CONTRIBUŢII LA UTILIZAREA REALITĂŢII VIRTUALE ÎN PROIECTAREA ASISTATĂ DE CALCULATOR

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul ŞTIINŢA CALCULATOARELOR
de către

Ing. Daniel Cioi

Conducător ştiinţific: Prof.dr.ing. Savii George
Referenţi ştiinţifici: Prof.dr.ing. Burdescu Dumitru Dan
Prof.dr.ing. Talabă Doru
Prof.dr.ing. Creţu Vladimir

Ziua susţinerii tezei: 14.03.2008

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor

Universitatea „Politehnica" din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Prezenta lucrare este rezultatul activităţii prestate în perioada stagiului ca doctorand cu frecvenţă la Facultatea de Mecanică, Departamentul de Mecatronică, Universitatea "Politehnica" din Timişoara.

Pentru început, doresc să mulţumesc coordonatorului Prof.dr.ing. Savii George, pentru sfaturile şi suportul acordat în toate problemele apărute pe parcursul elaborării tezei.

Îmi exprim întreaga consideraţie faţă de membrii comisiei de doctorat, domnul preşedinte al comisiei Prof.dr.ing. Bereteu Liviu, decanul Facultăţii de Mecanică din Timişoara şi domnii Prof.dr.ing. Burdescu Dumitru Dan de la Universitatea din Craiova, Prof.dr.ing. Talabă Doru de la Universitatea Transilvania din Braşov şi Prof.dr.ing. Creţu Vladimir de la Universitatea "Politehnica" din Timişoara, care au răspuns solicitării de a face parte din comisia de analiză a tezei, pentru observaţiile făcute şi pentru timpul acordat lucrării.

De asemenea mulţumesc tuturor ce m-au ajutat cu diverse sfaturi şi indicaţii utile finalizării tezei.

Această teză a fost redactată în LaTEX utilizând distribuţia "free" MiKTeX pentru Windows.

Timişoara, martie 2008 Cioi Daniel

Cioi, Daniel

Contribuţii la utilizarea Realităţii Virtuale în Proiectarea Asistată de Calculator

Teze de doctorat ale UPT, Seria 10, Nr. 3, Editura Politehnica, 2008, 182 pagini, 80 figuri, 6 tabele.

ISSN: 1842-7707

ISBN: 978-973-625-613-4

Cuvinte cheie:
Realitate Virtuală, Proiectare Asistată de Calculator, VRforCAD, SphereDevice

Rezumat:
Puse împreună Realitatea Virtuală (VR) şi Proiectarea Asistată de Calculator (CAD) înzestrează proiectantul cu o mai bună comunicare cu calculatorul. Ceea ce înseamnă că nu mai este necesar să condensăm toate informaţiile pe un ecran 2D în faţa utilizatorului, acest tip de afişare a informaţiei ducând la posibilitatea apariţiei de ambiguităţi.
Această teză de doctorat prezintă diverse tehnici de utilizare a Realităţii Virtuale în scopul creşterii interactivităţii şi a funcţiilor de vizualizare în sistemele CAD. În acest scop, autorul a dezvoltat o aplicaţie intitulată "VRforCAD", aplicaţie ce este prezentată mai amănunţit în conţinutul tezei. Aplicaţia dezvoltată în Java şi Java3D se încadrează în domeniul softuri de realitate virtuală. De asemenea este prezentat un echipament cu feedback haptic realizat de autor în contextul manipulării modelelor CAD, echipament ce face parte din domeniul hardware a echipamentelor de realitate virtuală.
Din punct de vedere financiar, la realizarea acestei teze, gratul de tip TD CNCSIS nr. 87, derulat pe parcursul a 2 ani 2006-2007 la care autorul a fost director, a adus o mare contribuţie calităţii rezultatelor.
O parte a rezultatelor tezei sunt prezentate şi pe web la adresa: http://www.vrforcad.org

Cuprins

Prefață

Pentru început, doresc să mulțumesc coordonatorului Prof.dr.ing. Savii George, pentru sfaturile și suportul acordat în toate problemele apărute pe parcursul elaborării tezei.

De asemenea mulțumesc tuturor ce m-au ajutat cu diverse sfaturi și indicații utile finalizării tezei.

Această teză a fost redactată în LaTEX utilizând distribuția "free" MiKTeX pentru Windows.

Capitolul 1

Introducere

Prezenta teză își propune să reliefeze contribuțiile proprii, rezultate din activitatea de cercetare pe parcursul a 4 ani de studiu în cadrul programului de doctorat cu frecvență cu tema "Contribuții la utilizarea Realității Virtuale în Proiectarea Asistată de Calculator".

Pentru a sistematiza volumul de informație cuprins în cadrul tezei, prezenta lucrare este structurată pe șase capitole.

Capitolul 1, conține definiții și noțiuni introductive a ceea ce înseamnă Realitate Virtulală (VR) și Proiectare Asistată de Calculator (CAD). De asemenea se evidențiază beneficiile utilizării realității virtuale împreună cu sistemele de proiectare asistată de calculator.

În continuare se prezintă stadiul actual al sistemelor VR - CAD. Se face o scurtă introducere a conceptului VR - CAD, după care, mai multe exemple introduc realizările integrării softurilor VR și CAD la diferite nivele, prin cuplarea online completă a aplicațiilor sau prin integrarea funcționalității nucleului CAD în sistemele VR.

La sfârșit de capitol, se prezintă obiectivele prezentei tezei. Astfel, în conformitate cu obiectivele formulate, în următoarele capitole sunt prezentate contribuțiile teoretice și practice ce au dus la îndeplinirea acestor obiective.

Capitolul 2, "Limbaje de Realitate Virtuală" prezintă o o scurtă introducere a limbajelor de programare în realitate virtuală, câteva exemple realizate de autor pe parcursul stadiului de pregătire al tezei (publicate la conferințe în domeniu) și se încheie

prin justificarea motivelor pentru care s-a ales limbajul Java și API-ul Java3D pentru dezvoltarea aplicației realizată conform obiectivelor prezentate în capitolul 1.

Capitolul 3, "Contribuții la realizarea unui sistem VR - CAD" este menit să justifice necesitatea unui astfel de sistem. Se evidențiază probleme ce apar la interschimbabilitatea modelelor 3D între sistemele VR și sistemele CAD. În continuare se prezintă diverse formate de export a modelelor 3D. Capitolul se încheie prin alegerea a două formate de export modele 3D, formate ce sunt apoi folosite ca bază pentru dezvoltarea aplicației prezentată în capitolul următor.

Capitolul 4, "Contribuții la dezvoltarea și testarea unui software pentru sistemul VR-CAD" prezintă detaliat aplicația intitulată "VRforCAD" dezvoltată în limbajul Java și API-ul Java3D de către autorul prezentei teze, reprezentând componenta software a sistemului VR - CAD.

Capitolul 5, "Contribuții la proiectarea, realizarea și testarea unui echipament hardware pentru sistemul VR-CAD" prezintă detaliat echipamentul "SphereDevice" proiectat și realizat fizic de către autorul prezentei teze, reprezentând componenta hardware a sistemului VR - CAD. Sunt prezentate problemele și soluțiile ce rezolvă aceste probleme, apărute pe parcursul realizării echipamentului.

Capitolul 6, "Concluzii finale", conține o sinteză a concluziilor și contribuțiilor desprinse din prezenta lucrare, analizate din diverse puncte de vedere. De asemenea în subcapitolul: "Stadiul realizării obiectivelor tezei", se prezintă punctual gradul de realizarea a obiectivelor prezentate în capitolul 1 prin soluțiile prezentate pe parcursul tezei, cât și problemele întâmpinate pe parcursul atingerii fiecărui obiectiv formulat.

1.1 Ce este realitatea virtuală?

- "Realitatea Virtuală reprezintă o simulare generată de calculator a unui mediu tridimensional în care utilizatorul este capabil să vizualizeze și să manipuleze conținutul acestui mediu. Realitatea Virtuală este de obicei manipulată și explorată utilizând diverse echipamente de intrare cum ar fi: ochelari, cască audio, mănuși și/sau computer. Utilizând aceste echipamente, un user poate naviga prin lumea virtuală și manipula obiectele virtuale" [21].

- "Realitatea Virtuală este un sistem folosit pentru a crea o lume artificială pentru un utilizator astfel încât acesta să aibă impresia că se află în această realitate în care se poate mișca și interacționa cu obiectele înconjurătoare" [60].

- "Grafica interactivă în timp real cu modele 3D combinată cu o tehnologie de afișare care oferă utilizatorului imersiunea în modelul lumii și posibilitatea manipulării directe a acestuia" [38].

- "Iluzia participării într-un mediu sintetic în locul observării externe a acestui mediu. RV se bazează pe display-uri 3D stereoscopice purtate de utilizator, urmărirea mâinilor/corpului și un sunet binaural. RV este o experiență imersivă, multi senzorială" [40].

- "Simulări pe calculator care utilizează o grafică 3D și astfel de dispozitive, cum sunt DataGlove, pentru a permite utilizatorului să interacționeze cu simularea" [51].

- "Realitatea Virtuală se referă la medii imersive, interactive, multi-senzoriale, centrate spre utilizator, tridimensionale, generate de calculator și combinarea tehnologiilor necesare construirii acestor medii" [23].

- "Realitatea Virtuală ne permite să navigăm și să vedem o lume în trei dimensiuni în timp real, cu șase grade de libertate, fiind, în esență, o clonă (virtuală) a realității fizice" [92].

1.2 Ce este CAD?

CAD este acronimul de la termenul englez Computer Aided Design - în traducere, Proiectare Asistată de Calculator - și grupează tehnicile computerizate utilizate în proiectarea și modelarea produselor. Informațiile din aceste modele constituie baza pentru analiza proiectului, proiectarea tehnologiei de execuție și planificarea producției.

Există multe proprietăți care trebuie incluse în model, dintre care cele mai importante sunt forma, dimensiunile, toleranțele de execuție și structura. Pentru aceste aspecte sunt necesare reprezentări geometrice plane și spațiale, precum și modelări matematice ale caracteristicilor structurale ale produsului. Utilizarea calculatorului necesită, pe de

altă parte, tehnici de creare și manipulare a imaginilor pe ecran. Din acest motiv, principiile CAD-ului au la bază geometria plană și spațială, pe de o parte, și grafica asistată de calculator, bazele de date și cunoștințe, pe de altă parte. Cu aceste instrumente, în ultimii 20 de ani, au fost concepute și lansate pe piață pachete software CAD extrem de puternice, care automatizează într-un grad foarte înalt munca de modelare și proiectare inginerească. Ele înglobează un număr de funcții care crește continuu, exprimate în forme tot mai apropiate comunicării umane, menite să scurteze timpul de proiectare și astfel de lansare pe piață a produsului [85].

1.3 Realitate Virtuală și CAD

Diferența dintre programele de Realitate Virtuală (VR) și programele/aplicațiile de proiectare asistată de calculator (CAD) sunt deseori greșit înțelese. Majoritatea oamenilor asociază Realitatea Virtuală cu jocurile pe calculator, în timp ce CAD este asociat cu design-ul arhitectural și ingineresc. Realitatea Virtuală, deseori se referă la modele ce sunt prezentate într-un mod mai realistic, în timp ce CAD prezintă modele cu o mai mare acuratețe și precizie.

De fapt, CAD și Realitatea Virtuală sunt tehnologii complementare. Modelele VR pot fi realistice, dar de asemenea pot fi și modele CAD. Modelele CAD pot fi de o mare acurtatețe și precizie, dar de asemenea pot fi și modele VR. Distincția între cele două tehnologii este mai puțin accentuată de tipul modelelor utilizate și mai mult accentuată de cele mai bune caracteristici ale programelor utilizate.

CAD a fost dezvoltat pentru a crea modele de precizie matematică, în trei dimensiuni a obiectelor fizice din lumea reală. Scopul acestor aplicații este acela de a crea modele.

Sistemele de realitate virtuală sunt mai eficiente în ceea ce privește asigurarea unor posibile experiențe ce imersează utilizatorul în lumea virtuală în timp ce privește o imagine generată de calculator, chiar dacă imaginea este afișată pe un ecran mare sau vizualizată cu ajutorul unui HMD (*head-mounted display*) oferind o imagine binoculară. Aceste sisteme VR pot fi utilizate pentru a crea modele cu abilități de a le afișa și a permite mișcări în timp real în lumile virtuale.

Realitatea Virtuală, tinde să însemne abilitatea "walk around" a modelelor mate-

matice 3D și vizualizarea acestora ca și cum ar exista fizic în spațiu.

Programele CAD au fost create pentru a construi modele. Programele VR sunt calificate în afișarea modelelor. Chiar dacă fiecare a preluat unele din capabilitățile celeilalte nu înseamnă că și-au replicat toate funcțiile.

Această distincție între CAD și VR este importantă, deoarece multe persoane se gândesc serios la potențialul Realității Virtuale în afișarea monumentelor istorice. În timp ce sistemele VR sunt utilizate pentru a crea modele bune, sistemele CAD sunt ușor de utilizat pentru modele precise ceea ce este mult mai eficient.

De asemenea, este adevărat că sistemele CAD deși pot produce redări excelente, nu pot imersa un utilizator în lumea virtuală, nu pot prezenta modele ca și cum ar fi tangibile.

Ideal ar fi ca programele CAD să fie utilizate la crearea modelelor, iar programele VR să fie utilizate la afișarea acestora. În acest fel ambele sisteme pot aduce un aport semnificativ la crearea lumilor virtuale de o mai mare precizie și acuratețe.

Există însă un important impediment, acela că programele CAD și cele VR utilizează formate de date diferite. În timp ce unele formate CAD pot fi convertite în formate VR și pot fi afișate în medii virtuale, procedeul de convertire inversă nu este posibil.

Mediile virtuale (VE) sunt experiențe senzoriale care comunică unui operator uman componente fizice și abstracte și care asigură un mediu vizual portretistic ce poate da informații unui operator oriunde se uită acesta. Acest lucru înseamnă că nu mai este nevoie să se condenseze toată informația în ecranul bidimensional care se află în fața operatorului, în acest caz putând apărea ambiguități.

În ceea ce privește prezentarea și interacțiunea cu informația afișată, un mediu virtual asigură în mod sigur o mai bună cale de a comunica cu un calculator.

Există două ipoteze principale ce pot fi formulate în ceea ce privește avantajele pe care le poate aduce realitatea virtuală proiectării asistate de calculator:

- îmbunătățește vizualizarea superioară a produsului prin permiterea utilizatorului să coexiste în același spațiu cu modelul produsului și astfel prin câștigarea unei aprecieri mai bune la adresa formei și esteticii produsului;

- îmbunătățește interacțiunea cu design-ul în sensul unor manipulări intuitive mai ridicate a modelului și al unei experimentări fucționale, proiectantul putând inte-

racționa efectiv cu modelul produsului în schimbul folosirii tradiționalului mouse și al cursorului bidimensional.

Realitatea virtuală și instrumentele ei asigură utilizatorilor interfețe periferice puternice când aceștia interacționează cu modelele 3D. Cu ajutorul acestor interfețe (haptic devices) inginerii pot atinge un model, să simtă și să modifice modelul împingând sau trăgând suprafețele într-un mediu natural tridimensional. Astfel în câteva secunde se pot genera noi modele CAD ce sunt dificil de făcut cu interfețele tradiționale 2D.

De exemplu, pentru a crea și modifica o suprafață, proiectanții trebuie să se confrunte cu suprafețe de construcție curbe. Forma suprafeței este dificil de controlat, deoarece depinde de numărul, forma și poziția zonei curbate. Proiectarea interactivă a suprafețelor este de obicei un proces greoi, ineficient și repetabil.

De multe ori s-a întâmplat ca inginerii sau designerii să vrea să atingă un model pentru a putea modifica suprafața acestuia prin tragere sau împingere. Desigur acest lucru nu este posibil cu instrumentele interactive bidimensionale tradiționale, dar este posibil cu realitate virtuală.

Din punct de vedere al terminologiei se remarcă doi termeni care descriu două senzații diferite care intervin la atingerea obiectelor:

- reacția tactilă (*touch feedback*) - este senzația recepționată de piele atunci când este atinsă mecanic, termic, chimic sau electric. Senzorii tactili sunt plasați la periferia pielii, cea mai mare densitate a acestora găsindu-se în mână și furnizează informații despre rugozitatea suprafețelor sau temperatură. În realitatea virtuală se simulează reacția tactilă prin acționarea cu o forță repartizată spațial asupra extremității degetelor.

- reacția de forță (*force feedback*) - este răspunsul unui obiect la o acțiune externă, de exemplu rezistența la apăsare sau greutatea pe care o prezintă dacă este ridicat. Forța de reacție este recepționată de senzori plasați în interiorul corpului, în general în tendoanele musculare. În realitatea virtuală, forța de reacție se generează prin dispozitive care produc o forță de apăsare asupra organismului (în general asupra degetelor).

