

13. Grafica animată și imagini dinamice

1. Principii generale

Cea mai mare parte din informația manipulată de sistemele de calcul actuale este sub formă grafică. Tehnologia Microsoft a impus practic un standard în realizarea interfețelor grafice bazate pe Windows, dar în prezent aproape orice aplicație profesională se bazează pe interfață grafică – Graphical User Interface (GUI). (De pildă, Linux-ul care în mod tradițional se baza pe o comunicare cu utilizatorul de tip textual orientată pe linia de comandă, oferă în prezent interfață grafică).

În prezent, o bună parte din caracteristicile sistemelor de calcul se raportează la performanțele de a prelucra informația video. Acestea se referă în special la *capacitatea memoriei alocată informației grafice și la viteza de prelucrare a datelor aferente imaginilor*. Cerințele tot mai mari pentru prelucrări grafice de mare viteză au condus la dezvoltarea de hardware specializat sub forma unor *adaptoare video (plăci grafice)* performante. Concomitent cu acestea au crescut continuu și performanțele terminalelor de afișare mai ales caracteristicile: *dimensiunea de afișare, rezoluția, nivelul de contrast și timpul de răspuns*.

Formatul în care informația grafică este manipulată în calculatoarele personale este reglementat prin câteva standarde în funcție de *tipul aplicației*. În mod evident, fiecare format de codificare a informației grafice este accesibil prin intermediul diferitelor utilitare pentru vizualizarea și prelucrarea imaginilor respective.

În continuare dăm o clasificare a tipurilor de imagini și o descriere sumară a *formatelor* de uz general în grafica pe calculator. Tipurile de imagine în grafica pe calculator și metodologia de lucru cu imaginile sunt strâns legate de *tipul aplicației* în sine.

În principiu **tipurile de imagini** sunt:

- imagini statice
- interfețe grafice **interactive**
- **imagini dinamice, animație**
- aplicații video

Formate

Instrumentele software pentru creație multimedia acceptă anumite **formate** (tipuri) de fișiere ce conțin date imagine și de regulă sunet. Există o multitudine de formate diferite pentru imagini, dar numai câteva sunt folosite în mod curent, pe scară largă (cca. 40).

Graphics file formats	
Raster	ANI · ANIM · APNG · ART · BMP · BPG · BSAVE · CAL · CIN · CPC · CPT · DDS · DPX · ECW · EXR · FITS · FLIC · FLIF · FPX · GIF · HDRI · HEVC · ICER · ICNS · ICO · CUR · ICS · ILBM · JBIG · JBIG2 · JNG · JPEG · JPEG-LS · JPEG 2000 · JPEG XR · JPEG XT (JPEG-HDR) · JPEG XL · KRA · MNG · MIF · MIFF · NRRD · PAM · PBM · PGM · PPM · PNM · PCX · PGF · PICTor · PNG · PSD · PSB · PSP · QTVR · RAS · RGBE (Logluv TIFF) · SGI · TGA · TIFF (TIFF/EP · TIFF/IT) · UFO · UFP · WBMP · WebP · XBM · XCF · XPM · XWD
Raw	CIF · DNG
Vector	AI · CDR · CGM · DXF · EVA · EMF · EMF+ · Gerber · HVIF · IGES · PGML · SVG · VML · WMF · Xar
Compound	CDF · DjVu · EPS · PDF · PICT · PS · SWF · XAML
Metadata	Exchangeable image file format (Exif) · International Press Telecommunications Council § Photo metadata · Extensible Metadata Platform (XMP) · GIF § Metadata · Steganography

(Sursa : https://en.wikipedia.org/wiki/Image_file_formats)

Următorul tabel prezintă o listă cu cele mai utilizate formate de imagini (extensii de fișiere).

