţi.

Aproximarea diferitelor niveluri de intensitate nu se limitează doar la ecrane monocrome. În cazul unui ecran cu 2 biţi pe pixel (i.e., 4 niveluri de intensitate pentru un pixel), dacă utilizăm zone de 2×2 pixeli, se pot simula 13 niveluri de intensitate (bineînţeles cu preţul reducerii rezoluţiei spaţiale) : vezi figura 4.4, pag. 43.

4.2 Lumină cromatică

Percepţia culorii implică 3 valori : *nuanţă*[7], *saturaţie*[8], *luminozitate*[9]. Nuanţa distinge familiile de culori : avem nuanţe de roşu, verde, albastru, etc. Saturaţia se referă la distanţa culorii

[7] *Hue* în engleză
[8] *Saturation* în engleză
[9] *Lightness* în engleză

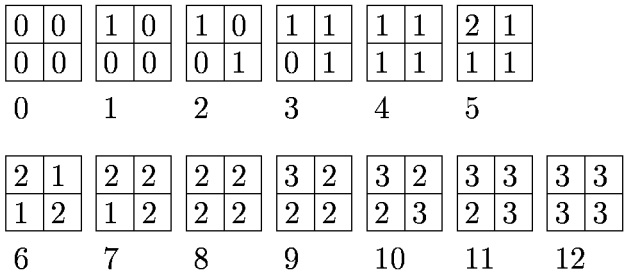


Figura 4.4: 13 niveluri de intensitate

respective faţă de culoarea gri care are aceeaşi intensitate (de ex., culoarea roşie este puternic saturată iar cea roz este relativ nesaturată, adică mai apropiată de culoarea gri de aceeaşi intensitate). Luminozitatea reprezintă intensitatea percepută a unui obiect care reflectă lumina. Strălucirea[10] se referă la intensitatea percepută a unui obiect care emite lumină (de ex., un bec, soarele, CRT).

Există mai multe modalităţi de specificare şi de măsurare a culorilor :

- prin compararea culorii necunoscute cu un set de culori standard (modelul de culori Munsell[11]),
- în modelul artistic culoarea este specificată prin 3 parametri : nuanţă[12], umbrire[13], ton[14], vezi figura 4.5, pag. 44. O *nuanţă* rezultă prin adăugarea culorii alb unei culori pure, astfel scăzând saturaţia. O *umbrire* provine din adăugarea de culoare neagră unei culori pure, astfel scăzând luminozitatea. Un *ton* este obţinut adăugând atât culoarea alb cât şi negru unei culori pure. Obţinem astfel culori de aceeaşi nuanţă (hue) dar cu saturaţie şi luminozitate variabile. Amestecarea culorilor alb/negru produce culori gri.

4.2.1 Modele de culori pentru grafica rastru

Definiţia 4.4

Un *model de culori* constă într-un sistem cartezian de coordonate 3D şi o submulţime a $\mathbf{R}^3$ care conţine toate culorile dintr-o anumită gamă de culori. ▪

De exemplu, modelul RGB constă din cubul $[0,1]^3$ definit într-un sistem de coordonate cartezian 3D.

[10] *Brightness* în engleză
[11] *Munsell space* în engleză
[12] *Tint* în engleză
[13] *Shade* în engleză
[14] *Tone* în engleză

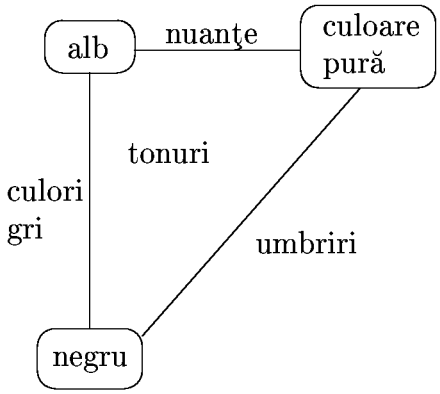


Figura 4.5: Modelul artistic

Modelul RGB

Modelul de culori RGB[15] este utilizat pentru CRTs în culori precum şi în grafica rastru. Spaţiul de culori este cubul unitate ($[0,1] \times [0,1] \times [0,1]$) (vezi figura 4.6, pag. 45). O anumită culoare este reprezentată printr-un triplet de culori primare : (roşu, verde, albastru). Aceste culori primare sunt şi *aditive* : fiecărei culori $C(r,g,b)$ îi corespunde bijectiv un punct din spaţiul de culori RGB (punct care este dat de vectorul $r \cdot \vec{R} + g \cdot \vec{G} + b \cdot \vec{B}$, unde $\vec{R}, \vec{G}, \vec{B}$ sunt versori). Pe diagonala principală regăsim diverse nuanţe de gri.

Modelul HSV

Spre deosebire de alte modele de culori orientate spre hardware (de ex. RGB), modelul HSV[16] este orientat spre utilizator, fiind apropiat de modelul artistic. Spaţiul de culori HSV este o piramidă hexagonală într-un sistem de coordonate polar (vezi figura 4.7, pag. 46). Culorile gri sunt situate pe dreapta *black-white*.

O culoare este specificată în modelul HSV prin tripletul (h,s,v), unde $h \in [0°, 360°)$, $s \in [0,1]$ şi $v \in [0,1]$. Parametrul h (hue) semnifică nuanţa culorii, s (saturation) specifică saturaţia culorii iar v (value) specifică strălucirea percepută a culorii. În modelul HSV, variind h alegem o culoare (dintr-o familie de culori), variind s modificăm saturaţia culorii (făcând-o să varieze între culoarea gri cu aceeaşi intensitate şi culoarea pură) iar variind v variem contribuţia culorii negre (variind astfel luminozitatea).

Corespondenţa dintre modelele RGB şi HSV

Conversia culorilor din modelul RGB în HSV se efectuează proiectând cubul unitate RGB pe un plan perpendicular pe diagonala principală în punctul *white*.

[15] *Red, Green, Blue* în engleză
[16] *Hue, Saturation, Value* în engleză

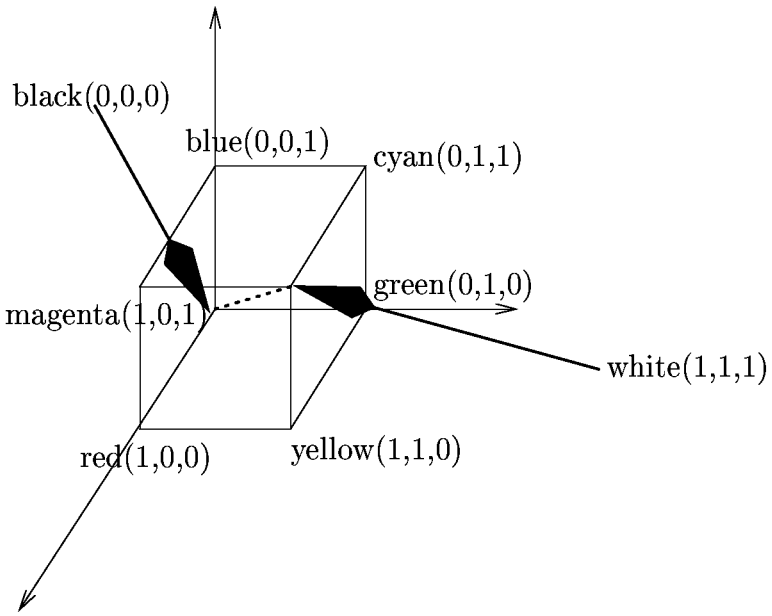


Figura 4.6: Modelul RGB

Algoritmul 1 (Conversia RGB în HSV)

Date de intrare : r, g, b : **real**

Date de ieşire : h, s, v : **real**

Metodă :

```
procedure RGBtoHSV(var h,s,v : real, r,g,b : real) {
  const int UNDEF = -1;
  real max, min, delta;

  max = valoarea_maxima(r,g,b);
  min = valoarea_minima(r,g,b);
  v = max;
  s = (max != 0.0) ? ((max - min)/max) : 0;
  if (s == 0) h = UNDEF;
  else {
    delta = max - min;
    if (delta == 0.0) { h = UNDEF; s = 0; v = 1; }
    if (max == 0) { h = s = UNDEF; v = 0; }
    if (r == max) h = (g - b) / delta;
    else if (g == max) h = (b - r) / delta + 2.0;
    else if (b == max) h = (r - g) / delta + 4.0;
    else {}
```