Progresele recente în dispozitive haptice au promovat folosirea reacției în aplicații de realitate virtuală.

Cuvântul "haptic" se referă la procedee de simt al atingerii [86].

Interfața haptică este un dispozitiv ce permite utilizatorului să interacționeze cu un calculator receptând reacție tactilă. Acestă reactie este realizată aplicând utilizatorului o forță reactivă.

Există două tipuri principale de dispozitive haptice:

- mănuși sau stilou creion, dispozitive ce permit utilizatorului să atingă și să manipuleze obiecte virtuale 3D;
- dispozitive ce permit utilizatorului să simtă texturi ale obiectelor 2D cu ajutorul unui stilou creion sau mouse.

Volkov si Vance [91] au făcut un studiu pentru a evalua cum un dispozitiv haptic afectează abilitatea unei persoane de a proiecta un anumit obiect. Rezultatele indică faptul că adăugarea reacției (*feedback*) permite utilizatorului să realizeze o instructiune într-un timp mult mai scurt și deseori cu o mai mare precizie.

Pentru crearea unui model în mediul virtual pot fi folosite diferite tipuri de interfețe om - calculator. Observația că oamenii distribuie sarciniile între mâinile lor a dus la dezvoltarea de interfețe cu două mâini bazate pe dispozitive de intrare, ca de exemplu mănușile și pointeri în combinație cu *head tracker*. În general, utilizatorul poartă două mănuși și are acces la un instrument tip stilou (*stylus device*), care poate fi folosit ca și instrument adițional.

În Figura 1.1 este prezentat un exemplu de sistem special [36] proiectat pentru lucrul cu noua generație de sisteme de proiecție stereo. Utilizatorul poartă ochelari cu un senzor de poziție (*head tracking*) pentru vederea stereoscopică și folosește pentru manipulare un set de mănuși combinate cu instrument tip stilou pentru interactiunea cu mediul virtual. Informația spațială descrisă de poziția capului utilizatorului și mișcarea mâinii, este complet integrată în mediul virtual. O scurtă descriere a dispozitivelor de intrare este făcută în cele ce urmează:

- stiloul: folosind un emițător fix ca referință, acest sistem calculează cu precizie poziția (coordonatele x, y, z) și orientarea (*yaw, pitch, roll*) obținute cu ajutorul

unui mic receptor conținut în stilou. În plus, este înzestrat cu un buton care poate fi folosit pentru acțiuni de selectare:

- mănușa: sistemul de manipulare foloseste mănuși de stofa cu senzori electrici în fiecare deget. Contactul între oricare două sau mai multe degete termină o traiectorie conducătoare, asigurând o varietate de gesturi cu degetele, care pot fi asociate cu diverse acțiuni. În plus, prin atașarea unui *tracker* electromagnetic se captează poziția fiecărei mănuși.

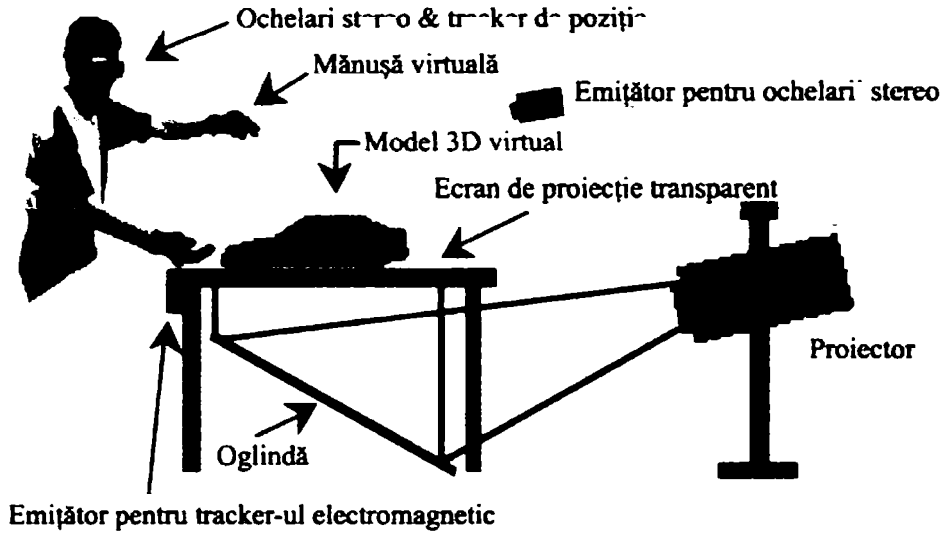


Figura 1.1: Exemplu de sistem cu proiecție stereo

Proiectarea orientată spre obiecte tratează fiecare componentă vizuală în interiorul mediului virtual ca un obiect ce poate fi vizualizat, analizat, verificat, poziționat si manipulat. Un comportament special de acțiune poate fi atașat oricărui obiect, acel obiect fiind transformat în instrument pentru manipularea altor obiecte. Orice obiect normal non-static din interiorul scenei grafice poate fi direct accesat, manipulat și grupat. Obiectele non-statice vin fără constrângeri, pe când obiectele statice vin cu un set distinct de atribute definite care permit manipularea si interactiunea în interiorul unor limite precise. Aceste obiecte sunt în particular folosite pentru proiectarea industrială deoarece ele păstrează constrângerile predefinite prin proiect. Odată ce un obiect este

creat, reprezentarea vizuală este adăugată ierarhic la scena grafică. Toate obiectele vizibile conținute în scena grafică pot fi selectate și manipulate folosind gesturi simple ale mâinii. Mediul este înzestrat cu un set de control de bază, ce include:

- creare de obiecte,
- selectare de obiecte,
- manipulare de obiecte,
- verificare de obiecte,
- analiză de obiecte.

O condiție necesară este aceea ca istoria obiectului proiectat să fie accesibilă. Acest mod de abordare permite utilizatorului să valideze caracteristici importante incluzând operațiile *undo* și *redo*, și de asemenea, copierea eficientă artistică sau inginerească a modelului proiectat.

Folosind *head tracking-ul*, proiectantul poate studia întregul model în mediul virtual prin simpla mișcare a capului sau plimbarea fizică în jurul modelului. Modul de navigare este împărțit în două direcții. Prin execuția unei acțiuni de prindere, utilizatorul poate selecta și repoziționa un obiect prin folosirea uneia dintre mânuși. Acest mod, numit mod de navigație al obiectului, permite utilizatorului să repoziționeze și să analizeze liber un obiect activ. Dacă ambele mânuși execută o acțiune de prindere în același timp, sistemul comută în modul de navigație în scenă. Segmentul imaginar dintre punctele de prindere este folosit ca și un manipulator cu cinci degete. Lungimea acestui segment, adică distanța dintre cele două puncte de prindere, controlează scara modelului. În cazul în care nici un obiect nu este selectat, acțiunea de navigare este aplicată întregii scene, care prin definiție, este tocmai un obiect ce face parte dintr-un grup de obiecte. Această schemă suportă un nivel definit de utilizator al preciziei, în care nivelul fin sau grosolan al preciziei poate fi definit prin scalarea spațiului de lucru. De asemenea, schema de navigație este intuitivă și universală și noii utilizatori au posibilitatea să examineze ușor chiar și scene complexe după numai câteva minute de practică.

Meniurile virtuale sunt componente vitale în toate sistemele de modelare deoarece ele în general asigură accesul intuitiv la sistemele aflate în funcționare. Cu tranziția

din mediul 2D în mediul 3D, a fost necesară implementarea unui nou set de dispozitive de intrare pentru realitatea virtuală și în consecință noi concepte.

De obicei apar probleme la interfața dintre meniul 3D și scenă și extinderea opțiunilor submeniurilor foarte multe (meniuri cascadă). Se face distincție între gesturi de bază, de acțiune și evenimente invocate deoarece utilizatorul poate activa și selecta din meniuri diferite. Pentru utilizatori începători, opțiunile *watch-meniu* și *communicator-badge* sunt cele preferate, deoarece în aceste moduri meniurile de bază pot fi vizualizate ca și cum acestia s-ar uita la ceas. Meniul este compus din butoane 3D asamblate pe palete rectangulare (Figura 1.2) și este atașat la o mănușă de urmărire. Toate submeniurile apar de pe paleta principală în diferite planuri și pot fi accesate sau închise cu cealaltă mână.

Toate meniurile sunt implementate ca și obiecte statice, deoarece pot fi translatate, rotite și scalate după dorință. Deasemenea, meniul poate fi aplicat ca și funcționalitate asociată la alte obiecte când sunt activate și pot fi reprezentate prin orice text, prezentare grafică de funcții asociate sau o combinație a acestora (Figura 1.2).

Figura 1.2: Meniuri virtuale

Crearea obiectelor se realizează interactiv prin folosirea primitivelor de desenare, a formelor libere, *octree* sau a unui set de primitive CAD. Primitivele de desenare includ linii, forme poligonale de bază, seturi de puncte triunghiulare și suport convențional pentru segment de dreaptă și poligon. Primitivele CAD sunt reprezentate în format Bézier, B-spline sau NURBS.

1.4 Stadiul actual al sistemelor VR - CAD

Cercetătorii au implementat trei căi prototip de modelare imersivă:

- conectând un sistem VR și un nucleu CAD;
- conectând un sistem VR și un sistem CAD;
- utilizarea modelelor bazate pe voxel pentru descrierea geometriei.

Exemple de conexiune VR și un nucleu CAD sunt: ARCADE (Advanced Realism CAD Environment) [82], Construct3D [44] și NAVIMODE [68]. Aceste sisteme sunt bazate pe sisteme VR și utilizează nucleul de modelare ACIS pentru descrierea geometrică. ACIS este un nucleu de modelare 3D proprietate a corporației "Spatial Corporation", fiind utilizat la dezvoltarea aplicațiilor cu caracterstici hibride de modelare.

ARCADE este un sistem comun de manipulare directă a modelelor 3D care se bazează pe o interfață prietenoasă cu utilizatorul.

Contruct3D utilizează realitatea augumentată ca interfață cu utilizatorul și permite funcții pentru crearea volumelor primitive și modificarea acestora utilizând operații de tip Boolean.

Sistemul SpaceDesign [35] este un exemplu de sistem de modelare imersivă a suprafețelor. SpaceDesign permite crearea și modificarea curbelor și a suprafețelor 3D într-un mediu VR sau AR.

Sistemul VADE (Virtual Assembly Design Environment) [52] a fost primul sistem care a conectat complet un sistem CAD și un mediu VR. Mediul VR utilizat a fost un HMD (Head Mounted Display). Acest sistem a fost dezvoltat pentru simularea ansamblării și planificarea unor strategii alternative de ansamblare a componentelor.

VCM (Virtual Clay Modeling) [55] este un exemplu de modelare bazată pe voxel. Componenta principală hardware a acestui sistem este un echipament de intrare cu un *tracker*

de poziție 3D și un senzor tactil. Mutarea obiectului virtual este în corespondență directă cu mutarea echipamentului de intrare și suprafețele sunt deformate când utilizatorul apasă pe senzorul tactil.

NAVIMODE este o interfață care conectează sistemele CAD și sistemele VR. Un nou echipament interactiv este configurat utilizând o tabletă PC portabilă. Aditional funcțiilor uzuale ale sistemelor CAD, utilizatorul poate găsi propria cale vizuală în proiectarea *draft*. O funcție specială, "Virtual Loupe" este creată pentru acest scop. Ecranul de vizualizare al modelului virtual este în acord cu poziția și orientarea modelului, utilizatorul poate astfel privi la lumea virtuală ca și cum ar privi printr-o fereastră (Figura 1.3).

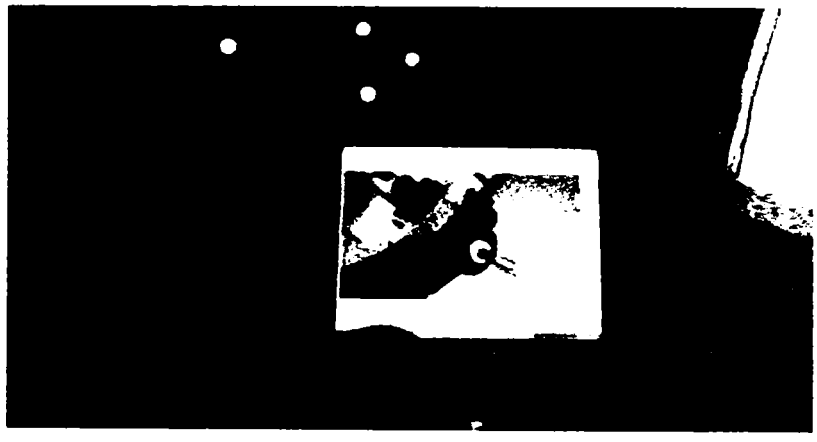


Figura 1.3: Virtual Loupe.

O adevărată provocare este aceea de a asocia parametrii sistemului VR (poziția camerei, unghiul diafragmei) la parametrii sistemului CAD (factorul de mărire, translatare și orientare) în timp real. Pentru acest scop a fost implementată metoda "ChangeView".

Construct|Tool [35] este un sistem de modelare și o unealtă de verificare. A fost dezvoltat în scopul implementării unui mediu imersiv de modelare care funcționează cu geometri exacte.

Aditional funcțiilor de modelare cum ar fi: substract, generarea suprafețelor, creare de primitive, sunt disponibile și funcții de analiză cinematică cum ar fi detectarea coliziunilor în timp real.

Construct|Tool stă la baza sistemelor AR VR "StudierStube" [74] și Construct3D

framework [44]. Funcțiile de detectare a coliziunilor sunt bazate pe metodele API-ului ACIS ceea ce asigură o apreciere foarte precisă la penetrarea geometriei (util de exemplu la calculul toleranțelor).

Interfața cu utilizatorul constă din PIP (personal interaction panel) și un *pen* (stilou). Stiloul conține două butoane care permit acces rapid la cele mai utilizate funcții. O altă functie este opțiunea de transformare a obiectelor în linie dreaptă și discontinuu. Geometriile din scena virtuală pot fi mutate doar într-o direcție particulară. Utilizatorul poate activa cu stiloul una sau mai multe constrângeri și numai deplasarea pe axa corespunzătoare va fi transmisă la obiect. În timpul deplasării sunt disponibile funcții de detectare în timp real a coliziunilor (Figura 1.4). Avertizarea coliziunilor este realizată acustic și vizual. Adițional, obiectele pot fi constrânse să nu penetreze alte geometri în timpul transformărilor interactive de poziție.

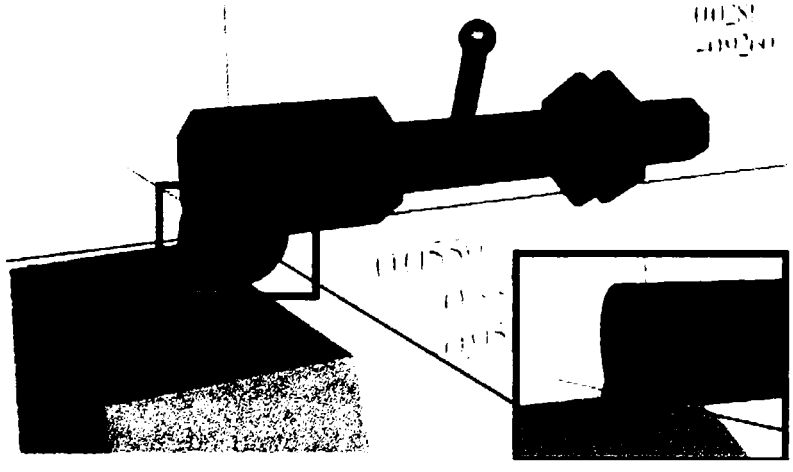


Figura 1.4: Detectarea coliziunilor bazată pe poligoane (stânga) și bazată pe model B-Rep (dreapta).

Construct|Tool asigură descriere geometrică disponibilă imediat ca model B-Rep. De asemenea asigură un mediu precis pentru testare și simulare cinematică.

În continuare sunt prezentate pe scurt câteva din realizările unor cercetători pionieri în domeniul VR-CAD.

Sachs [71] a dezvoltat o unealtă pentru modelarea *free form* a suprafețelor utilizând o pereche de echipamente *Tracker* 3D. Unealta software, permite utilizatorului să folosească un senzor în mâna stângă pentru operații de transformare a poziției și al doilea

senzor, în mâna dreapta, pentru desenarea și editarea suprafețelor 3D.

În [57], Liang descrie un sistem de modelare interactivă 3D care permite utilizatorului manipularea directă a obiectelor în spațiu 3D cu o baghetă ce asigură șase grade de libertate și o serie de manipulatoare 3D intuitive.