Format	Standardul	Utilitar specializat
.bmp	Windows Bitmap (BMP) 1-bit, 4-bit, 8-bit, and 24-bit uncompressed images; 4-bit and 8-bit run-length encoded (RLE) images	Paint
.pcx	ZSoft Publisher's Paintbrush 1-bit, 8-bit, and 24-bit images	Paint
.wmf	Windows Metafile	
.pcd	Kodak (Photo CD)	ACDSee
.tif sau .tiff	Tagged Image File Format (TIFF) 1-bit, 8-bit și 24-bit uncompressed images; 1-bit, 8-bit, and 24-bit images with packbits compression; 1-bit images with CCITT compression; also, 16-bit grayscale, 16-bit indexed, and 48-bit RGB images	Imaging
.jpg sau .jpeg	Joint Photographic Experts Group (JPEG)	Imaging, Adobe
.gif	Graphic Interchange Format	Medii grafice vectoriale cu opțiuni de compresie
.png	Portable Network Graphics including 1-bit, 2-bit, 4-bit, 8-bit, and 16-bit grayscale images; 8-bit and 16-bit indexed images; 24-bit and 48-bit RGB images	Adobe PhotoShop Corel PaintShop
.hdf	Hierarchical Data Format (HDF) 8-bit raster image datasets, with or without an associated colormap; 24-bit raster image datasets	HDF Product Designer Instrument interoperabil de creare a produselor de date HDF5 GUI HDF Explorer Un program de vizualizare a datelor care citește formatele de fișiere de date HDF, HDF5 și netCDF HDFView Un browser și editor pentru fișiere HDF ViTables Un browser și editor pentru fișiere HDF5 și PyTables scrise în Python
.xwd	X Window Dump (XWD) 1-bit and 8-bit ZPixmap; XYBitmaps; 1-bit XYPixmap	GIMP, ImageMagick, Netpbm, XnView
.tga	Format Targa pentru Televiziune	Utilizat pentru 3D și jocuri video
.avi	Audio-Video Interleaved	Specific Microsoft, ACDSee
.mpeg .mp3	Moving Picture Experts Group	Winamp
.mov	Quicktime	Quicktime
.swf	Flash Player format	Shockwave
.flc și .fli	Filme de animație	Autodesk

Pentru o comparație a formatelor de fișiere grafice se poate accesa legătura web următoare:

https://en.wikipedia.org/wiki/Comparison_of_graphics_file_formats

Un program de creație pentru *multimedia* acceptă să importe mai multe formate de imagini. Există însă software pentru conversia de format care oferă multe instrumente pentru reglarea în detaliu a procesului de conversie. De exemplu : programe cunoscute pentru Windows sunt ACDSee produs de ACD Systems Ltd., HiJaakProfessional produs de Inset Systems și ImagePals produs de U-LeadSystems. Un alt exemplu este Adobe Creative Suite 4 care include și modulul Animation creator.

Metodologia de lucru cu imaginile presupune anumite operații standard, după cum urmează:

- 1) editarea grafică;
- 2) captura imaginii;
- 3) aplicarea corecțiilor / filtrelor;
- 4) aplicarea **efectelor**;
- 5) organizarea scenelor;
- 6) montajul;
- 7) redarea (afișarea) aplicației.

În mod evident, orice aplicație de grafică pe calculator presupune și operația de *salvare* ca fișier într-un format specific, în vederea stocării.

O operație esențială, mai ales pentru aplicațiile video este *comprimarea datelor video*, care are ca rezultat micșorarea dimensiunilor aplicațiilor video în vederea înregistrării și transmiterii acestora. Redarea unei aplicații video presupune operația inversă, de decompresie. Operațiile de compresie-decompresie se realizează pe baza unor algoritmi speciali care sunt implementați sub forma de “*codec*” atât software cât și hardware.

2. Dezvoltarea aplicațiilor de animație

Animația cu ajutorul calculatorului este o aplicație de tip *movie*. Realizarea acesteia presupune atât operații de *grafică* propriu-zisă cât și operații de *organizare a scenelor* și de *montaj*.

Pentru a avea o viziune cât mai clară asupra filozofiei de realizare a aplicațiilor cu imagini dinamice vom da schema generală din Figura 13.1, care pune în evidență problematica generală în acest domeniu.

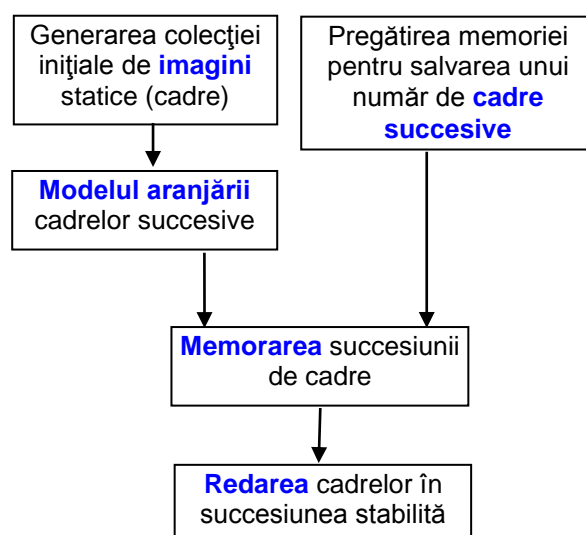


Figura 13.1.