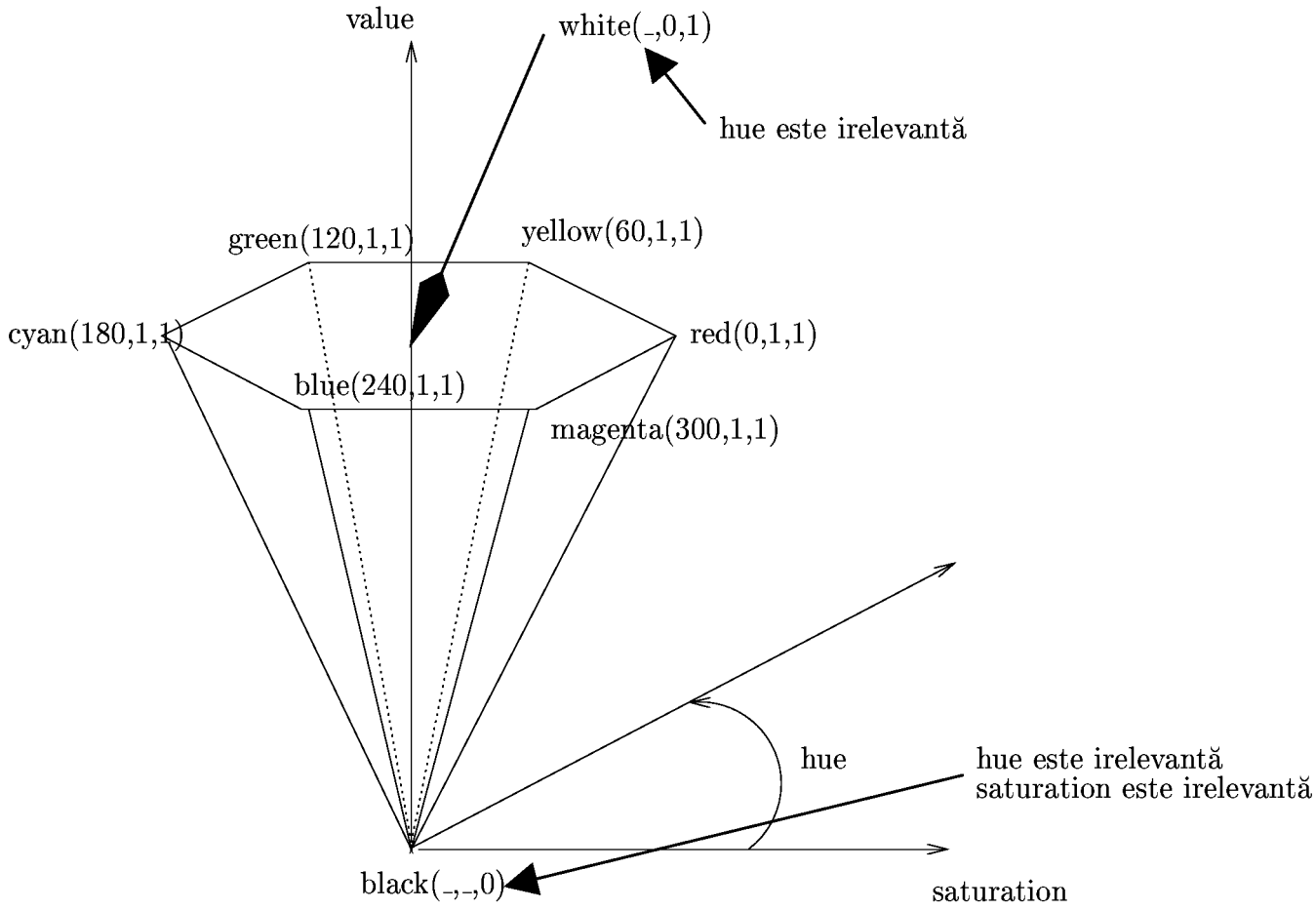


Figura 4.7: Modelul HSV

```
    h = h * 60;
    if (h < 0) h = h + 360;
  }
}
```

■

Algoritmul 2 (Conversia HSV în RGB)

Date de intrare : h, s, v : **real**

Date de ieşire : r, g, b : **real**

Metodă :

```
procedure HSVtoRGB(var r,g,b : real, h,s,v : real) {
  const int UNDEF = -1;
  real f,p,q,t;
  int i;

  if (s == 0) {
    if (h == UNDEF) r = g = b = v;
    else printf("Eroare");
  }
  else {
    if (h == 360) h = 0;
    h = h / 60;
    i = floor(h);
    f = h - i;
    p = v * (1 - s);
    q = v * (1 - s * f);
    t = v * (1 - (s * (1 - f)));
    switch (i) {
      case 0 : r = v; g = t; b = p; break;
      case 1 : r = q; g = v; b = p; break;
      case 2 : r = p; g = v; b = t; break;
      case 3 : r = p; g = q; b = v; break;
      case 4 : r = t; g = p; b = v; break;
      case 5 : r = v; g = p; b = q; break;
    }
  }
}
```

■

4.2.2 Specificarea interactivă a culorii

În aplicaţiile care utilizează culori pentru drepte, text, suprafeţe, etc. există mai multe modalităţi (vezi figura 4.8, pag. 49) prin care se permite utilizatorului selectarea unei culori :

- Utilizatorul poate alege o culoare din mai multe mostre de culori (modelul indexat),
- Utilizatorul poate specifica coordonate R, G, B,
- Utilizatorul poate specifica coordonate H, S, V.

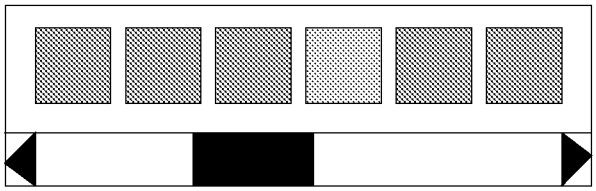
4.2.3 Utilizarea culorii în grafica pe calculator

Un *principiu fundamental* al utilizării culorii în diverse aplicaţii este următorul : utilizarea culorii trebuie să aibă un scop funcţional şi să nu aibă un subînţeles.

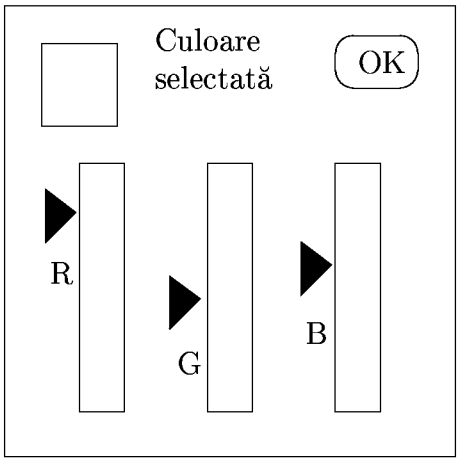
Reguli de estetică ale culorii

1. Culorile trebuie selectate conform unei anumite metode, de obicei prin traversarea unui drum continuu în modelul culorilor sau prin restricţionarea culorilor la anumite planuri sau secţiuni hexagonale.
2. Nu se recomandă alegerea aleatoare a H, S.
3. Dacă pentru un grafic se utilizează câteva culori atunci se recomandă pentru fundal[17] utilizarea unei culori care este complementară unei culori utilizate în grafic (culoarea complementara unei culori C este acea culoare care combinată cu C produce o culoare neutră : alb, gri sau negru. Pentru modelul RGB se obţine prin simetrie faţă de centrul cubului iar pentru modelul HSV se obţine prin simetrie faţă de centrul hexagonului). Dacă se utilizează mai multe culori atunci pentru fundal se recomandă culoarea gri.
4. Dacă alăturarea a 2 culori nu este armonioasă se recomandă utilizarea de chenare subţiri de culoare neagră.
5. Dacă dorim să asociem diverse înţelesuri culorilor atunci trebuie să luăm câteva precauţii :
   (a) Codurile culorilor pot avea înţelesuri implicite : de ex., *roşu* sugerează atenţie, eroare, *verde* corect.
   (b) Culorile strălucitoare denotă o importanţă crescută asociată unui element.
   (c) Două obiecte cu aceeaşi culoare pot fi interpretate ca având acelaşi cod (ceea ce nu e valabil întotdeauna).