Dani și Gadh [25, 26], descriu o platformă pentru modelare în mediu VR și au dezvoltat "Conceptual Virtual Design System", COVIRDS. Platforma constă din trei componente: reprezentarea grafului formei, algoritmi fundamentali pentru redare și modelare și unelte de interacțiune. COVIRDS înzestrează utilizatorul cu comenzi vocale și o interfață de intrare, facilitând crearea și vizualizarea modelului.

În [82], Stork și Maidhof prezintă o metodă de modelare precisă și eficientă utilizând un echipament de intrare 3D.

În [77], este prezentată o metodă de modelare solidă într-un mediu virtual semiimersiv cu scopul de a înzestra utilizatorul cu metode intuitive de creare, modificare, manipulare și vizualizare a modelelor solide prin manipulare directă și comenzi vocale.

1.5 Obiectivele tezei

Plecând de la impoteza că Realitatea Virtuală va include CAD, prezenta teză de doctorat are ca obiectiv central conceperea, realizarea și implementarea unui nou sistem VR-CAD alcătuit dintr-o aplicație *Open Source* cu scopul realizării unui *bridge* între sistemele CAD și sistemele VR, și un echipament hardware original în scopul manipulării în mediu virtual a modelelor CAD.

Ținând seama de limitările sistemelor VR-CAD actuale, lucrarea își propune următoarele obiective:

- Realizarea unui studiu al interacțiunii cu sistemele CAD și definirea unui model cognitiv al interacțiunii CAD.
- Realizarea unui studiu cu privire la posibilitățiile de implementare software a echipamentelor de realitate virtuală pentru vizualizare și manipulare disponibile.
- Realizarea unui studiu al modalității de comunicare între sistemele CAD și sistemele VR, în contextul schimbului de fișiere.

- Realizarea unui studiu al formatelor CAD disponibile în contextul stabilirii cel puțin a unui format comun celor mai multe aplicații CAD.

- Concepția unei arhitecturi *software* a interfeței care combină echipamentele de realitate virtuală și modulele *software* corespunzătoare modalităților de interacțiune considerate.

- Proiectarea și realizarea unei aplicații *software* care trebuie să cuprindă următoarele module:

  - modul de implementare echipamente de realitate virtuală specifice vizualizării stereoscopice;
  - modul de import / export modele CAD care va implementa funcții specifice operării cu formatul de fișier stabilit în scopul interschimbabilității modelelor CAD.
  - modul de implementare a unor funcții specifice operării cu modele 3D.
  - modul de implementare echipamente de realitate virtuală specifice manipulării.

- Aplicația trebuie să fie de tip *cross platform*. Alegerea limbajului de programare pentru dezvoltarea aplicației trebuie să țină cont de această cerință.

- Declarea aplicației *open source* sub licență GPL (General Public License) în scopul unei continue dezvoltări.

- Proiectare, realizarea și testarea unui echipament hardware original de realitate virtuală specific manipulării.

- Implementare echipamentului hardware în aplicația realizată.

- Realizarea de cercetări experimentale privind evaluarea percepției profunzimii modelelor CAD utilizând vizualizare de tip stereoscopic.

- Realizarea de cercetări experimentale privind percepția manipulării modelelelor CAD utilizând echipamentul hardware realizat.

- Testarea experimentală a eficienței noului sistem VR-CAD realizat.

- Proiectarea, realizarea și implementarea unui mediu de lucru colaborativ.
- Testarea aplicației în mediu de lucru colaborativ.

1.6 Concluzii

Deoarece cele mai multe companii s-au focalizat pe linia productivității ca și răspuns la competiția globală, migrația spre sistemele CAD, CAM și CAE a stabilit o nouă temelie în dezvoltarea industriei moderne.

Pentru început s-a acționat prin îmbunătățirea calității producției și reducerea costului de comercializare. În ultimii cinci ani, cele mai multe companii de automobile și aerospațiale au investit mulți bani în dezvoltarea și implementarea unor noi tehnologii bazate pe realitate virtuală. Noile tehnologii au dus la crearea unui mediu sintetic care a devenit un valoros instrument în aria factorului uman și aplicabilitatea studiilor în producție și simulare bazată pe proiectare.

Îmbunătățirea metodelor de producție prin *design digital*, în special CAD, reduc timpul de proiectare. Interfața cu utilizatorul este de obicei "desktop-based" și astfel modele de obicei nu sunt reprezentate în mărimea lor naturală. O mare parte a timpului de modelare este utilizat pentru navigare și selecția obiectelor. Deoarece aceste sisteme au un număr mare de componente, module de modelare și funcții de ansamblare, este necesar un timp adițional la navigare și selecție pentru parametrizarea geometriei modelelor.

Astfel, în acest capitol este prezentat stadiul actual al sistemelor VR - CAD. Exemplele prezentate pe scurt, fac o introducere a realizărilor integrării softurilor CAD și VR la diferite nivele, prin cuplarea completă online a aplicațiilor sau prin integrarea funcționalității nucleului CAD în sistemele VR.

În concluzie, se poate spune ca după o anumită perioadă de timp, Realitatea Virtuală va include CAD.

Capitolul 2

Limbaje de Realitate Virtuală

2.1 Introducere

Pentru programarea aplicațiilor grafice complexe necesare în crearea și redarea scenelor virtuale din mediile virtuale, se utilizează biblioteci și interfețe grafice, sisteme de dezvoltare de programe (*Toolkit*), care oferă programatorului/proiectantului posibilitatea să reutilizeze un număr mare de funcții grafice deja elaborate și implementate și, astfel să se concentreze mai eficient asupra proiectării și complexității aplicației. Limbajele destinate RV oferă diverse facilități și interfețe în vederea proiectării și elaborării de aplicații. Unele oferă o vedere de înalt nivel, în cadrul căreia utilizatorul își poate crea întreaga aplicație, utilizând limbaje, scriptice sau nu, și chiar ustensile grafice, sistemul fiind responsabil cu întreaga simulare, geometrie și interacțiune. Altele se bazează pe o anumită configurație grafică, utilizând API-uri (*Application Programming Interface*) și chiar alte limbaje de programare, pentru a asigura o performanță cât mai ridicată.

Biblioteca grafică OpenGL (GL de la *Graphic Library*; `http://www.opengl.org`) oferă o interfață simplă și directă pentru operațiile fundamentale de redare 3D. Pe un nivel superior de abstractizare, se situează OpenIV (IV de la Inventor) și Java3D al caror model al programarii permite efectuarea de sarcini dificile cum ar fi traversarea grafului scenei, gestionarea modificărilor atributelor stărilor etc., simplificând astfel sarcina aplicației. În particular, sunt disponibile implementări Java3D care utilizează Direct3D și OpenGL, OpenIV fiind de la bun început bazat pe OpenGL.[61]

2.2 VRML (Virtual Reality Modeling Language)

Istoric al limbajului

Scopul acestui limbaj este tranziția de la o interfață text a Web-ului la una având trei dimensiuni, în permanentă interacțiune cu utilizatorul.

VRML a fost creat de Mark Pesce și Anthony Parisi, cu ajutorul lui Gavin Bell și Brain Behlendorf. Primii doi au hotărât să proiecteze o interfață 3D pentru Web. Limbajul de modelare a acestei interfețe (VRML) trebuia să fie diferit de HTML, dar nu au dorit să inventeze un nou limbaj. După ce au trecut în revistă tehnologiile existente la acea dată, au investigat mai mulți candidați. După o serie de deliberări au ales formatul de fișier ASCI Open Inventor de la Silicon Graphics Inc. (SGI). Unul din motivele principale pentru care a fost adoptat Open Inventor este faptul că acesta permite o descriere completă a grafurilor de scenă 3D. Open Inventor include, de asemenea, facilități de iluminare, descrieri de materiale și texturi, precum și efecte speciale. Astfel, un subset al formatului de fișier SGI Open Inventor, cu adăugarea extensiilor de acces la rețea, a reprezentat baza VRML-ului.

VRML a fost proiectat în primăvara anului 1994, fiind prezentat la prima conferință WWW din Geneva, la dezvoltarea sa avându-se în vedere independența de platformă, extensibilitatea și abilitatea de a lucra în cadrul rețelelor cu lățime mică de bandă. Primul program suportând VRML (care încărca imaginea 3D a unei banane și o afișa pe ecranul calculatorului) s-a numit Labyrinth.

VRML s-a lansat public la data de 3 aprilie 1995 și specificația VRML versiunea 1.0 a urmat în luna mai a aceluiași an. Versiunea 1.0 a specificației VRML a fost pusă la punct de Gavin Bell, Anthonz Parisi și Mark Pesce. Ulterior a apărut VRML 1.1.

Pe 4 august 1996 se dă publicității specificația versiunii 2.0, permițând generarea de lumi virtuale animate și având o serie de facilitați noi precum:

- elemente de control geometric și proprietăți ale culorilor pentru forme;
- senzori simplificând operațiile efectuate cu ajutorul mouse-ului (*drag-and-drop*);
- gruparea nodurilor care permit detectarea coliziunii obiectelor;
- atribute de iluminare cum ar fi *fog* (ceața);

- furnizarea de informații despre scenă către browserele VRML:
- facilități pentru utilizarea surselor de sunet.

În 1997 a apărut o specificație apropiată de VRML 2.0 numită VRML97, iar în prezent un nou standard numit X3D (Extensible 3D).

X3D este succesorul VRML-ului adăugându-i noi capabilități: interfețe pentru aplicații avansate de programare, formate adiționale pentru *data encoding* incluzând *Extensible Markup Language* (XML), extinderea capabilității grafului și suport pentru ultimele arhitecturi hardware.

Criterii de design

VRML a fost proiectat pentru a îndeplini următoarele cerințe [70]:

- Independența de editor: Face posibilă utilizarea de generatoare și editoare disponibile pe orice platformă și posibilitatea de a importa date din alte formate.
- Perfecționarea: înzestrat cu toate informațiile necesare pentru implementarea și adresarea unui set de trăsături complete pentru acceptarea unei largi industrii.
- Compunerea: Abilitatea de a folosi elementele VRML-ului în diferite combinații și astfel permițându-se reutilizarea lor în alte contexte.
- Extensibilitatea: Abilitatea de a adauga noi elemente în cadrul lumilor VRML.
- Implementarea: Capabil de implementări variate pe un număr mare de sisteme.
- Potențialul multi-utilizator: Implementarea lumilor VRML în vederea folosirii în contextele utilizatorilor multipli (medii multi-utilizator).
- Ortogonalitatea: Elementele limbajului trebuie să fie independente unul de altul, iar orice dependență posibilă trebuie să fie structurată și bine definită.
- Performanța: Elementele trebuie să fie proiectate ținând cont de importanța și de performanțele interactive pe o varietate de platforme (software și hardware).
- Scalabilitatea: Elementele VRML trebuie să fie proiectate pentru o infinitate de compoziții spațiale.

- Standardizarea practică: Elementele VRML trebuie să suporte aplicații existente sau măcar propuneri care urmează a fi standardizate.
- Structuri bine-definite: Un element trebuie să posede o interfață bine-definită și o simplă stare cu scop necondiționat. Elementele cu scopuri multiple și efecte secundare trebuie evitate.

Limbajul VRML a fost standardizat de organizațiile ISO (Organizația Internațională pentru Standardizare) și IEC (Comisia Electrotehnică Internațională) constând din două părți: partea 1 (ISO/IEC 14772-1:1997) definește specificațiile funcționale și codificarea UTF-8 [11], iar partea 2 (ISO/IEC 14772-2:2004) *External authoring interface* (EAI). În limbajul VRML se pot realiza reprezentări statice și animate de obiecte și pot exista hiper-legaturi către alte elemente multimedia, cum ar fi sunetele și filmele. Interpretoarele sunt disponibile (de cele mai multe ori gratuit) pentru diferite platforme, la fel cum există și diverse instrumente capabile să creeze și să manipuleze fișierele VRML.

Prezentare generală

Un fișier VRML este un fișier ASCII care conține descrierea unei scene virtuale și a unor acțiuni interactive. Pentru a afișa o scenă grafică din rețeaua Internet sau dintr-un fișier de pe disc (cu extensia .wrl) este necesar un browser VRML, configurat ca plug-in al unui browser Web (ex: Cortona VRML Client, Cosmo Player). Observarea scenei virtuale și navigarea în direcția dorită se realizează prin comenzi ale browserului.

Formatul VRML permite construirea grafului scenei, furnizând toate informațiile despre obiectele tridimensionale, ierarhia acestora, condițiile de mediu ambiant, etc. Graful scenei, construit din unul sau mai multe fișiere VRML, este apoi traversat pentru obținerea imaginii obiectelor în diferite situații de navigare a scenei. Redarea imaginii pe ecran se realizează prin intermediul bibliotecilor grafice de nivel mai scăzut, cum sunt OpenGL sau Direct3D. Aceste biblioteci asigură interfața corespunzătoare cu acceleratorul grafic al sistemului.

Noțiunea de graf al scenei

Graful scenei reprezintă o noțiune de bază în grafica 3D. Pe acest concept se bazează construirea lumilor virtuale atât în VRML cât și în Java 3D. În continuare este

descrisă această noțiune [5].

O scenă virtuală (virtual scene) este compusă dintr-o colecție de modele de obiecte tridimensionale, specificate prin forma și poziția lor, prin aspect (culoare, material etc) și comportare (deplasare în spațiu, interacțiune).

Construirea ierarhică a scenei permite organizarea și parcurgerea eficientă a acesteia. Scena ierarhică este compusă din noduri conectate prin arcuri direcționate. Un nod în scenă descrie o anumită entitate: un obiect tridimensional, o grupare de obiecte tridimensionale, o transformare geometrică, o textură etc. Construirea ierarhică a scenei permite reutilizarea obiectelor și a transformărilor geometrice. Un obiect complex se compune din gruparea mai multor obiecte mai simple.

Redarea imaginii unei scene virtuale reprezentate printr-un graf se realizează prin parcurgerea acestuia. Un obiect tridimensional este instanțiat prin aplicarea transformărilor din nodurile de transformare aflate pe drumul de la nodul rădăcină la nodul terminal care reprezintă obiectul.

Structura fișierelor VRML

Structura generală a unui fișier VRML (de obicei având extensia wrl și aparținând tipului MIME x-world/x-vrml sau, mai nou, model /vrml) este dată de următoarele trei secțiuni:

1. Întotdeauna un fișier VRML începe cu un antet obligatoriu care indică navigatorului VRML versiunea de limbaj folosită;

   #VRML V2.0 <tipul codificarii> [comentariu opțional] <terminator de linie> Tipul codificării poate fi utf8 sau altă valoare autorizată și indică setul de caractere utilizat (de exemplu UTF-8, ISO 10646).

   #VRML V2.0 utf8

2. Comentariile sunt precedate de caracterul #;

3. Noduri, prototipuri și rute. Un fișier VRML poate conține noduri rădăcină, adică noduri care nu sunt conținute în alte noduri sau prototipuri. Nodurile rădăcină trebuie să fie noduri de tip children. Nodurile sunt aranjate în structuri ierarhice numite grafuri de scene, definind ordinea de reprezentare a nodurilor și folosind

noțiunile de stare și tranziție.

Sistemul de coordonate în care sunt afișate nodurile rădăcină este numit sistem de coordonate global (al lumii). Sistemele de coordonate definite relativ la transformările de la sistemele de coordonate ale ascendenților sunt numite sisteme de coordonate locale.

În VRML, un nod reprezintă orice entitate care grupează mai multe informații. Tipurile de noduri VRML sunt următoarele:

- forme și descrieri geometrice (*shapes*): *Box, Cone, Coordinate, Cylinder, EvaluationGrid, Extrusion, IndexedFaceSet, IndexedLineSet, Normal, PointSet, Shape, Sphere, Text*;
- aspect (*appearance*): *Appearance, Color, FontStyle, ImageTexture, Material, MovieTexture, PixelTexture, TextureCoordinate, TextureTransform*;
- grupuri (*grouping*): *Anchor, Billboard, Collision, Group, Inline, LOD, Switch, Transform*;
- mediu (*environment*): *AudioClip, background, DirectionalLight, Fog, PointLight, Sound, SpotLight*;
- vizualizare (*viewing*) *NavigationInfo, Viewpoint*;
- animație (*animation*): *ColorInterpolator, CoordinateInterpolator, NormalInterpolator, OrientationInterpolator, PositionInterpolator, ScalarInterpolator, TimeSensor;*
- interacțiuni (*interaction*) *CylinderSensor, PlaneSensor, ProximitySensor, SphereSensor, VisibilitySensor, TouchSensor*;
- Alte noduri: *Script, WorldInfo.*

Fiecare tip de nod are unul sau mai multe câmpuri.

Toate obiectele vizibile sunt definite în interiorul unor noduri. Aceste noduri au două câmpuri: *appearance* și *geometry*. *Appearance* defineste culoarea, textura ce sunt aplicate geometriei, iar *geometry* indică care forme (*shape*) vor fi desenate.