- 1) **Editarea** unei aplicații de grafică pe calculator se poate efectua la mai multe nivele. Există un nivel comun, accesibil tuturor programatorilor cu ajutorul limbajelor de programare de uz general, prin utilizarea funcțiilor grafice ale acestora. La acest nivel, programatorul trebuie să conceapă modelul aplicației grafice, să gândească algoritmul de implementare a problemei și să scrie efectiv codul programului, care va afișa grafica dorită. Nivelul profesionist de programare grafică presupune existența unui *software specializat* performant, care permite dezvoltarea de aplicații grafice pe baza de instrumente software specializate pentru definirea obiectelor complexe și a relațiilor dintre acestea, precum și pentru aplicarea unor efecte grafice avansate. În cazul aplicațiilor de animație, rezultatul editării se regăsește în setul de imagini statice inițiale, care vor constitui cadrele filmului.
- 2) **Captura** imaginilor constituie operația de import (achiziție) a informației grafice de la un dispozitiv extern calculatorului de tip videocaptor sau scanner, prin intermediul unei plăci (hardware) specializat. Camerele video obișnuite au dispozitive sensibile la lumină de tip CCD (charge-coupled device), care captează imaginea direct în format numeric. Aceste semnale pot fi

apoi manipulate în diverse scopuri : transmitere la distanță, înregistrare digitală pe discuri magnetice sau optice, etc.

- 3) *Corecția de imagine* constă în operațiile de ajustare a luminozității (brightness), contrastului (contrast), a culorilor (color) prin intermediul saturației (saturation) respectiv a nuanței (hue) și a clarității imaginii (sharpness). Se pot aplica multe alte filtre pentru prelucrarea imaginilor.
- 4) *Aplicarea efectelor* asupra imaginilor constituie operații opționale utilizate cu precădere în aplicațiile profesionale pentru crearea unor senzații vizuale deosebite. Majoritatea browserelor pentru imagini permit aplicarea unor efecte prin intermediul așa numitelor *filtre*. Filtrele sunt funcții de prelucrare a imaginii care poartă denumirea efectului pe care îl realizează. În tabelul de mai jos se descriu cele mai utilizate filtre în browserele de imagini.

Denumirea filtrului	Descrierea Efectului	Exemplificare/Aplicații
Blur	Încețosează imaginea ducând la <u>scăderea clarității</u> prin eliminarea zgomotului în porțiunile de imagine unde se produc schimbări importante de culoare.	Vezi functii specializate din Image Processing Toolbox din Matlab
Invert (negative)	Schimbă fiecare culoare din imagine în opusa din spectrul culorilor.	
Emboss	<u>Reliefează</u> imaginea prin suprimarea culorilor și trasarea contururilor cu negru.	
Find Edges	Pune în evidență contururile obiectelor din imagine	

În mediile de editare grafică profesionale cum sunt Adobe Premiere, Adobe Photoshop, Corel Draw, etc. sunt puse la dispoziția utilizatorului o gamă mult mai largă de efecte, împreună cu instrumente de setare a opțiunilor de aplicare a acestora asupra imaginilor care se editează.

În mediile dedicate pentru producția de filme și animație se pot aplica efecte speciale asociate cu tipuri particulare de dinamică.

Tot în această etapă se pot activa diverse acțiuni de editare a imaginilor prin aplicarea de text sau alte elemente grafice suplimentare în imaginea curentă.

- 5) *Organizarea scenelor* este operația cea mai importantă într-o aplicație de tip *movie*. Modelul aranjării cadrelor succesive depinde de cinematica care se dorește a fi obținută în aplicația respectivă. Problema de bază care se pune în aplicațiile de animație este legată de **realismul mișcărilor** obiectelor grafice. Pentru realizarea unor imagini dinamice realiste, cum sunt cele care conțin mișcări ale ființelor, sau în general, mișcări guvernate de legi fizice cunoscute, în procesul de realizare (editare) a scenelor grafice se rezolvă două probleme:
- editarea mișcării (motion editing);
 - stabilirea vitezei de baleiaj/sucesiune a scenelor (time base).
- 6) *Montajul* unui film în general constituie operația de asamblare a secvențelor (porțiunilor) de film, conform scenariului dorit. De asemenea în această etapă se elimină unele porțiuni greșite și se adaugă eventuale secvențe de început și sfârșit.

7) **Redarea** animației este operația curentă care se efectuează ori de câte ori se dorește vizualizarea aplicației. Redarea aplicației este echivalentă comenzii *play* care se activează din browserele specializate pentru aplicații multimedia.