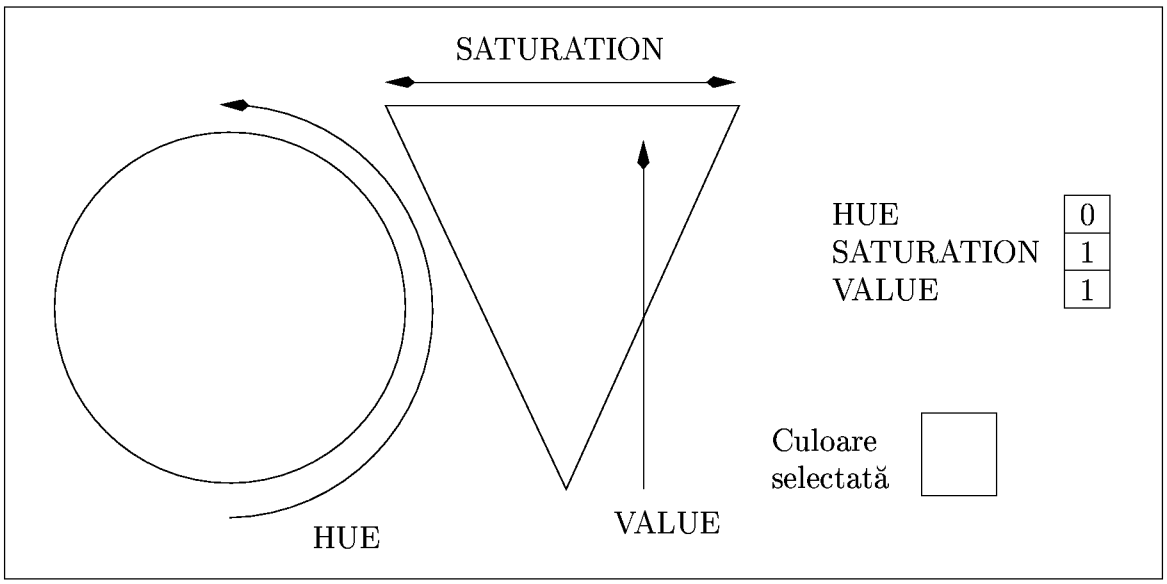
[17] *Background* în engleză



Modelul indexat



Modelul RGB



Modelul HSV

Figura 4.8: Specificarea interactivă a culorii

6. Din regula 5, pag. 48 rezultă că utilizarea culorilor în meniuri, ferestre de dialog, chenare de ferestre trebuie să fie restricţionată.

7. Detaliile fine (segmente de dreaptă, text) trebuie să difere de fundal nu doar în cromaticitate ci şi în strălucire (în mod deosebit când se utilizează culori conţinând albastru).

8. Combinaţii greşite de culori : albastru şi negru (deoarece diferă foarte puţin în strălucire), galben şi alb (idem). Combinaţii bune de culori : galben (pentru fundal) şi negru (pentru text), text alb pe fundal albastru. Alte combinaţii greşite sunt : roşu şi verde (când nu sunt strălucitoare pot crea probleme persoanelor care nu percep diferenţe între cele două culori).

9. Ochiul nu poate distinge culoarea obiectelor foarte mici, deci nu se recomandă, într-o imagine colorarea pixelilor vecini cu culori distincte.

10. Culoarea percepută de către ochi a unei suprafeţe este afectată de suprafaţa înconjurătoare. Se recomandă deci culori gri sau nesaturate pentru suprafeţele înconjurătoare.

11. Culoarea unei suprafeţe poate modifica mărimea sub care este percepută de către ochi (de ex., suprafeţele colorate roşu sunt percepute ca fiind mai mari decât cele colorate verde).

12. Nu se recomandă utilizarea de suprafeţe mari colorate cu o singură culoare.

13. Nu se recomandă colorarea a două suprafeţe cu două culori de la capetele opuse ale spectrului (ROGVAIV) : de ex., roşu pentru fundal şi albastru pentru prim-plan[18]. În acest caz roşu apare mai aproape iar albastru mai departe, invers faţă de ceea ce dorim.

[18] *Foreground* în engleză

Capitolul 5

Transformări geometrice

În acest capitol introducem transformările geometrice 2D/3D de bază : *translaţia*, *scalarea* şi *rotaţia*.

5.1 Transformări geometrice 2D

Definiţia 5.1 (Translaţie)
Se numeşte *translaţie* (notată $T(d_x, d_y)$) o transformare geometrică care asociază oricărui punct $P(x, y)$ un punct $P'(x', y')$ cu proprietatea că $x' = x + d_x$, $y' = y + d_y$.
Putem exprima matricial acest lucru :

$$P' = P + T(d_x, d_y), \tag{5.1}$$

unde $P' = \begin{pmatrix} x' \\ y' \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \end{pmatrix}$ şi $T(d_x, d_y) = \begin{pmatrix} d_x \\ d_y \end{pmatrix}$.

Translaţia unui obiect se face efectuând translaţii ale fiecărui punct al obiectului utilizând ecuaţia 5.1, pag. 51.

Translaţia unei drepte se face efectuând translaţiile a două puncte distincte aparţinând dreptei. ▪

Exerciţiul 5.1
Ce se obţine prin translaţia ($T(2, 0.5)$) poligonului cu vărfurile $A(1, 1)$, $B(2, 1)$, $C(2, 2)$, $D(1.5, 3)$, $E(1, 2)$ şi muchiile AB, BC, CD, DE, EA ? ▪

Definiţia 5.2 (Scalare)
Se numeşte *scalare* (notată $S(s_x, s_y)$) o transformare geometrică care asociază oricărui punct $P(x, y)$ un punct $P'(x', y')$ cu proprietatea că $x' = x \cdot s_x$, $y' = y \cdot s_y$.
Putem exprima matricial acest lucru :

$$P' = S(s_x, s_y) \cdot P, \tag{5.2}$$

unde $P' = \begin{pmatrix} x' \\ y' \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \end{pmatrix}$ şi $S(s_x, s_y) = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}$.

O scalare este *diferenţiată* dacă $s_x \neq s_y$ şi este *uniformă* dacă $s_x = s_y$. ▪

Observaţia 5.1

O scalare uniformă păstrează proporţiile obiectului. ▪

Exerciţiul 5.2

Ce se obţine prin scalarea $(S(\frac{1}{3}, \frac{1}{2}))$ poligonului cu vărfurile $A(3,2)$, $B(6,2)$, $C(6,4)$, $D(4,6)$, $E(3,4)$ şi muchiile AB, BC, CD, DE, EA ? ▪

Definiţia 5.3 (Rotaţie)

Se numeşte *rotaţie* (notată $R(\theta)$) o transformare geometrică care asociază oricărui punct $P(x,y)$ un punct $P'(x',y')$ cu proprietatea că $x' = x \cdot \cos\theta - y \cdot \sin\theta$, $y' = x \cdot \sin\theta + y \cdot \cos\theta$.

Putem exprima matricial acest lucru :

$$P' = R(\theta) \cdot P, \tag{5.3}$$

unde $P' = \begin{pmatrix} x' \\ y' \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \end{pmatrix}$ şi $R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$.

Unghiurile θ se măsoară în sens trigonometric. ▪

Exerciţiul 5.3

Să se arate că transformarea de rotaţie este bine definită: în figura 5.1, pag. 53 se aplică o transformare de rotaţie $R(\theta)$ punctului $P(x,y)$. Se obţine punctul $P'(x',y')$. Avem $OP = OP' = r$. Să se arate că P' are coordonatele date de ecuaţia 5.3, pag. 52. ▪

5.2 Coordonate omogene şi reprezentarea matricială a transformărilor 2D

Aşa cum se poate observa, ecuaţiile matriciale ale transformărilor geometrice translaţie (5.1, pag. 51), scalare (5.2, pag. 51) şi rotaţie (5.3, pag. 52) *nu* au aceeaşi formă (în cazul translaţiei este vorba de adunare de matrici iar în celelalte două cazuri de înmulţire de matrici). Acest lucru face dificilă exprimarea unor transformări geometrice compuse (de exemplu transformarea $R_1T_1R_2S_1T_2R_3$, unde prin litere majuscule $\{T, S, R\}$ s-au notat tipurile de transformări). Pentru a uşura exprimarea matematică a unor astfel de transformări compuse vom trata în mod uniform transformările geometrice, mai precis acestea vor fi exprimate exclusiv prin produs de matrici.

Pentru aceasta vom exprima transformările geometrice nu în coordonate carteziene 2D ci în *coordonate omogene.*

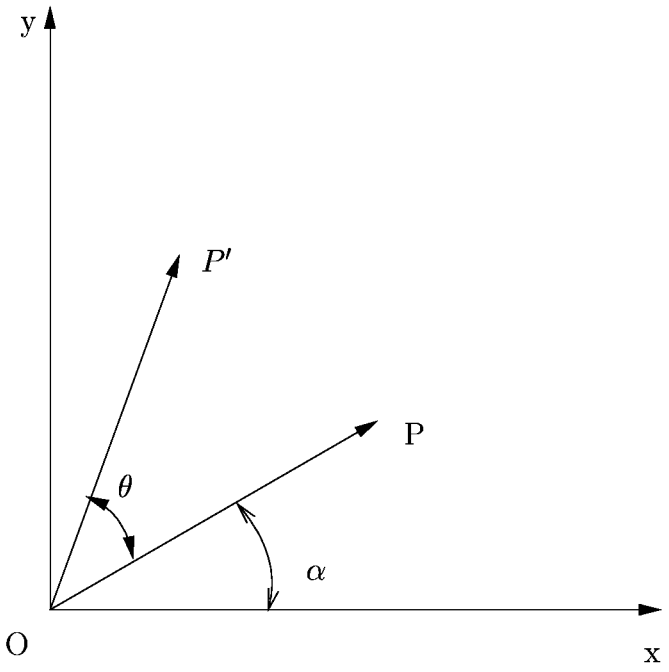


Figura 5.1: Transformarea de rotaţie este bine definită

Definiţia 5.4 (Coordonate omogene)

Spaţiul omogen 3D este spaţiul cartezian 3D fără origine (punctul $(0,0,0)$) şi deci tripletul (x,y,W) este un punct în *coordonate omogene 3D* dacă $x \neq 0 \vee y \neq 0 \vee W \neq 0$.