Sintaxă:

```
Shape {
      appearance NULL
      geometry NULL
}
```

Câmpul *appearance* este opțional, când acest câmp lipsește vor fi utilizate valorile default (pentru nodurile geometrice valoarea implicită este de 2 metri).

Prototipurile permit extinderea setului de tipuri de noduri de către utilizator. Definițiile de prototipuri pot fi incluse în fișierul în care sunt utilizate sau pot fi externe.

Rutarea evenimentelor permite realizarea unui mecanism, separat de ierarhia grafului scenei, prin care evenimentele pot fi propagate pentru a produce modificări în alte noduri.

Nodurile de tip script permit procesare arbitrară a evenimentelor. Pot adăuga sau îndepărta dinamic rute, modificând topologia rutării evenimentelor.

După antetul obligatoriu, un fișier VRML poate conține orice combinație din următoarele elemente:

- orice număr de declarații PROTO sau EXTERNPROTO;
- orice număr de declarații de nod;
- orice număr de declarații USE;
- orice număr de declarații ROUTE.

Sintaxa pentru reprezentarea nodurilor este:

```
[DEF <nume>] <tipul nodului> { <corpul> }
```

Sintaxa pentru reprezentarea câmpurilor:

```
<nume câmp> <valoare>
```

sau:

```
<nume câmp> [ <valori> ]
```

Sintaza pentru PROTO:

```
PROTO <nume> [ <declarare interfață> ] { <definiție> }
```

Declararea interfeței conține declarații de tip eventIn, eventOut, field și exposedField.

Sintaxa pentru USE: indică instanțierea unui nod anterior definit:

```
USE <nume>
```

Sintaxa pentru ROUTE:

```
ROUTE <nume>.<nume câmp/eveniment> TO <nume>.<nume câmp/eveniment>
```

Concepte în VRML

Semantica evenimentelor și câmpurilor

Câmpurile sunt în interiorul declarațiilor de noduri dintr-un fișier VRML și definesc starea stabilă a lumii virtuale. Prin *eventIn* și *eventOut* se definesc tipurile și numele evenimentelor pe care fiecare nod le poate emite sau recepționa. Fiecare nod interpretează valorile evenimentelor recepționate sau emise în concordanță cu implementarea sa.

Un câmp de tipul *exposeField* poate recepționa evenimente ca un *eventIn*, poate genera evenimente ca un *eventOut* și poate fi stocat în fișierul VRML ca un câmp obișnuit, *field*.

Există un set de recomandări pentru construirea numelor câmpurilor și evenimentelor:

- toate numele ce conțin cuvinte multiple încep cu minusculă, iar primul caracter al fiecărui cuvânt următor e majusculă;
- evenimentele de tip *eventIn* au prefixul set_;
- evenimentele de tip *eventOut* au prefixul _changed.

Procesarea evenimentelor

Conexiunea între nodul generator al unui eveniment și nodul receptor al evenimentului respectiv poartă numele de rută (route). Rutele nu sunt noduri. Ele sunt

declarații ce pot să apară oriunde în fișier, dar după ce toate nodurile referite în ele au fost definite. Tipurile de date conectate trebuie să coincidă (întreg pe 32 biți ↔ întreg pe 32 biți, flotant ↔ flotant etc.).

Din momentul în care un senzor sau un script a generat un eveniment inițial, evenimentul este propagat către toate nodurile spre care există legături prin rute. Aceste noduri pot răspunde producând noi evenimente, continuând până ce toate rutele au fost parcurse. Rezultă o cascadă de evenimente. Toate evenimentele dintr-o cascadă vor fi etichetate cu același moment de timp, fiind considerate ca având loc instantaneu.

Timpul

Programul browser controlează trecerea timpului într-o lume virtuală prin forțarea senzorilor de timp de a genera evenimente pe măsură ce timpul se scurge.

Navigarea

Conceptual, fiecare lume virtuală VRML conține un punct de vedere (*viewpoint*) din care lumea este privită la un moment dat. Navigarea este acțiunea utilizatorul de a modifica poziția și/sau orientarea acestui punct de vedere, prin aceasta modificând ceea ce utilizatorul vede. Aceasta permite utilizatorului să se miște prin lumea virtuală sau să examineze un obiect.

Semantica nodurilor

Fiecare nod are următoarele caracteristici [72]:

- un nume de tip; de exemplu: *Box, Color, Sphere*;
- nici unul, unul sau mai multe câmpuri, care particularizează nodul;
- o mulțime (chiar vidă) de evenimente pe care le poate emite sau recepționa;
- o implementare, care definește modul în care el recepționează la evenimentele pe care le poate recepționa și aparența sa vizuală sau auditivă în lumea virtuală (dacă există);
- un nume.

VRML este un limbaj *case-sensitive*; culorile modelelor în VRML sunt RGB (*Red, Green și Blue*), aceste valori sunt între 0.0 și 1.0. De exemplu 0.0 0.0 0.0 este negru, 0.0 0.0 1.0 este albastru și 1.0 1.0 1.0 este alb; Unitățile de măsură se presupune a fi în metri și unghiurile se măsoară în radiani, nu în grade, iar sistemele de coordonate sunt carteziene drepte tridimensionale. Implicit observatorul se află pe axa Z privind de-a lungul acestei axe spre originea sistemului, cu axa +X la dreapta și +Y în sus.

Pe lângă extensia normală wrl, multe navigatoare acceptă și extensia wrz, ce corespunde unor fișiere comprimate (gzip), care au avantajul că necesită un timp mai scurt de transfer. Aceste navigatoare au integrată funcția de decomprimare.

Exemple VRML

Fișierul de mai jos definește o scenă grafică compusă din trei obiecte: un cub, un cilindru și un con; rezultatul obținut este prezentat în Figura 2.1.

```
#VRML V2.0 utf8
Background{
     skyColor 1.0 1.0 1.0}
Transform {
     translation -3.0 0.0 0.0
     children [
          Shape {
               appearance Appearance {
               material Material {
               diffuseColor 0.0 1.0 0.0
                         }
                }
               geometry Sphere {
                         }
               }
               ]
      }
Transform {
     children [
          Shape {
               appearance Appearance {
               material Material {
               diffuseColor 0.0 0.0 1.0
                         }
                }
               geometry Cone {
                         }
               }
               ]
```

```
    }
Transform {
    translation 3.0 0.0 0.0
    children [
        Shape {
            appearance Appearance {
            material Material {
            diffuseColor 1.0 0.0 0.0
                        }
             }
            geometry Box {
                        }
            }
            ]
    }
```

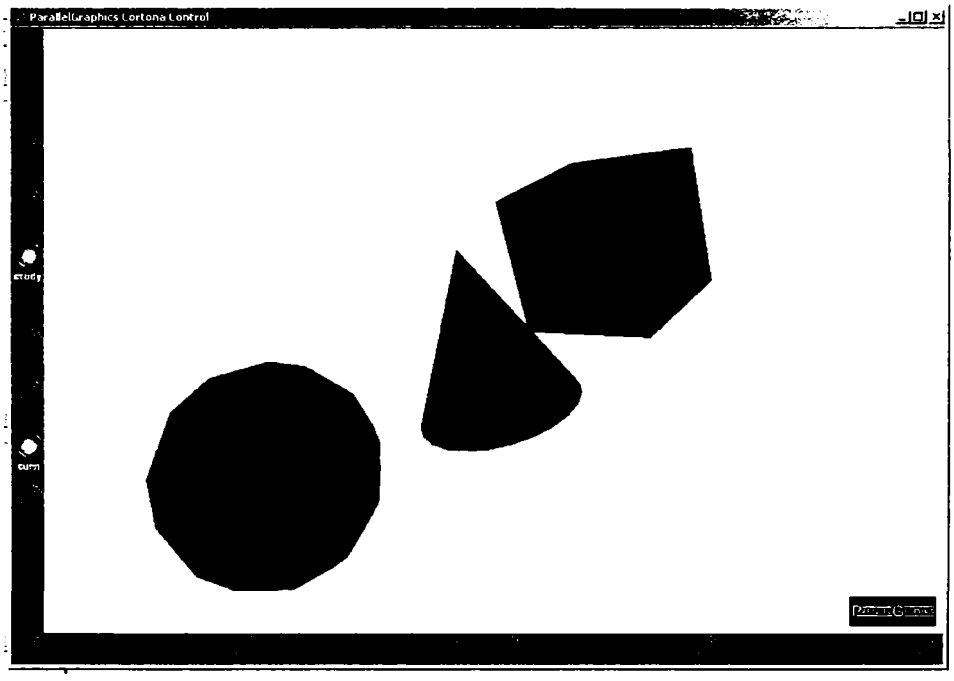


Figura 2.1: Exemplu de scenă VRML.

Următorul exemplu realizat de autor [14], prezintă utilizarea limbajul VRML la modelarea și controlul unui robot în scop educațional.

Figura 2.2 prezintă un *screenshot* al modelului VRML.

Robotul are o structură serială de tip RRRRR (SCORBOT). Fiecare din cele cinci cuple de rotație (Figura 2.4) pot fi controlate independent prin intermediul panoului de comandă virtual (Figura 2.3).

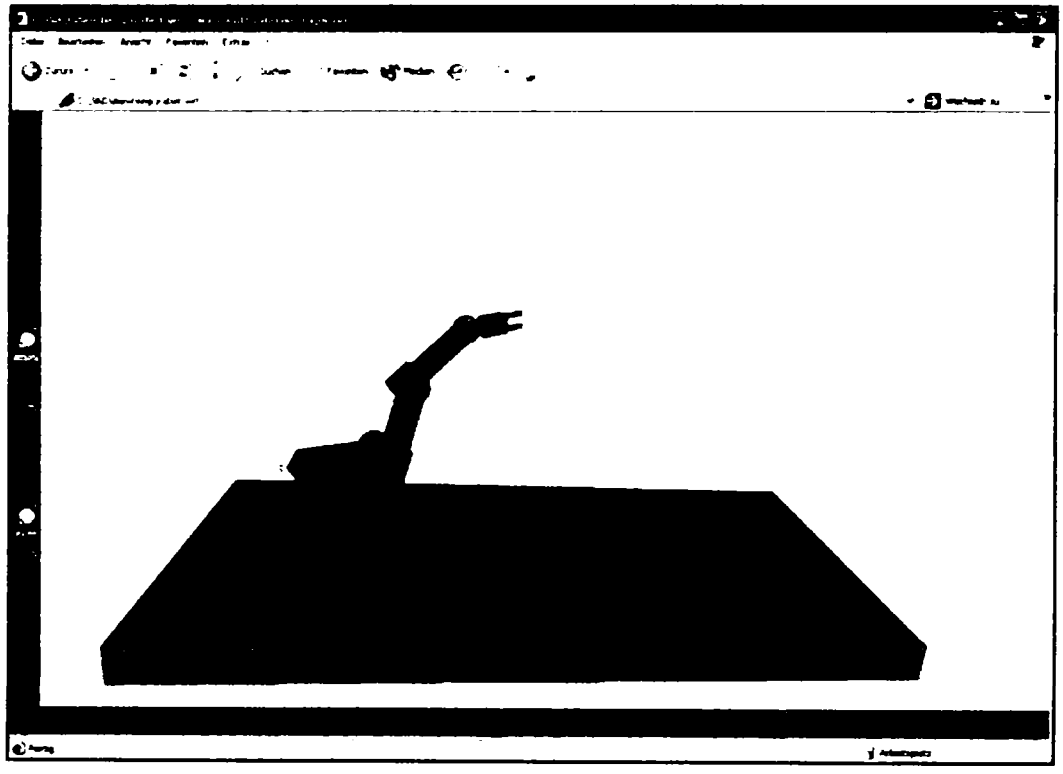


Figura 2.2: *Screenshot* modelarea si controlul unui robot in limbajul VRML.

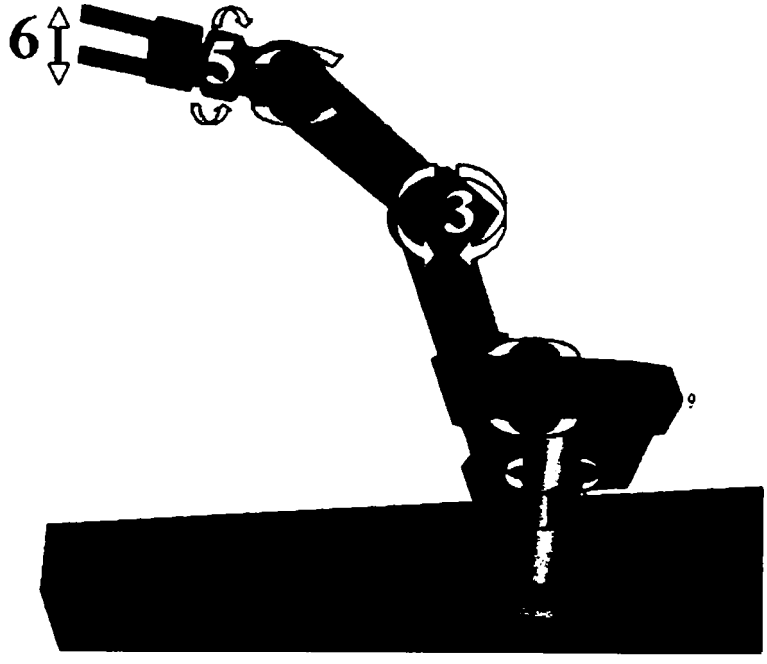


Figura 2.3: Robot SCORBOT modelat in VRML.

Figura 2.4: Panoul virtual de comandă a robotului.

În continuare este prezentată o secvență de cod VRML (utilizând nodul Script) ce realizează controlul mișcării independente a cuplelor de rotație (evenimente generate de apăsarea butoanelor panoului de comandă virtual) ale robotului modelat.

```
DEF control Script {
mustEvaluate TRUE
eventIn SFFloat master
eventOut SFBool autorepeat
field SFFloat rotdelta .05
#buton1 rot axa1  stanga dreapta
  field SFBool b1pos FALSE
  field SFBool b1neg FALSE
  field SFFloat axa1_rot 0
  field SFFloat axa1max 2.8
  field SFFloat axa1min -2.8
  eventIn SFBool axa1_lt
  eventIn SFBool axa1_rt
  eventOut SFRotation axa1_lt_rt


  url "vrmlscript:
  function master() {
    buton1 <
          if (b1pos) {
                axa1_rot += rotdelta;
                if (axa1_rot > axa1max)
                        axa1_rot = axa1max;
                axa1_lt_rt = new SFRotation(0,1,0,axa1_rot);
           }
           if (b1neg) {
                axa1_rot -= rotdelta;
                if (axa1_rot < axa1min)
                        axa1_rot = axa1min;
                axa1_lt_rt = new SFRotation(0,1,0,axa1_rot);
           }
        //buton1 >
```

2.3 X3D

X3D este un standard open pentru grafică 3D. Este următorul nivel al specificației VRML97 ISO, incorporând ultimele descoperiri și îmbunătățiri arhitecturale bazate pe ani de încercări pe modelul VRML97.

Limbajul X3D a fost supus pentru aprobarea standardizării de organizațiile ISO și IEC constând din trei părți: partea 1 (ISO IEC FDIS 19775:200x) definește conceptele abstracte, partea 2 (ISO IEC FCD 19776:200x) specifică codificarea fișierelor X3D utilizând *Extensible Markup Language* (XML), codificarea definită în X3D utilizând clasica codare VRML, iar partea 3 (ISO IEC FCD 19777:200x) specifică limbajul de programare [93].

Diferențe între VRML97 și X3D

Schimbările majore pot fi prezentate astfel, pe scurt:

- posibilitatea extensiei grafului de scenă;
- modelul programări revizuit;
- decriptări multiple de fișiere descriind același model abstract incluzând XML;
- arhitectura multiplă permite anumite niveluri de adoptare și suport pentru diferite și multiple tipuri de piețe;
- extinderea structurii de specificare.

Modificările aduse au ca rezultat extinderea numărului de noduri de la 74 cât are VRML97 la un număr de 114 noduri. X3D suportă multiple decriptări de fișiere, în principal VRML97 și în plus XML și compresii binare. Decriptarea XML oferă integrare ușoară cu servicii web și intersectarea între fișiere și transfer de date. Formatul binar compresat este în prezent în dezvoltare, finalizarea acestuia având ca rezultat o viteză mai mare de transfer a datelor.

X3D oferă o arhitectură modulară pentru a oferi o mai mare extensibilitate și flexibilitate. Cele mai multe domenii de aplicații nu necesită caracteristicile lui X3D și nu toate platformele suportă mulțimea de funcționalități definită în specificații. Trăsăturile lui X3D sunt grupate în componente care pot fi suportate prin implementări într-o

manieră "combină și potrivește" pentru a întâmpina nevoile unei anumite piețe sau platforme. De asemenea introduce conceptul de profile - colecții predefinite de componente întâlnite de obicei în anumite domenii de aplicații, platforme sau utilizate în diverse scenarii, Spre deosebire de VRML97, care solicită suport complet pentru toate caracteristicile pentru a fi în conformitate cu implementarea, X3D permite variații de la standard pentru a întâmpina diferite nevoi.