A) Editarea mișcării

Editarea mișcării în grafica pe calculator constituie de la caz la caz o problemă cu grad de dificultate variabil, ce depinde în primul rând de complexitatea scenelor. În funcție de software-ul utilizat pentru grafică realizarea animației poate deveni foarte productivă și totodată realistă sau dimpotrivă, extrem de laborioasă și mai puțin realistă. Astfel se pot folosi cele mai simple editoare grafice (cum sunt Paint sau chiar Drawing din MSWord) pentru a desena liber diferite scene statice ce compun animația, iar prețul plătit va fi timpul consumat și mai ales lipsa de realism a scenelor obținute. Editoarele grafice din categoria celor menționate mai sus pot fi folosite pentru realizare de pictograme sau iconițe cu un număr mic de elemente animate.

Dezvoltarea aplicațiilor de grafică avansată presupune utilizarea de medii software profesionale, specializate pentru modelarea grafică realistă a unor medii și fenomene diverse precum și pentru realizarea unei animații cu *consistență umană*. Toate aceste facilități au un impact deosebit în dezvoltarea de jocuri (PC Games) și în realizarea aplicațiilor multimedia profesionale (industria de publicitate). Astfel de medii de programare grafică avansată sunt: 3D Studio Max, Character Studio, LightScape, Maya5, etc., (în general cu prețuri ce variază de la mii la câteva sute de dolari). Unity este un mediu grafic performant care are și varianta free.

B) Stabilirea vitezei de succedare a scenelor (*time base*).

Viteza de succedare (baleiaj) a scenelor (cadrelor) se măsoară în cadre/secundă (frames per second –fps). Acest parametru este fundamental în grafica animată ca și în cinematografie și televiziune de altfel. Există câteva standarde cu privire la acest parametru:

- 30 fps -valoare rotunjită a standardului NTSC (National Television Standards Committee) specific înregistrărilor video care nu se transmit prin televiziune. (Standardul NTSC pentru transmiterea de calitate a înregistrărilor video prevede 29,97fps);
- 25 fps –specifică standardului European de televiziune;
- 24 fps –viteza la care se proiectează în mod normal un film.

Valoarea acestui parametru este dictată în principal din considerente de percepție a imaginilor dinamice de către ochiul uman (analizorul vizual). Acesta este adaptat să perceapă mișcările cu o continuitate naturală dacă viteza de succedare a cadrelor este de circa 24fps. O viteză de derulare mai mică provoacă senzația de discontinuitate a mișcării (*mișcare sacadată*), iar o viteză mai mare conduce evident la o mișcare mai rapidă (așa numita ”*mișcare cu repezitorul*”).

Viteza de succedare a cadrelor în aplicațiile de animație are implicații asupra volumului de scene statice ce trebuie realizate. Astfel cu cât mai mult se dorește realizarea unor mișcări mai expresive (în fond mai aproape de realitate) sunt necesare *cu atât mai multe poziții intermediare* desenate între o scenă considerată inițială și cea finală – *numărul standard de imagini fiind de 24 într-o secundă*.

Redarea lentă a imaginilor pentru fenomene rapide – așa numita “mișcare cu încetinitorul” se realizează prin supra-eșantionare a procesului dinamic cu o viteză mult mai mare decât 24fps (de exemplu 500fps). Aceste imagini se obțin cu camere de filmare ultrarapide și imaginile se redau apoi succesiv tot câte 24 pe secundă, ceea ce face ca durata să fie de cca 20 de ori mai lungă (500/24) astfel încât sunt percepute detalii dinamice ale procesului care în mod normal ar fi “scăpat ochiului”.

Exemplu: imagini preluate selectiv dintr-un proces rapid filmat cu 10.000fps. Durata procesului este mai mică de o secundă. Cantitatea de date corespunde pentru câteva mii de imagini.



(Filmăat cu *Photron Fastcam SA-X2 model 1000K*)

C) Organizarea scenelor cu instrumentul *storyboard*

Storyboard este un instrument clasic folosit în crearea aplicațiilor de animație. Acest instrument este asimilat cu o planșă de lucru 2D pe care sunt desenate un set de imagini statice sub formă de schițe (sketches) în succesiunea dorită. Pentru a realiza aplicații de animație, instrumentul storyboard este integrat într-un mediu software "inteligent", care rezolvă două probleme tipice:

- estimarea (costrucția) obiectelor/scenelor în 3D pe baza desenelor în 2D;
- generarea mișcărilor continue pe baza scenelor desenate în storyboard .

În figura 13.2 este prezentat conceptul de storyboard inteligent. Estimarea graficii 3D pe baza scenelor în 2D se face prin stabilirea relației dintre linia din planul imaginii și o parte a obiectului 3D. Pentru generarea mișcării continue se pleacă de la imaginile din storyboard, care în general reprezintă o descriere grosieră a mișcării elementelor și se efectuează o interpolare între două imagini succesive.