Unui punct cartezian 2D îi pot corespunde mai multe puncte omogene 3D şi anume : punctele omogene 3D (x,y,W) şi (x',y',W') reprezintă acelaşi punct cartezian 2D ddacă $(\exists \alpha \neq 0)(x' = \alpha x \wedge y' = \alpha y \wedge W' = \alpha W)$.

Punctele $(x,y,0)$ se numesc *puncte omogene la infinit.*

Numim transformare de *omogenizare* acea transformare care asociază unui punct omogen 3D (x,y,W), $W \neq 0$, punctul omogen 3D $(\frac{x}{W}, \frac{y}{W}, 1)$. ▪

Utilizând coordonatele omogene reformulăm definiţiile transformărilor geometrice de mai sus.

Definiţia 5.5 (Translaţie în coordonate omogene)

Se numeşte *translaţie* (notată $T(d_x,d_y)$) o transformare geometrică care asociază oricărui punct $P(x,y)$ un punct $P'(x',y')$ cu proprietatea că $x' = x + d_x$, $y' = y + d_y$.

Putem exprima matricial acest lucru :

$$P' = T(d_x,d_y) \cdot P, \tag{5.4}$$

unde $P' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ şi $T(d_x,d_y) = \begin{pmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{pmatrix}$. ▪

Proprietatea 5.1

Compunerea translaţiilor este o operaţie aditivă : daca P' se obţine din P prin translaţia $T(d_{x_1}, d_{y_1})$ iar P'' se obţine din P' prin translaţia $T(d_{x_2}, d_{y_2})$ atunci P'' se obţine din P prin translaţia $T(d_{x_1} + d_{x_2}, d_{y_1} + d_{y_2})$. ▪

Demonstraţie

Ca exerciţiu. □

Definiţia 5.6 (Scalare în coordonate omogene)

Se numeşte *scalare* (notată $S(s_x, s_y)$) o transformare geometrică care asociază oricărui punct $P(x, y)$ un punct $P'(x', y')$ cu proprietatea că $x' = x \cdot s_x$, $y' = y \cdot s_y$.

Putem exprima matricial acest lucru :

$$P' = S(s_x, s_y) \cdot P, \tag{5.5}$$

unde $P' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ şi $S(s_x, s_y) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$. ▪

Proprietatea 5.2

Compunerea scalărilor este o operaţie multiplicativă : daca P' se obţine din P prin scalarea $S(s_{x_1}, s_{y_1})$ iar P'' se obţine din P' prin scalarea $S(s_{x_2}, s_{y_2})$ atunci P'' se obţine din P prin scalarea $S(s_{x_1} \cdot s_{x_2}, s_{y_1} \cdot s_{y_2})$. ▪

Demonstraţie

Ca exerciţiu. □

Definiţia 5.7 (Rotaţie în coordonate omogene)

Se numeşte *rotaţie* (notată $R(\theta)$) o transformare geometrică care asociază oricărui punct $P(x, y)$ un punct $P'(x', y')$ cu proprietatea că $x' = x \cdot \cos\theta - y \cdot \sin\theta$, $y' = x \cdot \sin\theta + y \cdot \cos\theta$.

Putem exprima matricial acest lucru :

$$P' = R(\theta) \cdot P, \tag{5.6}$$

unde $P' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ şi $R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$. ▪

Proprietatea 5.3

Compunerea rotaţiilor este o operaţie aditivă : daca P' se obţine din P prin rotaţia $R(\theta_1)$ iar P'' se obţine din P' prin rotaţia $R(\theta_2)$ atunci P'' se obţine din P prin rotaţia $R(\theta_1 + \theta_2)$. ▪

Demonstraţie

Ca exerciţiu. □

Observaţia 5.2
Matricile transformărilor de rotaţie 2D/3D sunt matrici ortogonale proprii. ▪

Proprietatea 5.4

1. O matrice de transformare de forma $\begin{pmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$, unde matricea $\begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix}$ este ortogonală, păstrează unghiurile şi distanţele. Transformările asociate se numesc *transformări de corp rigid*[1].

2. O secvenţă arbitrară de matrici de rotaţie şi de translaţie creează o matrice de forma de mai sus.

3. Produsul unei secvenţe arbitrare de matrici de rotaţie, translaţie şi scalare se numeşte *transformare afină* şi păstrează doar paralelismul dreptelor (şi nu unghiurile şi distanţele).

▪

Exemplul 5.1
În figura 5.2, pag. 55 este prezentaţa o transformare afină, printr-o rotaţie de $-45°$, urmată de scalare neuniformă. ▪

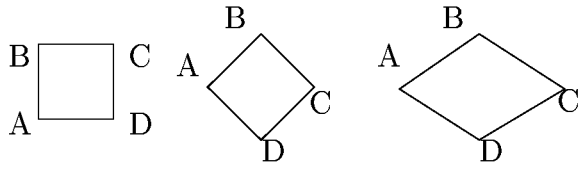


Figura 5.2: Transformare afină

Proprietatea 5.5
$R(\theta)$, $S(s_x, s_y)$, $T(d_x, d_y)$ sunt transformări afine. ▪

Definiţia 5.8 (Transformare forfecată)
Se numeşte *transformare forfecată* în lungul axei Ox (notată $SH_x(a)$) o transformare geometrică care asociază oricărui punct $P(x, y)$ un punct $P'(x', y')$ cu proprietatea că $x' = x + a \cdot y$, $y' = y$.

[1] *rigid-body transformation* în engleză

Putem exprima matricial acest lucru :

$$P' = SH_x(a) \cdot P, \tag{5.7}$$

unde $P' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ şi $SH_x(a) = \begin{pmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

O transformare forfecată în lungul axei Oy (notată $SH_y(b)$) este caracterizată matricial cu ajutorul matricii $SH_y(b) = \begin{pmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. ▪

Exemplul 5.2

În figura 5.3, pag. 56 sunt prezentate transformări forfecate în direcţia axelor Ox şi Oy aplicate unui pătrat de latură 1. ▪

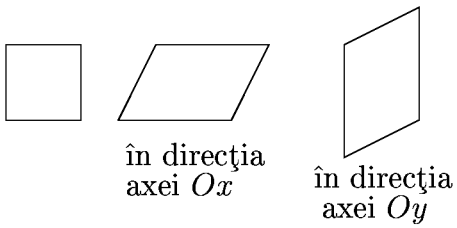


Figura 5.3: Transformări forfecate

5.3 Compunerea transformărilor 2D

Vom utiliza compunerea transformărilor 2D (înmulţirea matricilor aferente) pentru a combina transformările de translaţie, rotaţie, scalare într-o transformare generică pentru care dorim să-i calculăm matricea corespunzătoare.

Exemplul 5.3

În figura 5.4, pag. 57 este prezentată următoarea succesiune de transformări geometrice (care de fapt realizează rotaţia pentagonului în jurul punctului P_1) :

1. Translaţie obiect astfel încât P_1 să ajungă în originea sistemului de coordonate,
2. Rotaţie obiect,
3. Translaţie inversă astfel încât P_1 să ajungă în poziţia iniţială.

Pentru a obţine matricea transformării generale compunem transformările 2D componente : $T(x_1, y_1) \cdot R(\theta) \cdot T(-x1, -y1)$ şi obţinem matricea $\begin{pmatrix} \cos\theta & -\sin\theta & x_1(1-\cos\theta)+y_1\sin\theta \\ \sin\theta & \cos\theta & y_1(1-\cos\theta)+x_1\sin\theta \\ 0 & 0 & 1 \end{pmatrix}$.
Se verifică că este o transformare de corp rigid. ▪