Mecanismul component al lui X3D permite de asemenea utilizatorilor să-și implementeze propiile lor extensii în acord cu un riguros set de reguli, evitând astfel ca utilizatorul să aștepte un anumit număr de ani până când acestea vor fi incluse în modelul VRML.

Ca și browser de fișiere X3D a fost lansat pachetul Xj3D. Așa cum îi arată și numele, Xj3D este bazat pe Java 3D și necesită bibliotecile Java 3D. Deoarece Java 3D necesită Java 2 întregul pachet este destul de mare (aproximativ 20 MB).

Un alt browser disponibil pentru extensia X3D, este Flux produs de MediaMachines. Acesta rulează numai pe platforma Windows și utilizează pentru randare Direct3D. Se instalează asemănător plugin-urilor pentru VRML (pentru Internet Explorer) și are o mărime de sub 1MB.

X3D suportă două codificări (*encoding*): VRML Clasic și XML. Există anumite utilitare care fac conversii între cele două formate. Prima modificare în VRML Clasic se referă la tipul codificării.

Pentru VRML 97 avem:

```
#VRML V2.0 utf8
```

iar pentru X3D avem:

```
#X3D V3.0 utf8 Profile Immersive
```

Se observă schimbarea în numele versiunii și anume mai apare și declararea profilului. Aceasta sugerează că nodurile X3D și capabilitățile vor putea fi utilizate în întrega lume. Primul fișier încărcat trebuie să precizeze capabilitățile maxime utilizate. Dacă un fișier extern (prin *Inline* sau *EXTERNProto*) utilizează o capabilitate care nu este declarată în fișierul programului de tip X3D, atunci se va genera o eroare.

O altă diferență se referă la pseudo-protocolul utilizat la identificarea codu-

lui script. În VRML 97 este utilizat javascript sau vrmlscript, iar în X3D este utilizat ECMAscript.

Gruparea pe categorii a nodurilor X3D

Tipuri de noduri X3D sunt:

1. noduri nucleu: *MetadataDouble, MetadataFloat, MetadataInteger, MetadataSet, MetadataString*;
2. noduri timp: *TimeSensor*;
3. noduri rețea: *Anchor, Inline, LoadSensor*;
4. noduri de grupare: *Group, StaticGroup, Switch, Transform, WorldInfo*;
5. noduri de randare: *Color, ColorRGBA, Coordinate, IndexedLineSet, IndexedTrinagleFanSet, IndexedTraingleSet, IndexedTriangleStripSet, LineSet, Normal, PointSet, TrinagleFanSet, TriangleSet, TriangleStripSet*;
6. noduri formă geometrică: *Appearance, FillProperties, LineProperties, Material, Shape*;
7. noduri geometrie 3D: *Box, Cone, Cylinder, ElevationGrid, Extrusion, IndexedFaceSet, Sphere*;
8. noduri geometrie 2D: *Arc2D, ArcClose2D, Circle2D, Disk2D, Polyline2D, Polypoint2D, Rectangle2D, TriangleSet2D*;
9. noduri text: *FontStyle, Text*;
10. noduri sunet: *AudioClip, Sound*;
11. noduri sursă de lumină: *DirectionalLight, PointLight, SpotLight*;
12. noduri de tip textură: *ImageTexture, MovieTexture, MultiTexture, MultiTextureCoordinate, MultiTextureTransform, PixelTexture, TextureCoordinate, TextureCoordinateGenerator, TextureTransform*;

13. noduri interpolatoare: *ColorInterpolator, CoordinateInterpolator, CoordinateInterpolator2D, NormalInterpolator, OrientationInterpolator, PositionInterpolator, PositionInterpolator2D, ScalarInterpolator*;

14. noduri senzori de dispozitiv indicator: *CylinderSensor, PlaneSensor, SphereSensor, TouchSensor*;

15. noduri senzori de dispozitiv *Key*: *KeySensor, StringSensor*;

16. noduri senzor de mediu: *ProximitySensor, VisibilitySensor*;

17. noduri de navigație: *Billboard, Collision, LOD, NavigationInfo, Viewpoint*;

18. noduri efecte de mediu: *Backgroud, Fog, TextureBackground*;

19. noduri geospațiale: *GeoCoordinate, GeoElevationGrid, GeoLocation, GeoLOD, GeoMetadata, GeoOrigin, GeoPositionInterpolator, GeoTouchSensor, GeoViewpoint*;

20. noduri de tip animație umanoidă (H-Anim): *HanimDisplacer, HanimHumanoid, HanimJoint, HanimSegment, HanimSite*;

21. noduri NURBS: *Contour2D, ContourPolyline, CoordinateDouble, NurbsCurve, NurbsCurve2D, NurbsOrientationInterpolator, NurbsPatchSurface, NurbsPositionInterpolator, NurbsSet, NurbsSurfaceInterpolator, NurbsSweptSurface, NurbsSwunqSurface, NurbsTextureCoordinate, NurbsTrimmedSurface*;

22. noduri de tip simulare interactivă distribuită (DIS): *EspduTransform, ReceiverPdu, SignalPdu, TransmitterPdu*;

23. noduri de tip script: *Script*;

24. noduri de tip eveniment: *BooleanFilter, BooleanSequencer, BooleanToggle, BooleanTriqqer, IntegerSequencer, IntegerTriqqer, TimeTriqqer.*

2.4 Java3D

Prezentare generală

Java 3D oferă programatorilor Java posibilitatea de a scrie applet-uri și aplicații Java cu conținut grafic 3D interactiv.

Java 3D reprezintă o interfață de programare (API) de nivel înalt, orientată spre obiecte. Comparativ, API-urile 3D procedurale, DirectX sau OpenGL, au fost proiectate pentru a optimiza la maximum performanțele, oferind programatorilor un control total asupra aplicațiilor; în schimb, Java 3D reprezintă o cale prin care un programator Java experimentat poate crea grafică 3D suficient de performantă, fără să fie necesar să cunoască toate detaliile implementărilor de nivel scăzut.

Aplicațiile Java 3D sunt performante deoarece codul Java al acestor aplicații nu este interpretat de mașina virtuală Java (asemenea aplicațiilor Java standard), ci se realizează o compilare *Just-In-Time*; adică în momentul în care clasa este descărcată de browserul de web, codul Java este tradus în codul mașinii gazdă.

Biblioteca Java 3D este foarte complexă, conținând peste 100 de clase, care permit crearea celor mai variate aplicații de grafică tridimensională sau realitate virtuală cu sunet, animație și interacțiuni. Totuși, un univers simplu poate fi creat doar cu puține clase și funcții.

Toate resursele necesare compilării și execuției apleturilor / aplicațiilor Java 3D sunt disponibile gratuit pe site-ul firmei Sun, `http://java.sun.com`.

Resurse pentru scrierea aplicațiilor: J2SE SDK (include și JRE); Java 3D SDK (include și Java 3D for Windows Runtime for JRE)

Resurse (doar) pentru rularea aplicațiilor: Java Runtime Environment (JRE); Java 3D for Windows Runtime for JRE.

Reprezentarea grafului scenei virtuale

Graful scenei este compus din noduri și arce. Un nod este un element care conține date, iar un arc reprezintă legătura (relația) dintre două noduri. Arcele pot reprezenta două tipuri de relații: relația părinte-fiu și relația de referire.

Cea mai comună relație este relația părinte-fiu. Un nod de grupare (instanță

a clasei *BranchGroup* sau *TransformGroup*) poate avea un număr oricât de mare de fii dar nu poate avea decât un singur părinte. Un nod terminal (*Leaf*) poate avea un singur parinte, dar nici un fiu. Se observă deci că în Java 3D graful scenei este de fapt un arbore, fiecare nod având un singur parinte (cu excepția nodului radăcină). De aceea există o singură cale (*path*) de la rădăcină la fiecare nod terminal.

Alt tip de relație între noduri intră în categoria de referință. O referință asociază un nod din graful scenei cu un obiect de tipul (clasa) *NodeComponent*. Un obiect de tip *NodeComponent* conține atribute referitoare la geometria (*Geometry*) și aspectul (*Appearance*) obiectelor. Obiectele referite pot fi partajate permițând astfel folosirea lor din orice parte a grafului scenei.

Pentru a reprezenta graful unei scene Java3D se folosește un set standard de simboluri (Figura 2.5), preluată din Java 3D Tutorial [83].

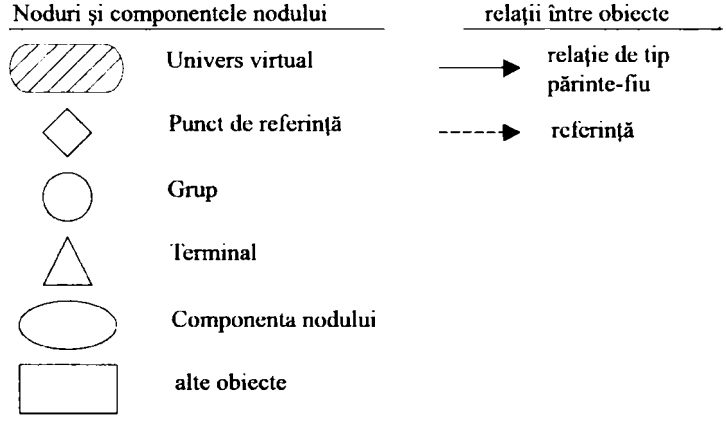


Figura 2.5: Simbolurile prin care se reprezintă obiectele în graful scenei.

Primele două simboluri reprezintă obiecte din clase specifice: Univers virtual reprezintă un obiect din clasa *VirtualUniverse*, iar Punct de referință reprezintă un obiect din clasa *Locale*. Următoarele trei simboluri reprezintă obiecte din clasele *Group*, *Leaf* și *NodeComponent*. Ultimul simbol din stânga reprezintă orice alt obiect. Simbolul săgeată continuă reprezintă o relație de tip părinte-fiu dintre două obiecte. Săgeata întreruptă reprezintă o referință la un obiect din orice alt tip.

Un exemplu de univers virtual este reprezentat în Figura 2.6 (preluată din Java 3D Tutorial [83]).

Un graf al scenei conține, de regulă, o singură instanță a clasei *VirtualUniverse*. Chiar dacă este posibil să se creeze mai multe universuri virtuale (fiecare cu rădăcina un obiect *VirtualUniverse*), acest lucru nu este recomandabil.

Fiecare obiect *VirtualUniverse* conține o listă de obiecte *Locale*, deci se pot defini mai multe puncte de referință în universul virtual; cea mai mare parte din aplicații definesc un singur obiect Locale, deci un singur punct de referință în universul virtual.

Fiecare obiect *Locale* este rădăcină a mai multor (de regulă două) subgrafuri care încep (au ca rădăcină) un nod instanță al clasei *BranchGroup*.
Există două categorii diferite de subgrafuri ale scenei (Figura 2.6): subgraf de vizualizare (*view branch graph*) și subgraf de conținut (*content branch graph*).

Un subgraf de conținut specifică conținutul scenei virtuale - obiecte, aspecte, materiale, comportare, lumini etc.

Un subgraf de vizualizare specifică parametrii de vizualizare: proprietățile ecranului, localizarea punctului de observare, și alte caracteristici specifice realității virtuale (avatari, dispozitive de urmărire a poziției observatorului etc.).

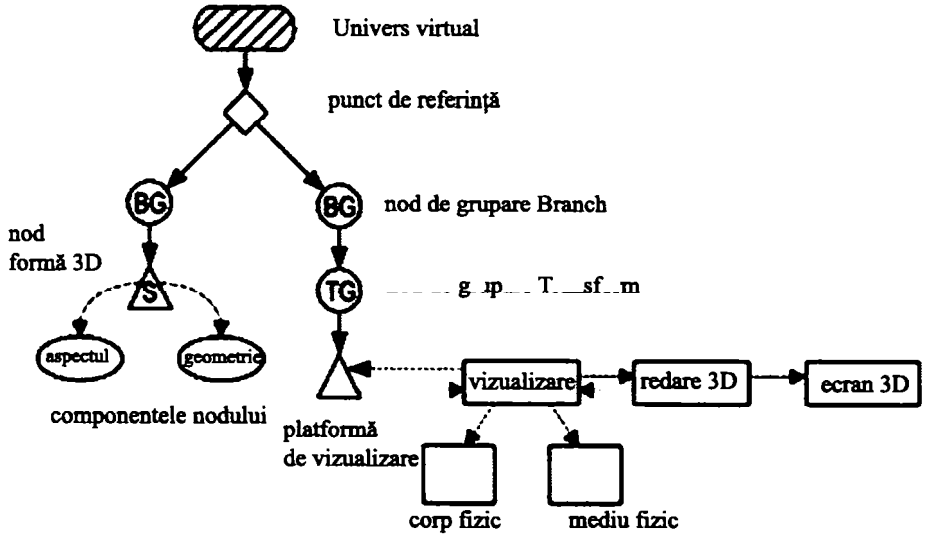


Figura 2.6: Un exemplu simplu de graf al scenei virtuale.

Dezvoltarea aplicațiilor Java 3D

Majoritatea aplicațiilor Java 3D se pot dezvolta folosind aceeași structură (implicită) a subgrafului de vizualizare și numai aplicațiile complexe sau specializate necesită ajustarea unor structuri specifice ale componentelor subgrafului de vizualizare (*PhysicalBody* pentru vederea stereoscopică, *PhysicalEnvironment* pentru dispozitive de intrare etc.). Mai trebuie menționat faptul că sistemul de vizualizare Java 3D (convențiile de sisteme de referință și transformările de vizualizare) este același cu cel din biblioteca OpenGL.

În cazul aplicațiilor simple, subgraful de vizualizare se construiește ca o instanță a clasei *SimpleUniverse*, care prevede un singur univers virtual (obiect *VirtualUniverse*) cu un singur punct de referință (obiect *Locale*) și cu valori implicite ale platformei de vizualizare.

Obiectul *SimpleUniverse* conține toată structura de obiecte din partea dreaptă a figurii 2.6: un obiect *VirtualUniverse* și subgraful de vizualizare.

În aceste cazuri, se poate urma un algoritm simplu de creare a unei aplicații Java 3D, care constă din următorii pași:

- se creează un obiect de tip *Canvas3D*, care reprezintă suprafața de redare;
- se creează un obiect de tip *SimpleUniverse* care se referă la obiectul *Canvas3D* creat anterior. Deoarece acest obiect conține punctul de observare, se specifică poziția inițială a punctului de observare care în mod implicit este în originea sistemului de referință;
- se creează subgraful de conținut al scenei, conform cerințelor aplicației;
- se inserează subgraful de conținut al scenei în obiectul *Locale* al universului (*SimpleUniverse*).

Prin inserarea unui subgraf în obiectul Locale, acesta devine activ (live) și fiecare obiect al său va fi redat în funcție de aspectul și poziția sa.

Programul de generare a imaginii (render) începe atunci când un obiect View (conținut în subgraful de vizualizare) devine activ (prin inserare în obiectul Locale) și după aceasta se execută o buclă infinită care efectuează următoarele operații (specificate în pseudocod):

```
while(true){
Prelucrează intrările;
if (cerere de terminare) break;
Efectuează operațiile de comportament (behaviors);
Traversează graful scenei și redă imaginea obiectelor
vizuale;
}
```

Exemplificarea pașilor de construire a unei aplicații Java 3D este dată în următorul program minimal, HelloJava3D [83].

```
//
// HelloJava3D.java
//
import java.applet.Applet;
import java.awt.*;
import java.awt.GraphicsConfiguration;
import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.geometry.ColorCube;
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.behaviors.mouse.*;
import javax.media.j3d.*;
import javax.vecmath.*;
public class HelloJava3D extends Applet {
public BranchGroup createSceneGraph() {
    BranchGroup objRoot = new BranchGroup();
    TransformGroup objTrans = new TransformGroup();
    objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
    objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
    objTrans.addChild(new ColorCube(0.4));
    objRoot.addChild(objTrans);
    // Se compileaza subgraful
    objRoot.compile();
    return objRoot;
    }
public void init() {
    setLayout(new BorderLayout());
    // Pasul 1: crearea obiectului Canvas3D
    GraphicsConfiguration config =
    SimpleUniverse.getPreferredConfiguration();
    Canvas3D canvas3D = new Canvas3D(config);
    add("Center", canvas3D);
    // Pasul 2: crearea obiectului SimpleUniverse
    SimpleUniverse simpleUniverse = new SimpleUniverse(canvas3D);
    // Setarea poziției observatorului la (0, 0, 2.41)
    simpleUniverse.getViewingPlatform().setNominalViewingTransform();
    // Pasul 3: crearea subgrafului de continut al scenei virtuale
```

```
        BranchGroup scene = createSceneGraph();
        // Pasul 4: inserarea subgrafului de continut al scenei
        simpleUniverse.addBranchGraph(scene);
        }
public static void main(String[] args) {
        new MainFrame(new HelloJava3D(), 256, 256);
        }
}
```

În funcția init() a applet-ului se pot urmări cei patru pași de creare a scenei virtuale.