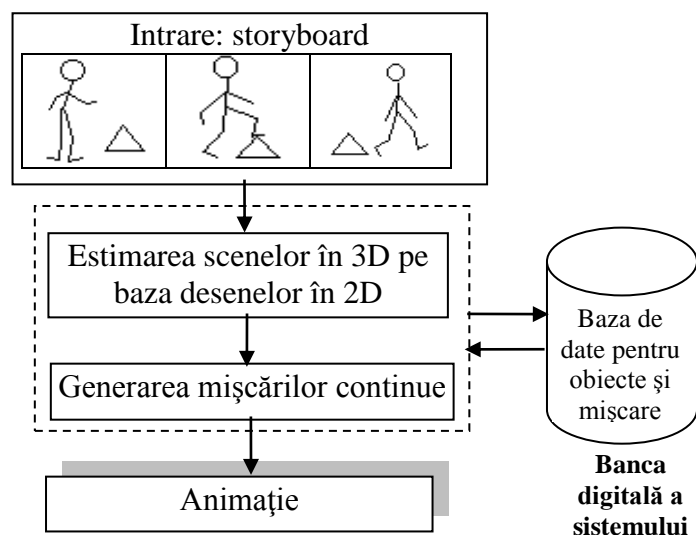


Figura 13.2.

De exemplu, în Unity animația se face într-o fereastră storyboard, folosind o tehnică tradițională de animație cu *cadru cheie*. Cadrele cheie sunt imagini de referință dintr-o cronologie a clipului de animație care conțin modificări ale obiectelor. Când animația este redată, Unity *interpolează* datele de la un cadru cheie la altul pentru a reda animația.

Unity poate să **controleze interpolarea datelor** de la un cadru cheie la altul folosind *curbe*. Unity oferă acces la instrumentul Curbe pentru a controla exact modul în care se va face interpolarea între

cadrele cheie. Acest lucru se face dacă interpolarea nu surprinde în mod realist mișcarea pe care un utilizator ar putea să o aibă în vedere.

De exemplu, crearea unei animații de săritură necesită trei cadre cheie: unul la început pentru când obiectul este la sol, un al doilea în mijloc pentru când obiectul este la înălțimea maximă a saltului și un al treilea la finalul saltului, când este din nou pe sol. Astfel, Unity determină automat interpolarea să fie *Smooth In* și *Smooth Out* pe fiecare cadru cheie, astfel că mișcarea va fi continuă, naturală.

Pentru a eficientiza producția de animație Unity utilizează așa numitele *controlerele de animație* denumite și animatoare. Acestea sunt *mașini de stare* (state machines) care determină ce animații sunt redată în mod curent și permit sincronizarea și suprapunerea fără probleme a animațiilor pentru diferite obiecte din scena grafică.

Pentru animația eficientă a caracterelor (în general obiecte ființă umană) în Unity folosește conceptul de *avatar umanoid*. Avatarurile sunt definiții pentru modul în care animațiile afectează transformările unui model.

Tot din considerente de productivitate există *modele de animație*. Există două tipuri de animație: *banda de alergat* (treadmill) și *mișcarea rădăcinii* (root motion). Banda de alergare înseamnă că animația rămâne la origine și folosim programul (script) pentru a muta acel obiect. Mișcarea rădăcinii înseamnă că mișcarea este încorporată chiar în animație și aceasta determină cât de departe se mișcă obiectul, mai degrabă decât programul. În Unity 5 este posibil să se genereze date de *mișcare rădăcină* din curbele animației fără ajutorul programului script.

3. Principii și metode în animația realistă

Realizarea scenelor realiste presupune două condiții de bază:

- realizarea unei *topologii* pertinente a obiectelor ce compun fiecare scenă;
- impunerea unei **cinematici realiste**.

Principiile care guvernează mișcarea tuturor elementelor în aplicațiile de animație se bazează pe legile dinamicii clasice. Relația topologie-mișcare în grafica pe calculator presupune elaborarea unor modele complexe în care intervin parametrii de poziție și de mișcare în anumite puncte, precum și diverși parametrii fizici (masa, momente de inerție, coeficienți de frecare, vâscozitate, accelerație gravitațională, etc.) cu rolul de a aduce dinamica virtuală cât mai aproape de realitate. De exemplu, cu ajutorul extensiilor recente ale mediului 3D Studio Max sunt puse la dispoziție instrumente pentru modelarea dinamicii diverselor clase de obiecte și medii fizice, astfel:

- modelarea dinamicii fluidelor (Fluid Dynamics) prin aplicarea unor efecte cum sunt turbulențele de curgere, valuri, interacțiuni dintre fluid și suprafețele obiectelor;
- modelarea firelor (Rope) cum ar fi lanțurile deformabile, corzile, cablurile, etc.;
- modelarea obiectelor de îmbrăcăminte (Cloth Dynamics) ținând cont de caracteristicile de material (elasticitate, greutate, șifonare, densitate);
- modelarea dinamicii mediilor solide rigide (Rigid Body Dynamics) cu aplicabilitate în dinamica prăbușirii zidurilor, alunecări de teren, coliziuni de vehicule, etc..