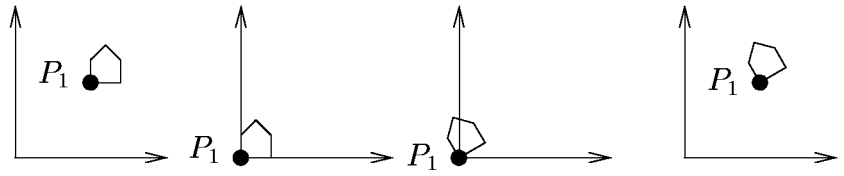


Figura 5.4: Rotaţia pentagonului în jurul P_1

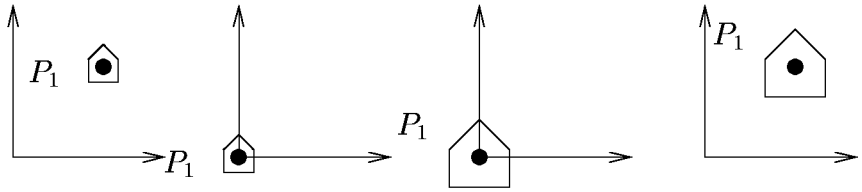


Figura 5.5: Scalarea pentagonului în raport cu P_1

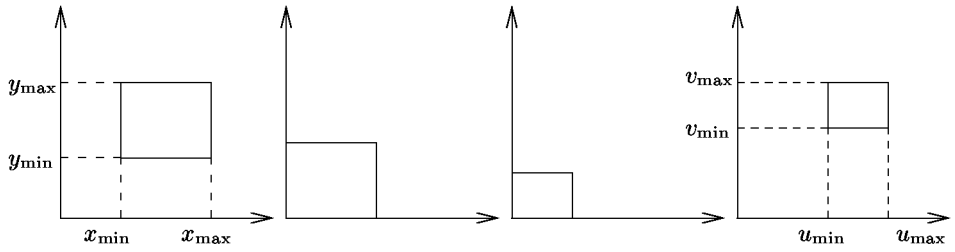


Figura 5.6: Transformarea window-viewport

Exemplul 5.4

În figura 5.5, pag. 57 este prezentată următoarea succesiune de transformări geometrice (care de fapt realizează scalarea pentagonului în raport cu punctul P_1) :

1. Translaţie obiect astfel încât P_1 să ajungă în originea sistemului de coordonate,
2. Scalare obiect,

3. Translaţie inversă astfel încât P_1 să ajungă în poziţia iniţială.

Pentru a obţine matricea transformării generale compunem transformările 2D componente : $T(x_1, y_1) \cdot S(s_x, s_y) \cdot T(-x1, -y1)$ şi obţinem matricea $\begin{pmatrix} s_x & 0 & x_1(1-s_x) \\ 0 & s_y & y_1(1-s_y) \\ 0 & 0 & 1 \end{pmatrix}$. Se verifică că este o transformare afină dar nu şi de corp rigid. ▪

Proprietatea 5.6

Dacă M_1 şi M_2 reprezintă fie o translaţie, fie o rotaţie, fie o scalare atunci $M_1 \cdot M_2 = M_2 \cdot M_1$ în următoarele cazuri :

1. M_1 şi M_2 sunt translaţii (deoarece translaţiile sunt aditive),
2. M_1 şi M_2 sunt scalări (deoarece scalările sunt multiplicative),
3. M_1 şi M_2 sunt rotaţii (deoarece rotaţiile sunt aditive),
4. M_1 este o scalare ($s_x = s_y$) şi M_2 este o rotaţie.

▪

Demonstraţie
Ca exerciţiu.

□

5.4 Transformarea window-viewport

Unele pachete grafice permit specificarea coordonatelor primitivelor de ieşire în *sistemul de coordonate al domeniului (universului) aplicaţiei*[2] utilizând unităţi de măsură care au o semnificaţie în acest sistem de coordonate : microni, metri, ani lumină. *Universul (domeniul) aplicaţiei* reprezintă o lume (univers) care este creată de către programul aplicaţiei în mod interactiv sau este afişată (tot de către programul aplicaţiei) pentru utilizator.

Apare astfel necesitatea *conversiei* coordonatelor din universul aplicaţiei în coordonate ale dispozitivului de ieşire (ecran, etc.). Putem efectua această conversie în două moduri :

1. Programatorul poate furniza o matrice de transformare pentru realizarea conversiei,
2. Programatorul poate specifica o regiune dreptunghiulară în sistemul de coordonate al universului aplicaţiei (*fereastra universului aplicaţiei*[3]) şi o regiune dreptunghiulară corespondentă în coordonatele ecranului (dispozitiv de ieşire) denumită *viewport* în care fereastra universului aplicaţiei va fi convertită. Această transformare este *universală* : se aplică tuturor primitivelor de ieşire din fereastra universului aplicaţiei.

[2] *world-coordinate system* în engleză
[3] *world coordinate window* în engleză

Exemplul 5.5

În figura 5.7, pag. 59 am pus în evidenţă conversia ferestrei universului aplicaţiei în viewport. Dacă fereastra universului aplicaţiei şi viewport-ul nu au acelaşi raport înălţime/lungime atunci apare o scalare neuniformă. ▪

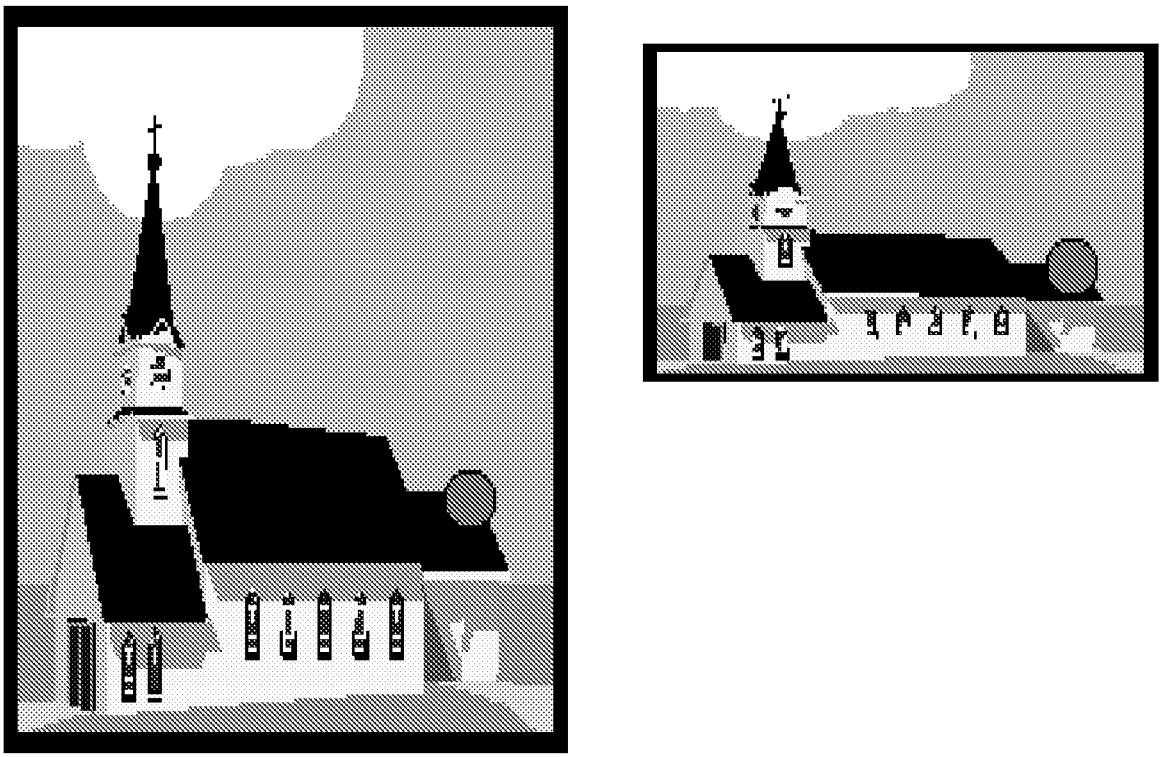


Figura 5.7: Conversia ferestrei universului aplicaţiei în viewport

Problema 5.1

Dată o fereastră (a universului aplicaţiei) şi un viewport, care este matricea de transformare care atribuie un punct din sistemul de coordonate al universului aplicaţiei unui punct din sistemul de coordonate ecran. ▪

Soluţie

Din punct de vedere grafic sunt efectuate următoarele transformări (vezi figura 5.6, pag. 57) :

1. translarea ferestrei universului aplicaţiei astfel încât colţul din stânga jos al ferestrei să ajungă în originea sistemului de coordonate al universului aplicaţiei,

2. scalarea ferestrei universului aplicaţiei pentru a căpăta dimensiunile viewport-ului,

3. translaţie pentru poziţionarea viewport-ului.

Obţinem astfel următoarea matrice : $M_{WV} = T(u_{min}, v_{min}) \cdot S(\frac{u_{max}-u_{min}}{x_{max}-x_{min}}, \frac{v_{max}-v_{min}}{y_{max}-y_{min}}) \cdot T(-x_{min}, -y_{min}) = \begin{pmatrix} \frac{u_{max}-u_{min}}{x_{max}-x_{min}} & 0 & -x_{min} \cdot \frac{u_{max}-u_{min}}{x_{max}-x_{min}} + u_{min} \\ 0 & \frac{v_{max}-v_{min}}{y_{max}-y_{min}} & -y_{min} \cdot \frac{v_{max}-v_{min}}{y_{max}-y_{min}} + v_{min} \\ 0 & 0 & 1 \end{pmatrix}$.

□

5.5 Eficienţa transformărilor geometrice

O secvenţă $(R^*S^*T^*)^*$ de transformări geometrice produce o matrice M de forma : $\begin{pmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$, unde matricea $\begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix}$ se obţine prin compunerea rotaţiilor R cu scalările S iar $t_{x,y}$ se obţin prin compunerea translaţiilor T.