Setarea poziției observatorului se poate face în mai multe moduri. Pentru început s-a apelat funcția *setNominalViewingTransform()* care inițializează poziția observatorului în punctul de coordonate (0, 0, 2.41), cu direcția de observare spre z negativ, ceea ce permite observarea comodă a obiectelor (plasate, în general, în originea sistemului de referință). Metodele și clasele implicate în aceste operații se pot găsi în documentația Java 3D.

Subgraful de conținut al scenei virtuale se crează în funcția *createSceneGraph()*, care returnează referință la un obiect de tipul *BranchGroup*, care se inserează în obiectul *SimpleUniverse*. În funcția *createSceneGraph()* se poate urmări modul de creare a diferitelor tipuri de noduri.

Un nod de tipul *BranchGroup* este un nod de grupare, care devine nodul părinte al tuturor nodurilor fii care se inserează prin funcția *addChild()*.

Un nod de tipul (clasă) *TransformGroup* poate, de asemenea, să fie părinte pentru toate nodurile fii care se inserează prin funcția *addChild()*, la fel ca și nodul de tipul *BranchGroup*. Însă, în plus, nodul *TransformGroup* mai conține o matrice de transformare (de dimensiune 4X4 în reprezentare coloană majoră, la fel ca în OpenGL), care este un obiect de clasa *Transform3D*. Această transformare se aplică (în cursul redării imaginii) tuturor obiectelor din subgraful al cărui rădăcină este, prin compunerea cu toate celelalte transformări de pe calea de la rădăcina grafului scenei până la obiectele tridimensionale (din noduri terminale - *Leaf*) redate.

O matrice de transformare (obiect *Transform3D*) poate fi prelucrată printr-un număr mare de funcții membre, prin care se poate modifica matricea în întregime sau diferitele ei componente (submatricea de rotație, vectorul de translație). Pentru ca o matrice de transformare a unui nod *TransformGroup* să poată fi accesată în cursul redării,

trebuie să fie setată capabilitatea de citire (*ALLOW_TRANSFORM_READ*), respectiv de scriere (*ALLOW_TRANSFORM_WRITE*) a nodului respectiv.

Nodurile terminale ale grafului conțin (de regulă) modele de obiecte tridimensionale, instanțe ale clasei *Shape3D* (sau ale unor subclase ale acesteia, cum este clasa *ColorCube*).

După crearea unui subgraf, se poate invoca funcția compile(), membră a clasei *BranchGroup*, care efectuează optimizarea subgrafului respectiv din punct de vedere al vitezei de parcurgere (traversare) a acestuia. În principal, optimizarea compune, dacă este posibil, noduri succesive de transformare (*TrasformGroup*), ceea ce reduce numărul de operații de transformări geometrice.

Graful scenei realizat în acest program este dat în Figura 2.7.

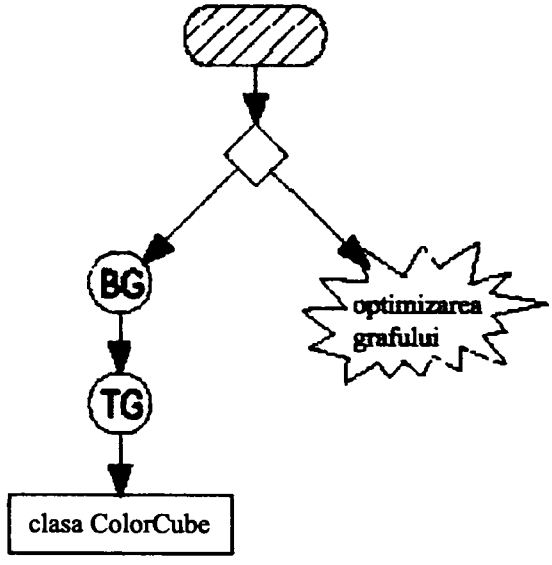


Figura 2.7: Graful scenei creat în programul HelloJava3D.

La execuția programului se va vedea imaginea unui cub. De fapt se vede un pătrat pe ecran, care este fața roșie a cubului (Figura 2.8).

Animația pentru rotirea cubului este realizată adăugând următoarele linii de cod în funcția createSceneGraph():

```
public BranchGroup createSceneGraph() {
...
    // Animatie
    Alpha rotationAlpha = new Alpha(-1, 10000);
```

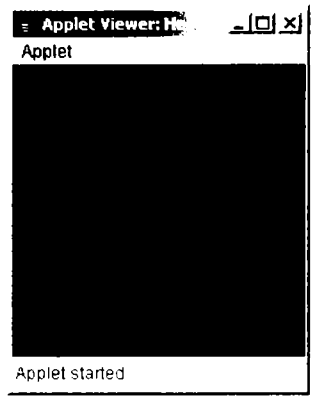


Figura 2.8: Rezultatul executiei programului HelloJava3D.

```
        RotationInterpolator rotator
        new RotationInterpolator(rotationAlpha, objTrans);
        BoundingSphere bounds = new BoundingSphere();
        rotator.setSchedulingBounds(bounds);
        objTrans.addChild(rotator);
            Se compileaza graful
        objRoot.compile();
        return objRoot;
    }
```

Biblioteca Java 3D oferă un număr mare de clase care permit modificarea în cursul generării imaginii a poziției obiectelor din scenă, a poziției observatorului, sau a grafului scenei (prin inserarea sau stergerea nodurilor). Astfel de modificări se efectuează ca răspuns la diferite evenimente (evenimente AWT, trecerea unui anumit interval de timp etc) si se prevăd la crearea scenei virtuale prin definirea comportării (*behaviors*) obiectelor (noduri) ale scenei virtuale.

În general, comportarea obiectelor scenei se defineste prin inserarea în graful scenei virtuale a unor noduri care sunt instante ale unor clase specifice animației (de exemplu *Rotation Interpolator*) sau interacțiunilor (de exemplu, *MouseRotate*). Astfel de noduri sunt instante ale unor clase derivate din clasa *Behavior*, care la rândul ei extinde clasa abstractă *Leaf*, deci nu pot fi decât noduri terminale. Aceste noduri *Behavior* primesc referința nodului din graf care este tinta modificărilor ce se vor efectua.

În exemplul de mai sus, obiectul rotator de tip *RotationInterpolator* aplică nodului *objTrans* o rotație în jurul axei Y; referința acestui nod țintă (*objTrans*) este pasată obiectului de interactiune *rotator*. Definirea comportării obiectelor scenei modifică graful

scenei, prin inserarea de noi noduri și prin definirea referințelor la nodurile țintă.

În Figura 2.9 este dat noul graf al scenei, după inserarea nodului de interacțiune, notat cu B (de la *Behavior*).

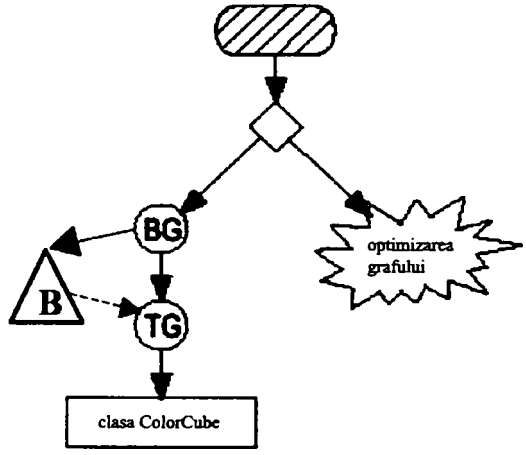


Figura 2.9: Graful scenei HelloJava3D după introducerea nodului de tip MouseRotate.

Înlocuind liniile de cod de la animația pentru rotirea cubului cu următoarele linii de cod în funcția *createSceneGraph()*, rotirea cubului va fi executată de către user prin interacțiunea cu ajutorul mouse-ului.

```
    Miscarea cu mouse-ul
MouseRotate myMouseRotate = new MouseRotate();
myMouseRotate.setTransformGroup(objTrans);
BoundingSphere bounds = new BoundingSphere();
myMouseRotate.setSchedulingBounds(bounds);
objRoot.addChild(myMouseRotate);
```

Exemplu Java3D

În următorul exemplu, autorul [19], prezintă o aplicație Java3D care modelează și simulează funcționarea unui robot de tip Scora ER-14 (Figura 2.11). Modelarea robotului a fost realizată conform modelului real existent în laboratorul CIM din Departamentul de Mecatronică.

În figura 2.10, este prezentat panoul de comandă real alături de cel virtual.

O restricție severă a acestei aplicații, o reprezintă faptul că utilizatorul poate interacționa cu lumea virtuală doar prin intermediul mouse-ului.

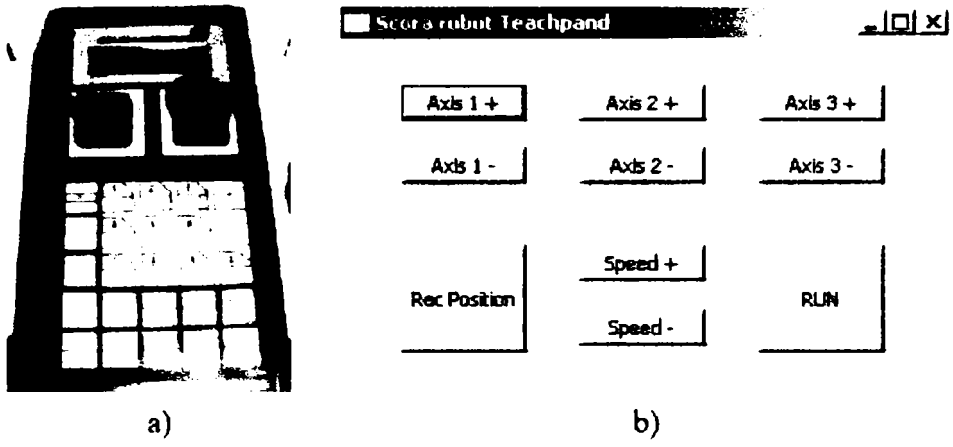


Figura 2.10: *Teach pendant*: a) model real, b) model virtual.

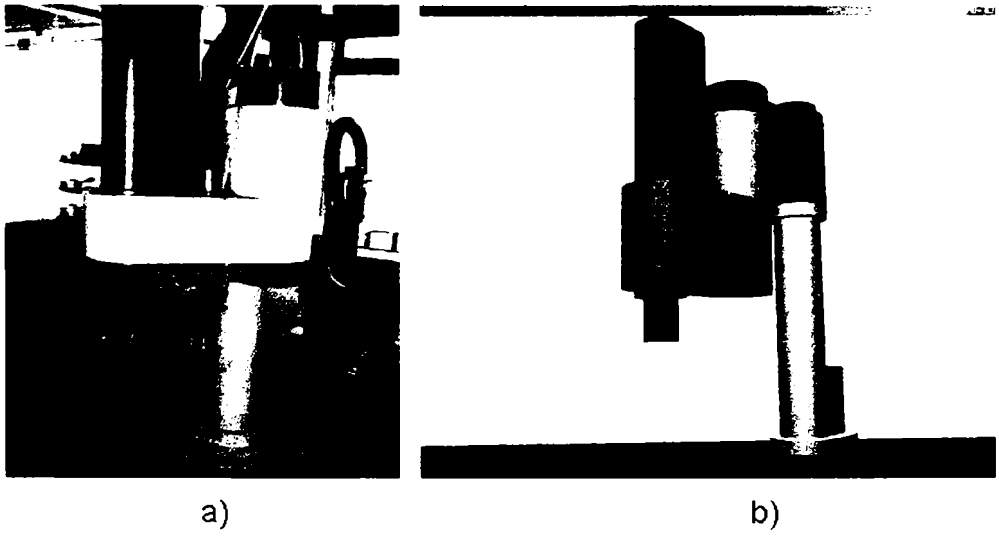


Figura 2.11: Robotul Scora ER 14: a) model real, b) model virtual.

Comparație cu VRML

S-a observat mai sus că există o anumită similaritate între nodurile grafului în VRML, respectiv în Java 3D. Principalul avantaj al tehnologiei Java 3D îl constituie puterea limbajului orientat pe obiecte Java. Astfel, din acest considerent decurg următoarele avantaje:

- posibilitatea de a defini obiecte care se construiesc luându-se în considerare un număr de parametri
  EX: Dacă am definit în VRML un templu constituit din 3 rânduri de câte 6 coloane și vrem să obținem un nou templu, dar de această dată format din 4 rânduri a câte 8 coloane, codul pentru obiect trebuie rescris. În schimb, în Java 3D obiectul templu poate primi valorile de mai sus ca parametri la construcție (alți parametri pot defini anumite dimensiuni etc).

- repetarea unor obiecte în cadrul scenei este mult mai simplă: acest lucru este cu atât mai evident cu cât pozițiile și orientarea noilor instanțe necesită anumite calcule
  EX: Dacă dorim să realizăm un cerc format din 100 de obiecte de tip Box, în VRML, trebuie să calculăm, înainte de a scrie codul, 100 de poziții diferite și de orientări diferite. În schimb, în Java 3D calculele le vom efectua chiar în program, iar codul rezultat va fi mult mai scurt.

Avantajul tehnologiei VRML este acela al simplității. Tehnologia se învață mai rapid, iar pentru crearea și vizualizarea lumilor virtuale sunt necesare mai puține resurse (doar interpretorul de VRML). Dar, după cum s-a vazut și mai sus, scrierea directă a codului VRML pentru scene complexe devine foarte dificilă. În astfel de cazuri se vor folosi aplicații care generează cod VRML (de ex. editoare CAD).

Un lucru interesant este faptul că cele două tehnologii pot fi folosite impreună. Se pot scrie aplicații Java 3D care citesc un fișier VRML și crează un graf de scenă conform specificațiilor din fișier. De asemenea, în cadrul unei aplicații Java 3D se poate realiza o parcurgere a grafului astfel încât să se genereze codul VRML corespunzător.

2.5 VisualPython

VPython este un modul 3D grafic numit "Visual" (dezvoltat de către David Scherer în perioada în care a fost student la Universitatea Carnegie Mellon) adăugat limbajului de programare Python.

Istoricul Python:

Python a fost creat la începutul anilor 1990 de către Guido van Rossum la Stichting Mathematisch Centrum din Olanda, ca succesor al limbajului ABC. Guido a rămas principalul autor al Pythonului, care acumulează multe contribuții ale altora. În 1995 Guido își continuă munca la Python la Corporation for National Research Initiatives în Reston, Virginia, unde a lansat câteva versiuni ale software-ului. În mai 2000, Guido și echipa de dezvoltare a Python-ului s-a mutat la BeOpen.com, unde a format echipa BeOpen PythonLabs. În octombrie 2000, colectivul PythonLabs s-a mutat la Digital Creations [28]. În 2001 ia naștere Python Software Foundation (PSF) [67] o organizație non profit creată special pentru a deține drepturile de autor referitoare la Python. Digital Creations este unul din sponsorii PSF.

Toate lansările Python sunt Open Source.

Visual Python permite crearea, animarea, navigarea în jurul scenelor 3D, zooming, și interacțiunea cu obiectele 3D cu ajutorul mouse-ului.

IDLE este un editor interactiv, scris de Van Rossem și modificat de Scherer, care permite scrierea de programe, testarea și primirea de informații despre program.

VPython împreună cu IDLE este disponibil pe următoarele platforme: Windows, Linux, Unix, Macintosh (X11 on OSX), Macintosh (System 9).

Invocarea modulului Visual necesită următoarea declarație la începutul fișierului (cu extensia: ".py"):

```
from visual import *
```

Un comentariu în program începe cu "#"

```
# această linie este un comentariu
```

Obiectele de bază:

cylinder, arrow, cone, pyramid, sphere, ring, box, ellipsoid, curve, convex, label,

frame. faces. Additional Attributes: visible. frame. display. class. members: Convenient Defaults. Rotating an Object. Specifying Colors. Deleting an Object. Limiting the Animation Rate. Floating Division: 3 4 este 0. dar 3. 4. este 0.75 în Python.

Calculul vectorului:

mag. mag2. norm. cross. dot. rotate.

Funcții de reprezentare grafică:

Graph Plotting: gcurve. gdots etc.

Interactiunea cu ferestre. mouse și tastatură:

Controlling Windows: controlul afișării ferestrelor. luminii și poziția camerei:

Mouse Interactions:

Keyboard Interactions:

Controls: buttons. sliders. toggle switches. and pull-down menus.