Alte [efecte de animație avansată](#) sunt:

- efectul de flacără (flame),
- efectul de incendiu (fire),
- efectul de fum (smoke),
- efectul de îngheț sau mătuire a suprafețelor (frost).

În mediile grafice avansate se pot defini [obiecte complexe](#) cu parametrii modificabili:

- o varietate bogată de obiecte dinamice,
- sisteme de particule,
- deformări ale spațiului virtual.

În mediile grafice avansate există un set de instrumente software numite "*reactori*" cu ajutorul cărora utilizatorul poate decide cum interacționează un obiect cu alte obiecte din scenă sau cu mediul în care se desfășoară acțiunea. Astfel, există conceptul de *sistem software pentru animație dotat cu inteligență* în care fiecare obiect al modelului este integrat în mediu, fiind în legătură cu entitățile învecinate. Modificarea unei entități va avea efecte asupra întregii compoziții potrivit complexității relațiilor de legătură predefinite. O metodă de bază prin care se implementează astfel de mecanisme este *sistemul vertex interactiv*, care constituie un sistem de noduri (puncte) interconectate în maniera unui graf la nivelul întregii compoziții grafice.

În principiu animația se poate realiza prin următoarele tehnici:

- generarea de poziții intermediare ale ființelor și obiectelor prin desenarea acestora de către grafician;
- captura (video) a mișcării ființelor și obiectelor prin marcarea cu puncte de referință a elementelor cinematice;
- aplicarea unor modele fizice programate pentru mișcări specifice (mers, alergare, salt, cădere, etc.) pentru ființe, obiecte.

3.1 Metoda cinematicii inverse

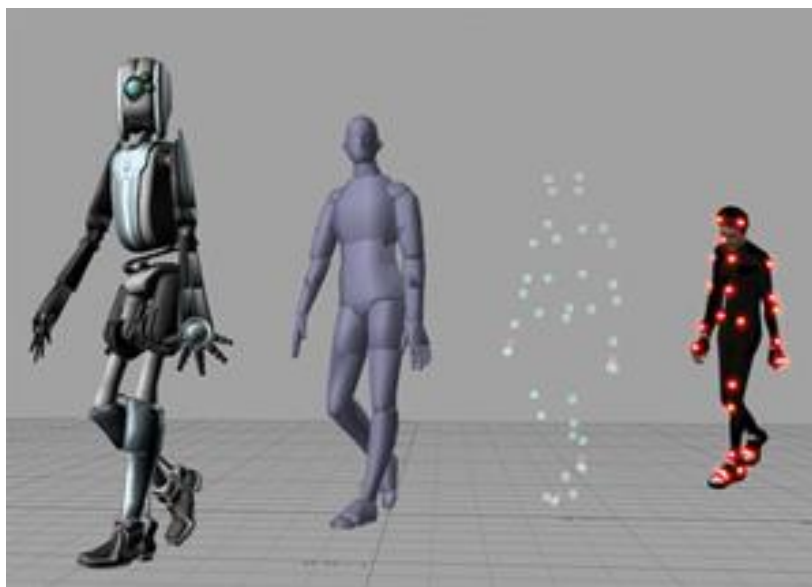
Metoda *cinematicii inverse* (inverse kinematics-IK) este complementară celei de *cinematică directă* (forward kinematics-FK), iar metodologia IK/FK constituie baza realizării de cinematici realiste în aplicațiile de animație.

Metoda cinematicii inverse constă în acționarea unui lanț de obiecte interconectate, prin manipularea unui obiect de control de la extremitatea lanțului respectiv rezultând astfel poziția articulațiilor (legăturilor).