Înmulţirea $M \cdot P$, unde P reprezintă vectorul coordonatelor unui punct, necesită, în general, 9 operaţii de înmulţire şi 6 operaţii de adunare. Deoarece ultimul rând al matricii M are forma (0 0 1) atunci produsul $M \cdot P$ se realizează numai cu 4 operaţii de înmulţire şi 4 operaţii de adunare : $x' = xr_{11} + yr_{12} + t_x$ şi $y' = xr_{21} + yr_{22} + t_y$. Această reducere este utilă de exemplu în cazul în care transformarea M trebuie aplicată miilor de puncte dintr-o anumită figură geometrică sau atunci când dorim să creem imagini succesive ale unui obiect (moleculă, avion) rotit doar cu câteva grade între 2 imagini succesive.

În acest din urmă caz (al rotaţiilor succesive doar cu câteva grade) putem ţine cont de faptul că unghiul de rotaţie $\theta \approx 0°$ şi deci $\cos\theta \approx 1$. Obţinem $x' = x - y\sin\theta$ şi $y' = x\sin\theta + y$ şi deci avem nevoie de 2 operaţii de înmulţire şi 2 operaţii de adunare. Pentru a micşora erorile care apar după aplicarea acestor ecuaţii de un număr mare de ori, înlocuim în a doua ecuaţie pe x cu x' : $x' = x - y\sin\theta$, $y' = x'\sin\theta + y = x\sin\theta - y\sin^2\theta + y = x\sin\theta + y\cos^2\theta$.

5.6 Reprezentarea matricială a transformărilor 3D

Aşa cum am reprezentat transformările 2D prin matrici $\in \mathcal{M}_{3\times3}(\mathbf{R})$ utilizând coordonate omogene aşa şi transformările 3D pot fi reprezentate prin matrici $\in \mathcal{M}_{4\times4}(\mathbf{R})$ dacă utilizăm reprezentarea în coordonate omogene a punctelor din spaţiul 3D.

Definiţia 5.9 (Coordonate omogene)
Spaţiul omogen 4D este spaţiul cartezian 4D fără origine (punctul $(0,0,0,0)$) şi deci tripletul (x,y,z,W) este un punct în *coordonate omogene* 4D dacă $x \neq 0 \vee y \neq 0 \vee z \neq 0 \vee W \neq 0$.

Unui punct cartezian 3D îi pot corespunde mai multe puncte omogene 4D şi anume : punctele omogene 4D (x, y, z, W) şi (x', y', z', W') reprezintă acelaşi punct cartezian 3D ddacă $(\exists \alpha \neq 0)(x' = \alpha x \wedge y' = \alpha y \wedge z' = \alpha z \wedge W' = \alpha W)$.

Punctele $(x, y, z, 0)$ se numesc *puncte omogene la infinit.*

Numim transformare de *omogenizare* acea transformare care asociază unui punct omogen 4D (x, y, z, W), $W \neq 0$, punctul omogen 4D $(\frac{x}{W}, \frac{y}{W}, \frac{z}{W}, 1)$. ▪

Observaţia 5.3

Un *sistem de coordonate cartezian* se defineşte prin originea O şi trei axe perpendiculare Ox,Oy,Oz, orientate după regula mâinii drepte sau după regula mâinii stângi (vezi figura 5.8, pag. 61). ▪

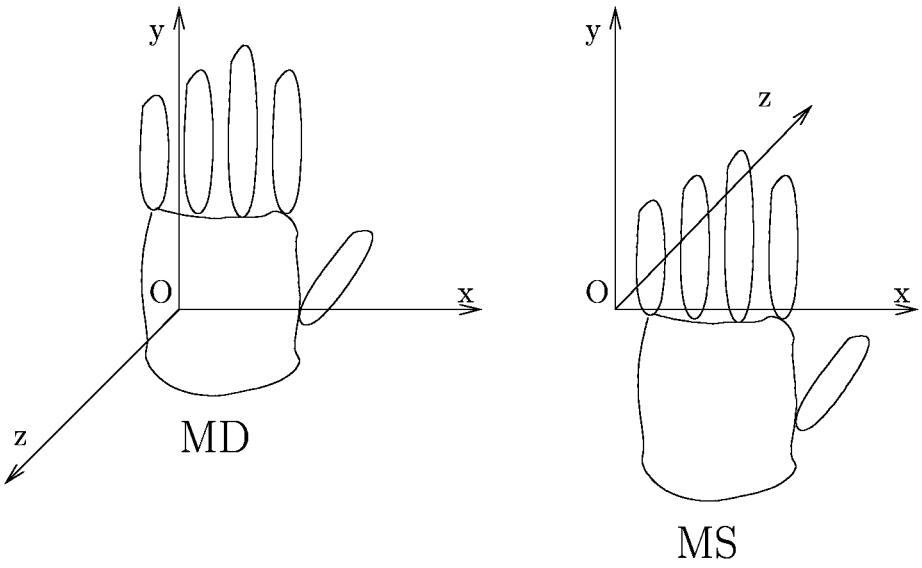


Figura 5.8: Regulile mâinii drepte şi stângi

Definiţia 5.10 (Translaţie 3D în coordonate omogene)

Se numeşte *translaţie* (notată $T(d_x, d_y, d_z)$) o transformare geometrică care asociază oricărui punct $P(x, y, z)$ un punct $P'(x', y', z')$ cu proprietatea că $x' = x + d_x$, $y' = y + d_y$ şi $z' = z + d_z$.

Putem exprima matricial acest lucru :

$$P' = T(d_x, d_y, d_z) \cdot P, \tag{5.8}$$

unde $P' = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$ şi $T(d_x, d_y, d_z) = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$. ▪

Definiţia 5.11 (Scalare 3D în coordonate omogene)

Se numeşte *scalare* (notată $S(s_x, s_y, s_z)$) o transformare geometrică care asociază oricărui punct $P(x, y, z)$ un punct $P'(x', y', z')$ cu proprietatea că $x' = x \cdot s_x$, $y' = y \cdot s_y$ şi $z' = z \cdot s_z$.

Putem exprima matricial acest lucru :

$$P' = S(s_x, s_y, s_z) \cdot P, \tag{5.9}$$

unde $P' = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$ şi $S(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$. ▪

Definiţia 5.12 (Rotaţie 3D în coordonate omogene)

Transformarea de *rotaţie 3D* se obţine prin trei tipuri de rotaţii :

1. Rotaţia în raport cu axa Oz, dată prin matricea $R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

2. Rotaţia în raport cu axa Ox , dată prin matricea $R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

3. Rotaţia în raport cu axa Oy , dată prin matricea $R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

▪

Proprietatea 5.7

Matricile formate din primele trei linii şi trei coloane ale matricilor $R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$ sunt matrici ortogonale proprii. Aceeaşi proprietate o are orice matrice corespunzătoare unei secvenţe arbitrare de transformări de rotaţie. În concluzie, o secvenţa arbitrară de transformări de rotaţie păstrează unghiurile şi distanţele. ▪

Demonstraţie

Ca exerciţiu.

□

Proprietatea 5.8

Transformările geometrice inverse[4] translaţiei $T(d_x, d_y, d_z)$, scalării $S(s_x, s_y, s_z)$ sau rotaţiei $R_{x,y,z}(\theta)$ sunt respectiv $T(-d_x, -d_y, -d_z)$, $S(\frac{1}{s_x}, \frac{1}{s_y}, \frac{1}{s_z})$ şi $R_{x,y,z}(-\theta)$. ▪

[4]produsul matricilor respective este matricea identitate I_4

Demonstraţie
Ca exerciţiu.

□

Proprietatea 5.9
Fie B matricea corespunzătoare unei secvenţe arbitrare de transformări de rotaţie. Atunci avem $B^{-1} = {}^tB$. ▪

Demonstraţie
Ca exerciţiu.

□

Definiţia 5.13 (Transformare forfecată 3D)
Se numeşte *transformare forfecată* în raport cu axele Ox, Oy (notată $SH_{xy}(sh_x, sh_y)$) o transformare geometrică care asociază oricărui punct $P(x, y, z)$ un punct $P'(x', y', z')$ cu proprietatea că $x' = x + sh_x \cdot z$, $y' = y + sh_y \cdot z$, $z' = z$.

Putem exprima matricial acest lucru :

$$P' = SH_{xy}(sh_x, sh_y) \cdot P, \tag{5.10}$$

unde $P' = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$ şi $SH_{xy}(sh_x, sh_y) = \begin{pmatrix} 1 & 0 & sh_x & 0 \\ 0 & 1 & sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

O transformare forfecată în raport cu axele Ox, Oz (notată $SH_{xz}(sh_x, sh_z)$) este caracterizată matricial cu ajutorul matricii $SH_{xz}(sh_x, sh_z) = \begin{pmatrix} 1 & sh_x & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & sh_z & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

O transformare forfecată în raport cu axele Oy, Oz (notată $SH_{yz}(sh_y, sh_z)$) este caracterizată matricial cu ajutorul matricii $SH_{yz}(sh_y, sh_z) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ sh_x & 1 & 0 & 0 \\ sh_z & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$. ▪

Observaţia 5.4

1. Transformarea geometrică a unui segment de dreaptă se face efectuând transformarea extremităţilor segmentului şi apoi trasând segmentul prin cele două puncte.