În continuare este prezentat un exemplu simplu [27]. rezultatul fiind redat in figura 2.12.

```
from visual import *
redbox=box(pos=vector(4,2,3), size=(8,4,6),color=color.red)
greenball=sphere(pos=vector(4,7,3), radius=2, color=color.green)
```

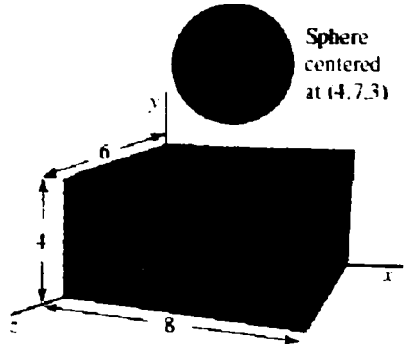


Figura 2.12: Exemplu VPython

Următorul exemplu este mai complex (*bounce2.py*) [27] iar rezultatul este redat in figura 2.13.

```
from visual import *

side = 4.0
```

```
thk    0.3
s2    2*side - thk
s3    2*side - thk
wallR    box (pos ( side, 0, 0),
              length=thk, height=s2, width=s3, color    color.red)
wallL = box (pos=(-side, 0, 0),
              length=thk, height=s2, width=s3, color    color.red)
wallB    box (pos=(0, -side, 0),
              length=s3, height=thk, width=s3, color    color.blue)
wallT    box (pos=(0, side, 0),
              length=s3, height=thk, width=s3, color    color.blue)
wallBK = box(pos=(0, 0, -side),
              length=s2, height=s2, width=thk, color    (0.7,0.7,0.7))
ball = sphere (color = color.green, radius    0.4)
ball.mass = 1.0
ball.p = vector (-0.15, -0.23, -0.27)
side = side - thk*0.5 - ball.radius
dt = 0.5
t=0.0
while 1:
    rate(100)
    t = t - dt
    ball.pos = ball.pos - ball.p*(dt ball.mass)
    if not (side    ball.x    -side):
        ball.p.x = -ball.p.x
    if not (side    ball.y    -side):
        ball.p.y = -ball.p.y
    if not (side    ball.z    -side):
        ball.p.z = -ball.p.z
```

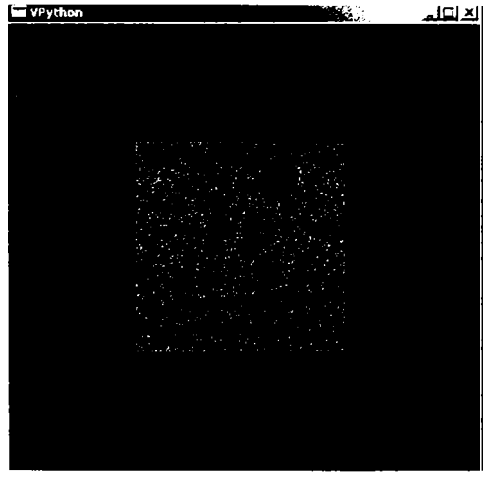


Figura 2.13: Exemplu bounce ball

2.6 Open GL

Prezentare generală

Biblioteca grafică OpenGL elaborată în limbajul C−− este una dintre cele mai utilizate biblioteci ce oferă un număr mare de funcții primitive grafice necesare în crearea aplicațiilor grafice interactive și care asigură o interfață independentă de platforma hardware. Scrisă în anul 1992, biblioteca grafică OpenGL implementează un standard de grafică multiplatformă având stabilitate, portabilitate și fiabilitate, cu o documentație tehnică adecvată, cu exemple de cod și informații utile accesibile gratuit pe Internet la adresa web: (`http://www.opengl.org`). OpenGL este intens folosit pentru o mare varietate de patforme hardware și pentru o mare varietate de aplicații grafice 2D și 3D, folosite în cele mai variate domenii, de la proiectarea asistată de calculator (CAD), la modelare-simulare și realitate virtuală.

Aplicațiile OpenGL se execută pe platforme foarte variate și sub cele mai cunoscute sisteme de operare: Windows 98 2000, Windows NT, Mac OS, Unix, Linux. De asemenea, funcțiile OpenGL se pot apela din limbajele de programare Ada, C, C−− și Java. OpenGL este o interfață software ce include câteva sute de proceduri și funcții care permit programatorului să specifice obiecte și operațiile asupra lor implicate în producerea imaginilor grafice de înaltă calitate, în special imagini ale corpurilor 3D.

Ce nu este OpenGL?

- nu este un mediu de programare;
- nu este o interfață grafică;
- nu este orientat pe obiecte.

Ce oferă OpenGL:

- primitive geometrice: puncte, linii și poligoane;
- primitive rastru;
- mod de lucru RGBA;
- modelarea și vizualizarea transformărilor de geometrie;

- eliminarea muchiilor ascunse (invizibile);
- antialiasing - tehnica netezirii muchiilor prin modificarea nuanței și luminozității pixelilor pentru ca muchiile să apară netede;
- aplicarea texturilor - aplicarea de texturi 2D pe obiect 3D;
- efecte speciale - fum, ceață și dizolvări;
- evaluatori polinomiali - pentru reprezentări NURBS;
- operații cu pixeli;
- plane șablon "stencil planes" - lucrul cu porțiuni de ecran;
- memorie tampon dublă "double-buffering" - lucrul simultan cu două rânduri de imagini, una care este afișată și una care este pregătită pentru afișare.

OpenGL este proiectat pentru utilizarea în aplicații tehnice 3D cum ar fi CAD (aplicații precum Solid Edge, Pro Engineer sau I-DEAS), animații de calitate (cum oferă Softimage) sau în aplicații de simulare vizuală (Design Review, GVS, ExoDIS).

Dat fiind că OpenGL prevede un set puternic, dar de nivel scăzut, de comenzi de redare a obiectelor, mai sunt folosite și alte biblioteci de nivel mai înalt care utilizează aceste comenzi și preiau o parte din sarcinile de programare grafică. Astfel de biblioteci sunt:

- biblioteca de funcții utilitare GLU (OpenGL Utility Library) permite definirea sistemelor de vizualizare, redarea suprafețelor curbe și alte funcții grafice;
- pentru fiecare sistem Windows există o extensie a bibliotecii OpenGL care asigură interfața cu sistemul respectiv: pentru Microsoft Windows, extensia WGL, pentru sisteme care folosesc X Window, extensia GLX, pentru sisteme IMB OS/2, extensia PGL;
- biblioteca de dezvoltare GLUT (OpenGL Utility Toolkit) este un sistem de dezvoltare independent de platformă, care ascunde dificultățile interfețelor de aplicații Windows, punând la dispoziție funcții pentru crearea și inițializarea ferestrelor și pentru execuția programelor grafice bazate pe biblioteca OpenGL.

Toate funcțiile bibliotecii OpenGL încep cu prefixul gl, funcțiile GLU încep cu prefixul glu, iar funcțiile GLUT încep cu prefixul glut.

2.7 Concluzii

În acest capitol, autorul a făcut un studiu cu privire la limbajele de realitate virtuală existente, referitor la fundamentele teoretice și la utilizarea acestora așa cum sunt prezentate în literatura de specialitate. În cadrul acestui studiu au fost prezentate limbajele de realitate virtuală ce permit dezvoltarea de aplicații *Open Source*. De asemenea, în partea de prezentare a limbajelor de realitate virtuală, în cadrul limbajelor VRML și Java & API-ul Java3D, sunt prezentate cîte un exemplu pentru fiecare limbaj de programare, dezvoltate și publicate în conferințe internaționale, de către autor, pe parcursul desfășurării activității de doctorat.

În urma acestui studiu, autorul a ales limbajul Java & API-ul Java3D în vederea dezvoltării unei aplicații *software* intitulată "VRforCAD", parte a sistemului VR-CAD. Aplicația respectivă este detaliată în capitolul "Contribuții la dezvoltarea și testarea unui software pentru sistemul VR-CAD".

Autorul subliniază faptul că amploarea acestui capitol se justifică prin locul central pe care il au limbajele de realitate virtuală în prezenta teză.

Capitolul 3

Contribuții la realizarea unui sistem VR - CAD

3.1 Introducere

În mod tradițional pentru a schimba modele între reprezentări B-Rep CAD și reprezentări de realitate virtuală, trebuie folosite fișiere. De exemplu, pentru a schimba modele între sisteme CAD și realitatea virtuală, sistemele CAD salvează mai întâi modelele B-Rep (Boundary-representation - modelare prin frontiere) ca fișiere. Apoi, după un proces de translatare necesar, fișierele CAD se pot importa în realitatea virtuală. Această metodă nu permite comunicarea și schimbarea de date în mod eficient și în timp real. De aceea sistemele actuale de realitate virtuală sunt folosite mai mult pentru verificare decât pentru modificarea unui proiect.

Se caută soluții de comunicare în timp real, fără a folosi fișiere, între sisteme de proiectare asistată de calculator (CAD) și sisteme de realitate virtuală. Din problemele majore ale acestor soluții se pot aminti: schimbul între sisteme VR și CAD în timp real a reprezentărilor geometrice de talie industrială este o problemă crucială și încărcarea procesorului ce poate cauza probleme mari la rularea în mod corespunzător a celor două sisteme.

Astfel, principiul schimbului de fișiere rămâne încă cea mai fiabilă soluție de a

integra sisteme VR în sistem CAD, deoarece soluția de mai sus necesită dezvoltarea unor plugin-uri (un plugin este un program care poate să se integreze într-un alt program (de bază) pentru a îndeplini funcții specifice) de comunicare separat pentru fiecare sistem CAD și sistem VR.

Figura 3.1 prezintă o schema bloc propusă de către autor [12] în referatul nr. 2, a unei aplicații ce realizează integrarea unui sistem de proiectare asistată de calculator cu un sistem de realitate virtuală. Comunicarea între cele două sisteme se realizează prin schimbul de fișiere. În [18], autorul stabilește care formate CAD corespund din punct de vedere al schimbului de date între sistemele CAD și sistemele de realitate virtuală.

Fișierele CAD sunt convertite în format VR și se crează scena 3D care este prelucrată pentru afișare și trimisă către sistemul de vizualizare (HMD, monitor stereo sau sistem de monitor cu ochelari stereo).

Schema bloc cuprinde trei module: modulul de vizualizare, interfața cu utilizatorul și modulul de interacțiune.

Pentru modulul de vizualizare, frecvența de repetare nu este necesar să fie mai mare de 30 Hz deoarece vederea utilizatorului nu percepe diferențe majore la frecvențe mai mari.

În modulul de interacțiune, frecvența de repetare trebuie să fie cât mai mare. O adevărată provocare este aceea de a se realiza sisteme integrate de proiectare asistată de calculator cu sisteme de realitate virtuală utilizând echipament haptic device, care pentru a menține duritatea suprafețelor virtuale, forța de răspuns, trebuie să crească detectarea coliziunilor și a forței de răspuns la 1000Hz.

În general un asemenea sistem include următorii pași pentru fiecare cadru redat:

- în momentul coliziunii se calculează adâncimea de penetrare;
- în funcție de adâncimea de penetrare, echipamentul haptic răspunde cu o anumită forță;
- se afișează pe ecranul sistemului (monitor, HMD) deformația obiectului.

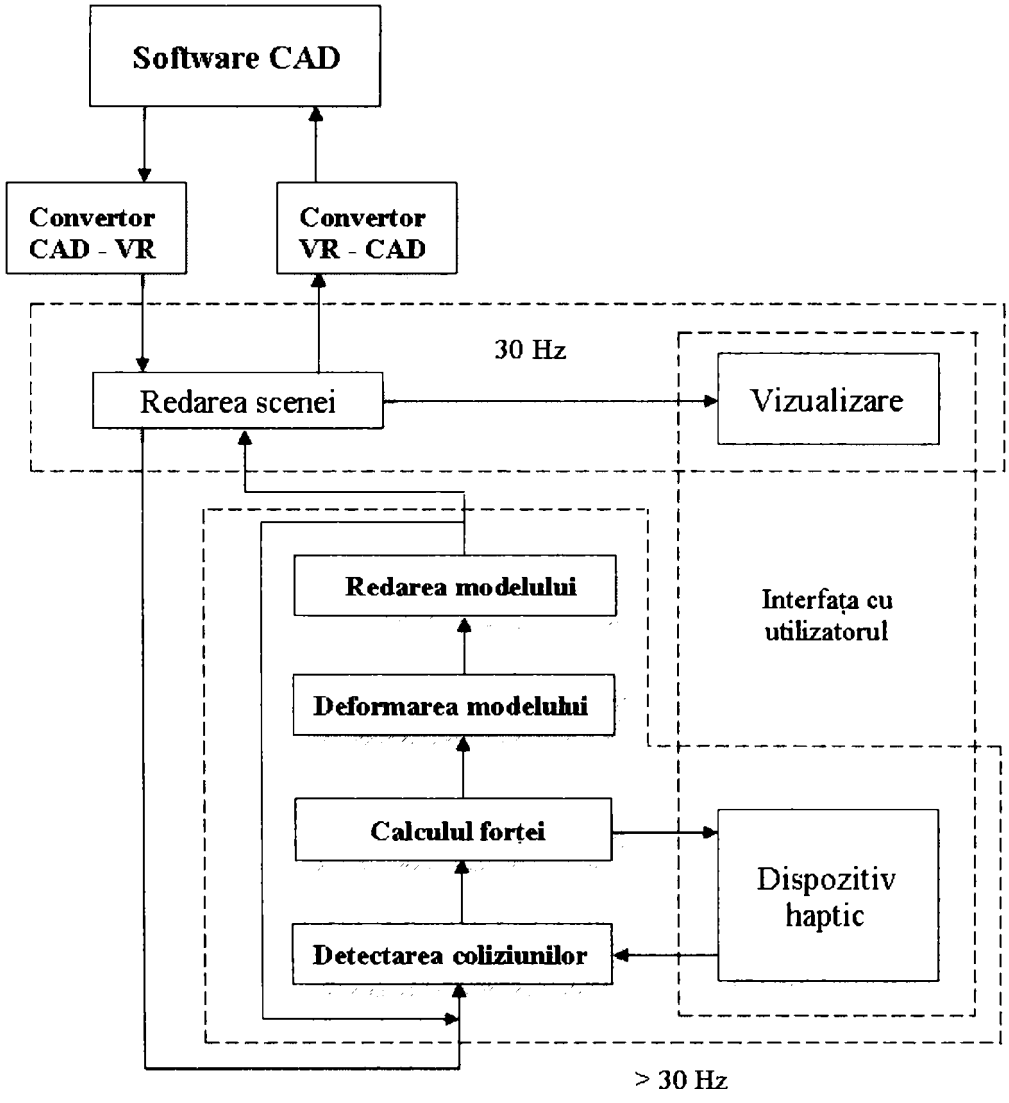


Figura 3.1: Schemă bloc sistem VR - CAD.

În continuare, autorul tezei a proiectat și realizat un sistem VR-CAD bazat pe schema bloc din figura 3.2. Sistemul este compus din două componente: *software* - aplicația "VRforCAD" și *hardware* - echipamentul "SphereDevice".