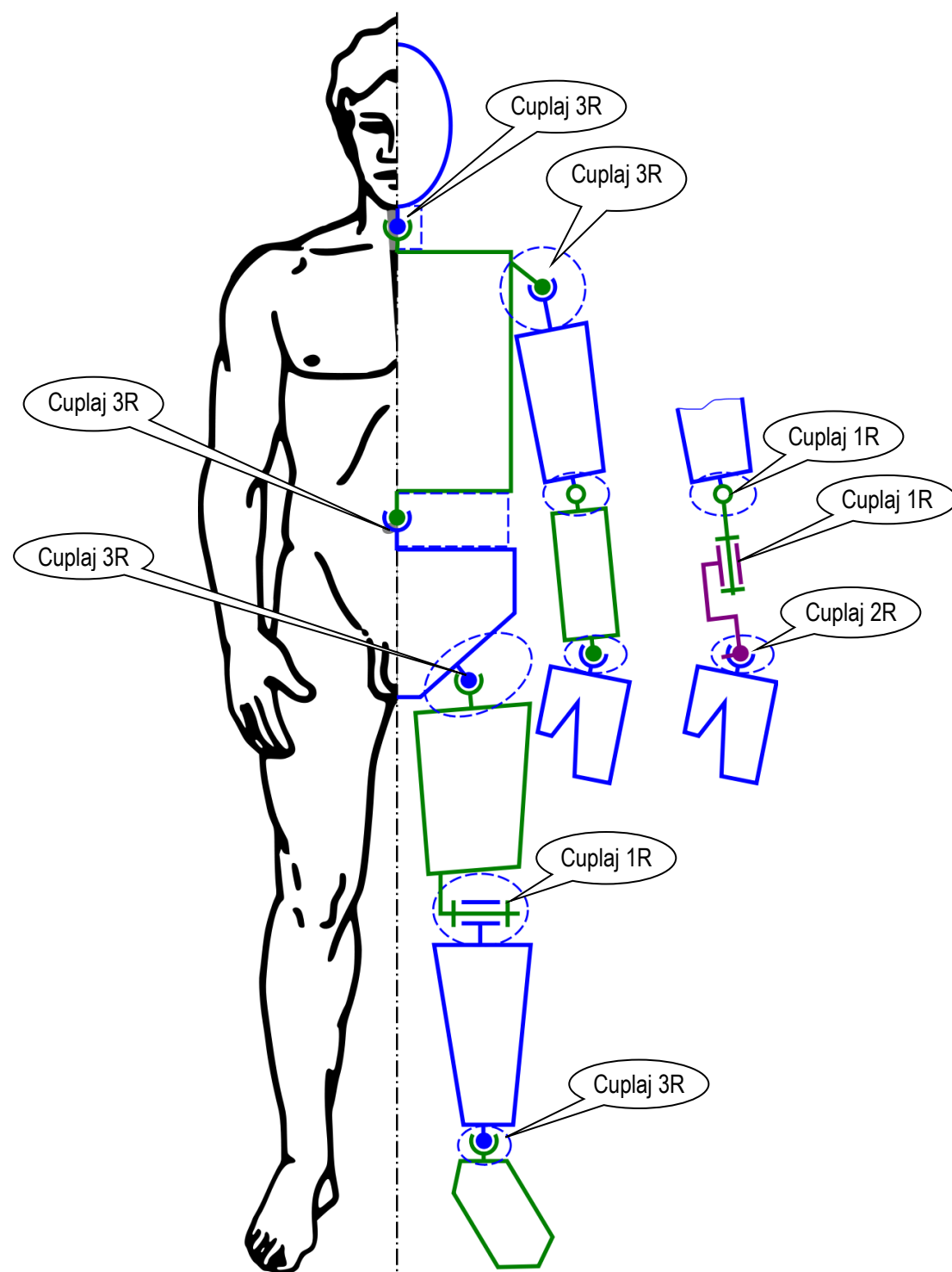
Prin *metoda cinematicii directe*, în mod uzual se verifică rezultatul obținut, prin cunoașterea datelor punctelor de legătură (articulații) și determinarea poziției elementului final.

În general *animația umanizată* se bazează pe structuri de tip schelet (skeleton) format din oase (bones) și articulații. Mișcările normale ale acestor structuri sunt caracterizate de anumite distribuții de viteză pe fiecare element cinematic, impuse de natura legăturilor (cuplajelor) dintre acestea. De exemplu, un picior care merge, un braț cu mână care apucă un obiect, sau un autovehicul care rulează pe o suprafață vor avea cinematica impusă de elementul de execuție (end

effector), astfel: traiectoria tălpii piciorului va defini cinematica mersului, traiectoria mâinii va defini mișcarea întregului braț, iar roata motoare a autovehiculului va determina cinematica acestuia. Astfel, dacă se impune o mișcare convenabilă elementelor finale, care realizează efectiv acțiunea de deplasare a unui obiect în raport cu altul sau cu mediul, prin contactul cu acestea (de ex. centrul instantaneu de rotație), atunci se poate obține recursiv mișcarea corectă a întregului ansamblu.