2. Transformarea geometrică a unui plan se face alegând trei puncte necoliniare din plan şi aplicând transformarea geometrică acestor trei puncte.

3. Pentru transformarea geometrică a unui plan putem proceda şi mai simplu : să presupunem că ecuaţia planului este $Ax + By + Cz + D = 0$. Putem reprezenta această ecuaţie în două moduri :

(a) ca un produs scalar : $N \cdot P = 0$, unde $N = \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix}$ şi $P = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$,

(b) ca un produs de matrici : ${}^tN \cdot P = 0$.

Fie M matricea de transformare şi să presupunem că dorim pentru planul transformat să avem o ecuaţie similară cu cea de mai sus : ${}^tN' \cdot P' = 0$, unde $P' = M \cdot P$. Să presupunem că N' este obţinută prin aplicarea transformării Q : $N' = Q \cdot N$. Deci avem ${}^t(QN)MP = 0 \iff {}^tN{}^tQMP = 0 \Rightarrow {}^tQM = \lambda I$. Presupunem că ${}^tQM = I \Rightarrow {}^tQ = M^{-1} \iff Q = {}^t(M^{-1})$. Deci $N' = {}^t(M^{-1})N$, în cazul în care există matricea inversă M^{-1}.

▪

5.7 Compunerea transformărilor 3D

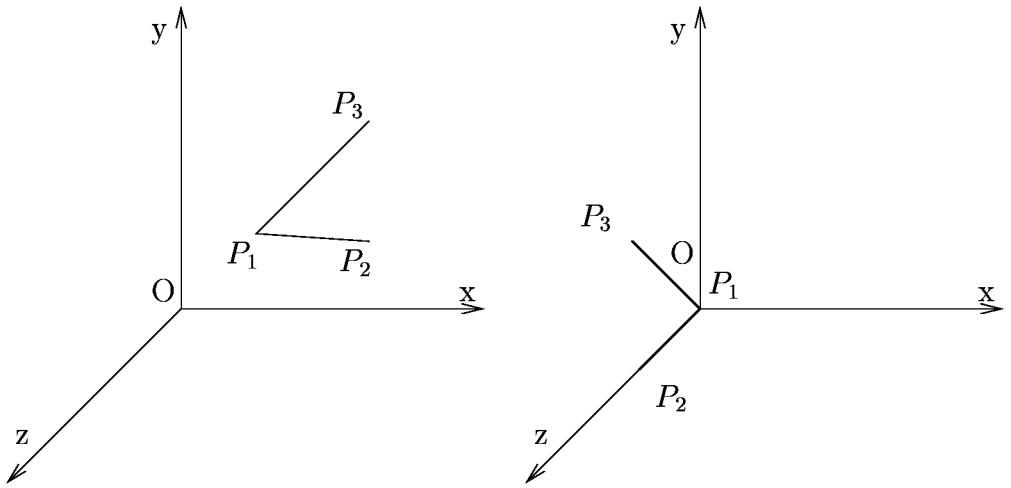


Figura 5.9: Transformare 3D compusă

Fie problema următoare :

Problema 5.2

Să se obţină o matrice de transformare astfel încât segmentele P_1P_2 şi P_1P_3 (şi planul determinat de aceste puncte) să se transforme aşa cum este arătat în figura 5.9, pag. 64 : în poziţia finală segmentul P_1P_2 se află pe axa Oz iar P_1P_3 se află în planul Oyz (în cadranul I). În plus, dorim să nu afectăm, prin această transformare, lungimea segmentelor (i.e., să fie o transformare de corp rigid). ▪

Soluţie

Vom împărţi această problemă în probleme mai simple :

1. Translarea punctului P_1 în originea sistemului de coordonate. Obţinem astfel punctele $(P_i')_{1\leq i\leq 3}$.

2. Rotaţie în jurul axei Oy astfel încât $P_1'P_2' \in Oyz$. Obţinem astfel punctele $(P_i'')_{1\leq i\leq 3}$.

3. Rotaţie în jurul axei Ox astfel încât $P_1''P_2'' \in Oz$. Obţinem astfel punctele $(P_i''')_{1\leq i\leq 3}$.

4. Rotaţie în jurul axei Oz astfel încât $P_1'''P_3''' \in Oyz$.

Problema 1, pag. 65 se rezolvă aplicând transformarea $T(-x_1, -y_1, -z_1)$. Punctele $(P_i)_{1\leq i\leq 3}$ devin punctele $(P_i')_{1\leq i\leq 3}$, unde $P_1' = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$, $P_2' = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \\ 1 \end{pmatrix}$ şi $P_3' = \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \\ 1 \end{pmatrix}$.

Problema 2, pag. 65 se rezolvă astfel : fie $Q(x_2', 0, z_2')$ proiecţia punctului P_2' pe planul Oxz. Fie $D_1 = \mid P_1'Q \mid = \sqrt{x_2'^2 + z_2'^2} = \sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2}$. Efectuăm o rotaţie în jurul axei Oy astfel încât $P_2'' \in Oyz$ iar proiecţia punctului P_2'' pe planul Oxz să se găsească pe axa Oz. Fie θ unghiul $\widehat{QP_1', Ox}$. Vom aplica o rotaţie de unghi $-(90-\theta)$: $R_y(\theta - 90)$ şi vom obţine matricea $\begin{pmatrix} \cos(\theta - 90) & 0 & \sin(\theta - 90) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta - 90) & 0 & \cos(\theta - 90) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ şi deci matricea $\begin{pmatrix} \frac{z_2 - z_1}{D_1} & 0 & -\frac{x_2 - x_1}{D_1} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{x_2 - x_1}{D_1} & 0 & \frac{z_2 - z_1}{D_1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Punctele $(P_i')_{1\leq i\leq 3}$ devin punctele $(P_i'')_{1\leq i\leq 3}$, unde, de exemplu $P_2'' = \begin{pmatrix} 0 \\ y_2 - y_1 \\ D_1 \\ 1 \end{pmatrix}$ (evident $P_2'' \in Oyz$).

Problema 3, pag. 65 se rezolvă aplicând o rotaţie în jurul axei Ox de unghi $\varphi = \widehat{P_1''P_2'', Oz}$: $R_x(\varphi)$. Unghiul φ se obţine ştiind că $\cos\varphi = \frac{z_2''}{D_2}$ şi $\sin\varphi = \frac{y_2''}{D_2}$, unde $D_2 = \sqrt{y_2''^2 + z_2''^2} = \sqrt{(y_2 - y_1)^2 + D_1^2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$. Deci $\sin\varphi = \frac{y_2 - y_1}{D_2}$ şi $\cos\varphi = \frac{D_1}{D_2}$. Obţinem deci matricea $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{D_1}{D_2} & -\frac{y_2 - y_1}{D_2} & 0 \\ 0 & \frac{y_2 - y_1}{D_2} & \frac{D_1}{D_2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$. În concluzie, punctele $(P_i'')_{1\leq i\leq 3}$ devin

punctele $(P_i''')_{1\le i\le 3}$, unde, de exemplu $P_2''' = \begin{pmatrix} 0 \\ 0 \\ D_2 \\ 1 \end{pmatrix}$, unde $D_2 = \mid P_1'P_2' \mid = \mid P_1P_2 \mid$.

Problema 4, pag. 65 se rezolvă aplicând o rotaţie $R_z(\alpha)$, unde $\alpha = \widehat{OQ, Oy}$ iar Q este proiecţia pe planul Oxy a punctului P_3'''. Fie $D_3 = \mid OQ \mid = \sqrt{x_3'''^2 + y_3'''^2}$ şi atunci avem $\cos\alpha = \frac{y_3'''}{D_3}$ şi $\sin\alpha = \frac{x_3'''}{D_3}$.

În concluzie, matricea transformării este $R_z(\alpha)R_x(\varphi)R_y(\theta - 90)T(-x_1, -y_1, -z_1)$ şi este matricea $\begin{pmatrix} \frac{z_2-z_1}{D_1} & 0 & -\frac{x_2-x_1}{D_1} & \frac{x_2z_1-x_1z_2}{D_1} \\ -\frac{(x_2-x_1)(y_2-y_1)}{D_1D_2} & \frac{D_1}{D_2} & -\frac{(y_2-y_1)(z_2-z_1)}{D_1D_2} & \frac{(y_1-y_2)(x_1^2+z_1^2-x_1x_2-y_1z_2)}{D_1D_2} - \frac{y_1y_2}{D_2} \\ \frac{x_2-x_1}{D_2} & \frac{y_2-y_1}{D_2} & \frac{z_2-z_1}{D_2} & \frac{x_1^2+y_1^2+z_1^2-x_1x_2-y_1y_2-z_1z_2}{D_2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

□

5.8 Transformări geometrice văzute ca transformări ale sistemului de coordonate

Putem vedea transformările geometrice în două moduri :

1. Ca o transformare a unei mulţimi de puncte în altă mulţime de puncte, în acelaşi sistem de coordonate.

2. Ca o schimbare a sistemului de coordonate. Această modalitate de a vedea transformările geometrice este utilă în cazul în care dorim să combinăm diverse obiecte, fiecare definite în propriul sistem de coordonate şi se doreşte exprimarea coordonatelor acestor obiecte într-un singur sistem global de coordonate.

Fie $M_{i\leftarrow j}$ transformarea care converteşte reprezentarea unui punct din sistemul de coordonate j în reprezentarea sa în sistemul de coordonate i. Fie $P^{(i)}$, $P^{(j)}$, $P^{(k)}$ reprezentările unui punct în sistemele de coordonate i, j, k. Avem atunci relaţiile $P^{(i)} = M_{i\leftarrow j} \cdot P^{(j)}$ şi $P^{(j)} = M_{j\leftarrow k} \cdot P^{(k)}$ pe baza cărora putem obţine $P^{(i)} = M_{i\leftarrow j} \cdot P^{(j)} = M_{i\leftarrow j} \cdot M_{j\leftarrow k} \cdot P^{(k)} = M_{i\leftarrow k} \cdot P^{(k)}$. Deci se obţine $M_{i\leftarrow k} = M_{i\leftarrow j} \cdot M_{j\leftarrow k}$.

Exemplul 5.6

În figura 5.10, pag. 67 avem patru sisteme de coordonate diferite. Avem relaţiile : $M_{1\leftarrow 2} = T(4,2)$, $M_{2\leftarrow 3} = T(2,3) \cdot S(\frac{1}{2}, \frac{1}{2})$, $M_{3\leftarrow 4} = T(8 - \sqrt{2}, 6 - 3\sqrt{2}) \cdot R(+45°)$. Deci $M_{1\leftarrow 3} = M_{1\leftarrow 2} \cdot M_{2\leftarrow 3} = T(4,2) \cdot T(2,3) \cdot S(\frac{1}{2}, \frac{1}{2})$. În plus avem $P^{(1)}(10,8)$, $P^{(2)}(6,6)$, $P^{(3)}(8,6)$ şi $P^{(4)}(4,2)$.

Se mai poate observa că $M_{i\leftarrow j} = M_{j\leftarrow i}^{-1}$. ▪

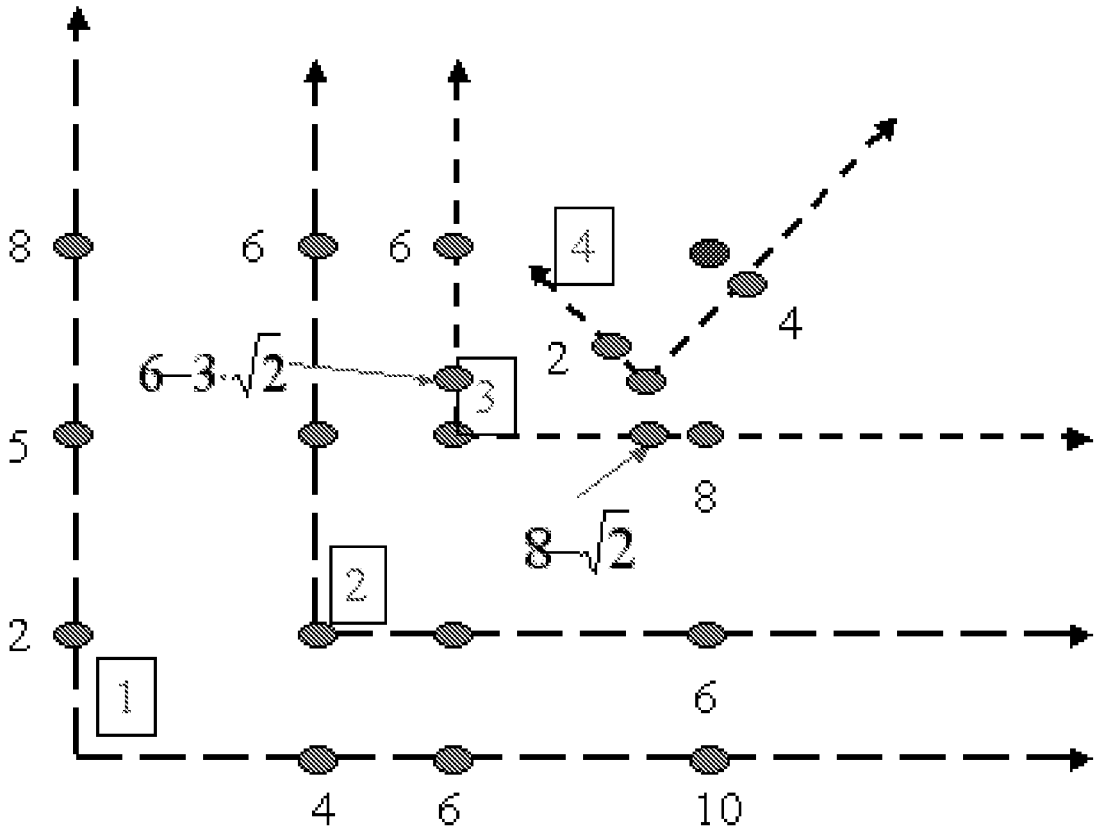


Figura 5.10: Coordonatele unui punct în mai multe sisteme de coordonate

Observaţia 5.5

Matricea transformării punctelor din sistemul de coordonate ce respectă regula mâinii drepte într-un sistem de coordonate ce respectă regula mâinii stângi (şi viceversa) este

$$M_{L\leftarrow R} = M_{R\leftarrow L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$ ▪

Am văzut în secţiunile precedente că putem aborda transformările geometrice ca având loc în acelaşi sistem de coordonate (ceea ce presupune că obiectele sunt definite iniţial unul peste celălalt în acelaşi sistem de coordonate şi de aici sunt transformate în poziţia dorită). Mai putem avea şi altă perspectivă : fiecare obiect este definit în propriul sistem de coordonate şi apoi este scalat, rotit, translatat prin redefinirea coordonatelor sale în noul sistem de coordonate.

Să considerăm exemplul 5.5, pag. 57. Acesta presupunea aplicarea translaţiei $T(-x_1, -y_1)$ astfel încât punctul $P(x_1, y_1)$ să ajungă în originea sistemului de coordonate. Putem vedea lucrurile şi altfel : în figura 5.11, pag. 68 se observă că transformarea sistemului de coordonate 1 în sistemul de coordonate 2 este $M_{2\leftarrow 1} = T(-x_1, -y_1)$ şi deci $M_{1\leftarrow 2} = T(x_1, y_1)$. Regula este următoarea : putem fie aplica o transformare geometrică M într-un unic sistem de coordonate fie schimba sistemul de coordonate iniţial i într-un sistem de coordonate j şi în acest caz avem $M_{i\leftarrow j} = M^{-1}$.

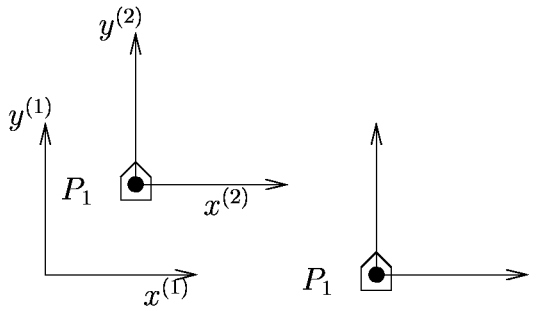


Figura 5.11: Transformarea sistemului de coordonate

Capitolul 6

Bibliografie

Pentru realizarea acestei lucrări au fost folosite următoarele referinţe bibliografice :

1. F. Ionescu, *Grafica în realitatea virtuala*, Ed.Tehnica 2000,
2. M. Vlada, I. Nistor, A. Posea, C. Constantinescu, *Grafica pe calculator în limbajele Pascal si C*, Ed. Tehnica 1991,
3. J.D. Foley, A.v. Dam, S. Feiner, J. Hughes, *Computer Graphics: Principles & Practice in C (2nd edition)*, Addison- Wesley 1995,
4. C.-D. Neagu, S. Bumbaru, *Sisteme multimedia - Grafica pe calculator*, Ed. Matrix Rom, 2001,
5. D. Hearn, M.P. Baker, *Computer Graphics, C Version (2nd Edition)*, Prentice Hall 1996,
6. L. Raicu, *Grafic şi vizual între clasic şi modern*, Ed. Paideia, 2000,
7. F. Moldoveanu, *Grafica pe calculator*, Ed. Teora, 1996.