Exemplu de captare a mișcării pentru generarea animației realiste
(https://en.wikipedia.org/wiki/Computer_animation)





Exemplu de model cinematic al ființei umane pe bază de lanțuri cinematice
(https://en.wikipedia.org/wiki/Inverse_kinematics)

4. Aplicații cu imagini dinamice. Exemple în Matlab

Metodologia de realizare a efectelor de animație pe baza manipulării imaginilor statice poate fi pusă în evidență pe baza unor programe care utilizează funcțiile specializate pentru lucrul cu imagini din mediul Matlab (vezi tabelul următor).

Funcția	Descriere
<code>M=moviein(nr_cadre)</code>	Pregătirea memoriei pentru salvarea unui număr de cadre (imagini) sub forma unei matrici <code>M</code> cu dimensiunea $[1 \times nr_cadre]$ având următoarea structură: <code>cdata</code> - matrice cu dimensiunea $[nr_pixeli_Y \times nr_pixeli_X \times 3_RGB]$ ale cărei elemente indexează datele de culoare ale fiecărui element al imaginii din matricea de culoare <code>colormap</code> , <code>colormap</code> - matrice de culoare asociată reprezentării grafice, cu dimensiunea $[m \times 3]$ ale cărei elemente sunt numere reale în intervalul $[0,1]$. Implicit, <code>m=64</code> și definește scala culorilor (numărul de nuanțe disponibile).
<code>M(k)=getframe</code>	Funcție folosită pentru a asambla o matrice din <code>k</code> imagini (cadre) existente.
<code>movie(h,M,nr_cadre,f,loc)</code>	Lansează operația de redare a cadrelor unei aplicații de tip movie. Argumentele au următoarea semnificație: <code>h</code> - identificatorul obiectului grafic figură (este opțional), <code>M</code> - matricea cu succesiunea de cadre, <code>nr_cadre</code> - numărul de cadre care se vor reda (vizualizează), <code>f</code> - frecvența de succedare a cadrelor (în fps), <code>loc</code> - precizarea poziției <code>[X,Y]</code> de afișare în fereastra grafică.
<code>imshow</code>	Utilizată cu sintaxa <code>imshow(I,N)</code> funcția afișează o imagine în nuanțe de gri folosind <code>N</code> nivele de intensitate. Dacă <code>N</code> nu se specifică atunci se utilizează implicit 256 nivele de gri.

NOTA: A se vedea Toolboxul Matlab _Image Processing

<pre>%Exemplul1 %Rotirea unei imagini cu 20 grade clear %image(A) A=imread('C:\Matlabr11\cadru11','bmp'); %converteste in AN I=RGB2GRAY(A); %redimensioneaza imaginea I=IMRESIZE(I,0.5,'nearest'); imshow(I); %aloca matricea de memorie M=moviein(20); %introduce efectul dinamic/animatia for j=1:20 J=imrotate(I,-j,'bilinear','crop'); %brighten(1/j) %memoreaza cadrele succesive M(:,j)=getframe; imshow(J); end %redarea cadrelor figure(2) h=gcf movie(h,M,20,24)</pre>	
<pre>%Exemplul2 %animatie pe baza a doua cadre clear %image(A) [A,map]=imread('C:\Matlabr11\cadru11','bmp'); [B,map]=imread('C:\Matlabr11\cadru12','bmp'); %converteste in AN (optional) I=RGB2GRAY(A); J=RGB2GRAY(B); %redimensioneaza imaginea (recomandat) I=IMRESIZE(A,0.75,'nearest'); J=IMRESIZE(B,0.75,'nearest'); %aloca matricea de memorie M</pre>	

```

M=moviein(20)
%introduce efectul dinamic/animatia
for k=1:2
    imshow(I);
    I=J;
    %memoreaza cadrele succesive in matricea M
    M(k)=getframe;
end
close
%redarea cadrelor
figure(2)
h=gcf
movie(h,M,20,8)

```

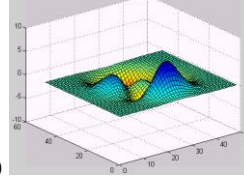


```

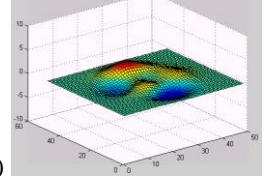
%Exemplul3
%animatia unei figuri predefinite
clear
z=peaks
surf(z)
lim=axis
M=moviein(20);
for j=1:20
    surf(sin(2*pi*j/20)*z,z)
    axis(lim)
    M(j)=getframe
end
movie(M,20,12)
end;

```

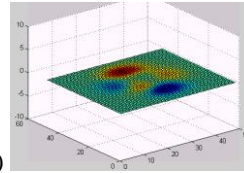
(1)



(3)



(2)



(4)