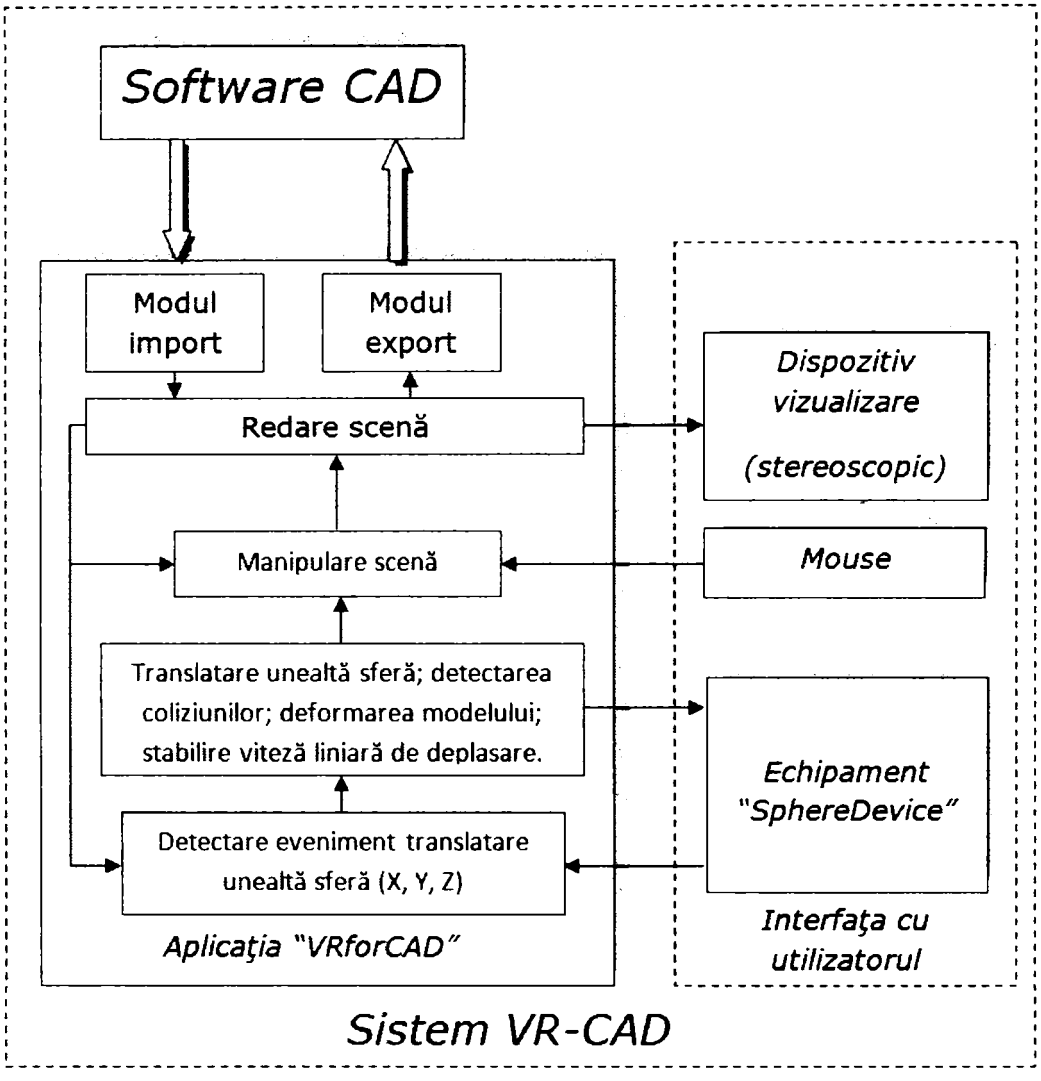


Figura 3.2: Schemă bloc sistem VR - CAD compus din aplicația "*VRforCAD*" și echipamentul "*SphereDevice*".

Schimbul de modele între sistemul CAD și sistemul VR se face utilzând formatele de fișier DXF și STL. Modulul de import citește fișierul CAD, extrage datele ce conțin geometria modelului și le convertește în instrucțiuni Java3D. Modulul de export descarcă datele din memoria calculatorului și le convertește în formatul de fișier CAD. Vizualizarea

scenei în aplicația "VRforCAD" se poate face atât cu un dispozitiv 2D (monitor, video proiector) cât și prin intermediul unor dispozitive stereoscopice (monitor + ochelari *shutter*, videoproiector stereo cum ar fi: InFocus). Mouse-ul permite operații: *rotate, pam, zoom* a întregii scene.

Deformarea suprafețelor în funcție de tipul materialui declarat, se face cu echipamentul "SphereDevice" (de tip *haptic*) prin translatarea unei unelte de formă sferă. În momentul coliziunii dintre sferă și modelul 3D, se determină coordonatele coliziunii după care aplicația decide noile coordonate ale vârfurilor afectați și parametrii de deplasare liniară a uneltei sferă din componența echipamentului. Aplicația este prezentată pe larg în capitolul 3, iar echipamentul în capitolul 4.

3.2 Compatibilitate CAD

Următorul tabel prezintă compatibilitatea între cele mai utilizate programe CAD și diverse formate de fișier ce se pot importa de către acestea.

Program CAD	Native	DWG, DXF	IGES	Para-solids	SAT	STL	VDA	VRML 1 or 2	XGL, ZGL
Autodesk AutoCAD	√	√			√	√			√
Autodesk Inventor	√		√		√	√			√
ArchiCAD		√							
Autodesk ADT		√							√
Autodesk MDT		√				√			√
CADKEY		√	√	√	√	√			
CATIA v4	√		√			√	√		
CATIA v5			√			√	√		
CoCreate Designer			√		√	√			
Cosmos			√	√	√				√
Form/Z		√	√		√	√		√	
I-DEAS			√			√	√	√	
Microstation		√	√	√	√	√		√	
PTC Pro/DESKTOP	√								
PTC Pro/Engineer	√		√			√			
Solid Edge	√		√	√	√	√			√
SolidWorks	√		√	√	√	√	√	√	√
Think3			√			√	√	√	
Unigraphics			√	√		√	√		

Din tabelul de mai sus, rezultă că cele mai utilizate formate de fișier pentru

schimburi sunt IGES și STL.

3.3 Formate de export modele 3D

Terminologia "model" reprezintă reflectarea informatică a corpului fizic (existent sau închipuit de proiectant). De exemplu, un reper este reprezentat informatic printr-un model matematic/geometric (ansamblu de elemente geometrice: linii, cercuri etc. între care programul creează anumite legături conform modelului matematic care stă la baza soft-ului) care reflectă forma reperului și un set de atribute alfanumerice atașate acestui model geometric care reflectă proprietățile sau comportamentul acestuia. Acest ansamblu (geometrie + atribute descriptive) reprezintă "modelul" reperului fizic. Modelul există așa numai cînd este compus în memoria calculatorului, din informația salvată în fișiere (fișiere tip .dwg, .dgn etc. și fișiere tip baze de date). Modelul nu înseamnă fișierele de pe disc; acestea sunt numai forma persistentă (pe disc sau pe alt suport de înregistrare a informației) a modelului. Calculatorul reface în memorie modelul reperului din informațiile organizate în fișiere, pe baza procedurilor din soft. Dacă aceste proceduri sunt propriii aplicației respective (adică nu sunt expuse într-o specificatie publică, accesibilă oricarui utilizator), atunci modelul nu se poate recompune corect din fișiere decît în aplicația care l-a creat. Acesta este cazul majorității programelor CAD actuale (de ex. AutoCAD) ale căror formate de fișiere (.dwg) nu au structură publicată oficial [8].

Formatul de fișier DXF

Formatul **.dxf** exportă o bază de date a sceneler prin poligoane din 3 sau 4 puncte utilizând entități 3DFACE sau POLYFACE MESH. Poligoanele cu 5 sau mai multe vârfuri, poligoane concave sau poligoane cu găuri sunt automat triangulate.

Deși sistemul este departe de a acoperi necesitățile sistemelor CAD majore, el nu poate fi totuși eliminat datorită popularității sale.

Formatul DXF a fost conceput inițial ca o a doua formă de reprezentare a entităților AutoCAD (în format ISO), dar cum AutoCAD-ul este foarte răspândit, s-a preferat pentru transport utilizarea directă a acestor fișiere.

Fișierul se bazează tot pe entități grafice și este foarte voluminos, fiecare instrucțiune

ocupând o linie, datele sunt codate ISO.

Fișierul DXF cuprinde patru părți: *Header* (pentru variabilele de sistem), *Tables* (pentru stiluri de linie și sisteme de coordonate), *Blocks* (pentru blocuri) și *Entities* (pentru definirea entităților).

Se poate construi un fișier DXF valid și numai cu secțiunea Entities. De exemplu, pentru un fișier al unui desen format dintr-o singură linie, secțiunea Entities are forma următoare:

```
ENTITIES0
LINE
5
40
330
1F
100
AcDbEntity
8
0
100
AcDbLine
10 Coordonata x a primului punct
1.0
20 Coordonata y a primului punct
1.0
30 Coordonata z a primului punct
0.0
11 Coordonata x a celui de-al doilea punct
5.0
21 Coordonata y a celui de-al doilea punct
5.0
31 Coordonata z a celui de-al doilea punct
0.0
0
ENDSEC
```

Formatul de fișier VRML

VRML este un format de fișier creat pentru descrierea scenelor 3D care sunt utilizate în aplicații de realitate virtuală interactive.

Caracteristici principale ale formatului VRML 1.0/Inventor v2:

- exportă date *mesh* cu *vertex normals* și *vertex texture coordinates*;
- lumini (punctiforme, spot și direcționale) și vedere în perspectivă;
- include imaginile de textură în fișierele VRML în format *raw* sau crează legături cu fișierele folosite pentru textură;
- convertire automată a formatului bitmap și modificarea automată a mărimii din / sau către cele mai populare formate de fișier 2d bitmap;
- include parametrii de material: *ambient color, diffuse color, specular color, emissive color, shininess* și *transparency*;
- segmentarea mesh-ului în multiple primitive mesh cu asignarea texturii corespunzătoare.

Caracteristici principale ale formatului VRML2:

- ierarhie pentru geometrie;
- camera și animații cu surse de lumină;
- culori ale vârfurilor pentru date de tip mesh;
- permite instanțierea geometriei, ceea ce reduce semnificativ mărimea fișierelor dacă o rețea sau un set de rețele sunt folosite de mai multe ori în același fișier.

Formatul de fișier IGES

Primul standard de interschimbabilitate a fost IGES (*Initial Graphics Exchange Specification*), dezvoltat în 1980 de marii producători de sisteme CAD din SUA. A fost conceput sub format ASCII/ISO pentru a putea fi cât mai portabil. Din acest motiv, fișierele sunt mult mai voluminoase decât cele originale.
Primele versiuni au fost destul de nefiabile, ținând cont doar de câteva mari sisteme CAD americane; versiunile ulterioare au fost însă completate.

IGES definește un model CAD reducându-l la listă de entități. Aceste entități sunt descrise într-un format neutru ASCII. Fiecare tip de entitate este asociat cu un număr, după cum urmează:

- Grupa 100

Entități geometrice. Exemple:

100 - arc circular,

102 - curbă compusă,

110 - linie,

116 - punct etc.

- Grupa 200

Entități de indicare. Exemple:

202 - dimensiuni unghiulare,

216 - dimensiuni liniare,

214 - leader etc.

- Grupa 300

Entități de structură. Exemple:

302 - definire asociativitate,

310 - definire font pentru text,

314 - definire culoare,

406 - proprietate,

600 - 699 - macro-uri etc.

Fișierul de format neutru are cinci secțiuni:

- Secțiunea START cuprinde informații cu privire la transferul însuși, care sunt date de utilizator chiar la export;
- Secțiunea GLOBAL cuprinde 24 de câmpuri cu parametrii necesari transferului, separate prin virgulă. Semnificațiile câmpurilor sunt următoarele:

  1, 2 - caractere de delimitare,

  3 - identificatorul sistemului original,

  4 - numele fișierului,

  5 - identificatorul soft-ului sursă (producător),

  6 - versiunea translatorului IGES,

  7 - 11 - precizia (Integer, Real etc.),

  12 - identificatorul receptorului,

  13 - scara spațiului model,

14 - unități,

15 - denumirea unităților,

16 - număr de grosimi utilizate pentru linii,

17 - grosimea maximă a liniilor,

18 - ora generării fișierului,

19 - distanța minimă,

20 - cea mai mare coordonată,

21, 22 - persoana și organizația care crează fișierul,

23 - versiune IGES,

24 - standard de desenare.

- Secțiunea DIRECTORY este generată de translatorul IGES și conține câte două linii pentru fiecare entitate. Fiecare linie la rândul ei conține 9 câmpuri formate din câte 8 caractere. Liniile conțin codul entității și pointerii către datele numerice din secțiunea următoare;

- Secțiunea PARAMETER DATA conține datele specifice pentru fiecare entitate cum ar fi valorile coordonatelor, text, numărul punctelor de control etc. Primul parametru al fiecărei linii identifică tipul entității. În funcție de aceasta, rezultă apoi semnificația datelor care urmează;

- Secțiunea TERMINATION marchează sfârșitul fișierului și conține subtotaluri ale înregistrărilor.

Fiecare linie are un identificator în coloanele 73 - 80. Primul caracter (col. 73) indică codul secțiunii: S - START, D - DIRECTORY etc.

Formatul de fișier STL

Un alt format neutru care se răspândește cu repeziciune este cel utilizat pentru dispozitivele de prototipare rapidă (*Rapid Prototyping*), în special cele bazate pe stereolitografie (formate *.STL), datorită faptului că fișierul respectiv poate descrie modele solide.

Formatul STL descrie numai geometria suprafețelor obiectelor 3D fără nici o

reprezentare a culorii, textură sau alte atribute ale modelelor CAD. Formatul STL specifică reprezentare ASCI și binară. Fișierele de tip binar sunt mai comune deoarece sunt mult mai compacte.

Un fișier STL descrie suprafețe triangularizate prin intermediul vîrfurilor (*vertices*) și a normalei (ordinea fiind dată de regula mâini drepte) triunghiurilor utilizând sistemul Cartezian de coordonate în trei dimensiuni.

Pentru formatul STL binar, există cel puțin două soluții de a adăuga informații de culoare, disponibile în pachetele *software* VisCAM și SolidView și în softul Materialise Magics.

Un fișier ASCII STL începe cu linia:

```
solid name
```

unde "name" este un string opțional. Fișierul continuă cu un număr nelimitat de triunghiuri, fiecare fiind reprezentat după cum urmează:

```
facet normal n1 n2 n3
  outer loop
    vertex v11 v12 v13
    vertex v21 v22 v23
    vertex v31 v32 v33
  endloop
endfacet
```

unde n1-n3 și v11-v33 sunt numere de tip *float*. Fișierul se încheie cu următoarea linie:

```
endsolid name
```

3.4 Schimbul de date între sisteme

Dacă sistemele CAD utilizează diverse tehnici de reprezentare a geometriilor, schimbul de date între sisteme se poate realiza doar dacă conversia între formatele de ieșire / intrare este posibilă. De exemplu, dacă un sistem CAD utilizează modele solide, aceste modele nu pot fi importate de către alt sistem ce utilizează modele wireframe sau desene 2D. Următorul tabel prezintă conversii posibile între CSG, B-Rep fațetat și B-rep avansat.

Sistem CAD import	Sistem CAD export		
	CSG	B-Rep fațetat	B-Rep avansat
CSG	precis	nu este posibil	nu este posibil
B-Rep fațetat	aproximativ	precis	aproximativ
B-Rep avansat	precis	precis	precis

3.5 Concluzii

În acest capitol, sunt prezentate noțiuni despre sistemele VR-CAD. De asemenea autorul tezei prezintă două scheme bloc, prima reprezintă o soluție posibilă de sistem VR-CAD, iar cea de-a doua este soluția realizată practic (sistem compus din două componente: *software* - reprezentat de aplicația "VRforCAD" și *hardware* - reprezentat de echipamentul "SphereDevice") în cadrul programului de doctorat.

În încheierea capitolului se face o trecere în revistă a celor mai utilizate formate de fișier pentru schimburi între sistemele CAD.

Atfel se poate afirma că în domeniul formatelor neutre, produsele *software* comerciale CAD nu sunt încă aliniate la un standard unic, în principal datorită intereselor comerciale divergente. Din acest motiv, majoritatea dintre ele includ ca facilități de transfer în format neutru mai multe opțiuni, dintre care în mod uzual se regăsesc în meniuri IGES, STEP, DXF, STL.

Conform celor prezentate în acest capitol, autorul tezei a ales să folosească următaorele două formate de fișier neutre: DXF și STL. Aceste două formate de fișier stau la baza modulului de import/export al aplicației "VRforCAD", prezentată pe larg în capitolul următor.

Capitolul 4

Contribuții la dezvoltarea și testarea unui software pentru sistemul VR-CAD

4.1 Introducere

În acest capitol sunt prezentate contribuțiile autorului acestei teze în domeniul *software*, contribuții concretizate prin dezvoltarea aplicației "VRforCAD". Pe parcursul capitolului, sunt prezentate secvențe de cod și algoritmi din structura aplicației.

Deoarece aplicația "VRforCAD" se încadrează în domeniul *Open Source*, la început de capitol este prezentată terminologia "Open Source", apoi o scurtă trecere în revistă a aplicațiilor *software open source* clasificată în două categorii: *software* VR și *software* CAD.

Capitolul se încheie cu o scurtă prezentare a domenilor de aplicabilitate ale aplicației "VRforCAD" (CAD, robotică, vizualizare și medicină).

4.2 Open Source

Open source [97, 37] descrie practica de a produce sau dezvolta anumite produse finite, permițând accesul utlizatorilor să acționeze liber asupra procesului de producție sau dezvoltare. Unii specialiști definesc "*open source*" ca un concept filozofic iar alții consideră că este o metodologie pragmatică.

Open source reprezintă dezvoltarea de programe software de către o comunitate, de către o companie sau de către o persoană și oferirea lor spre folosire sau îmbunătățire sub licența GPL.

Exemple de programe *open-source*: Azureus (client bit-torrent), Blender (grafică 3D), Gaim (client pentru mesagerie instant), o sumedenie de distribuții Linux, Mozilla Firefox (browser), OpenOffice.org (suita office), Apache (cel mai utilizat server web), Joomla, distribuții Latex, MySQL (motor de baze de date foarte popular) etc.

Software-ul liber e caracterizat de libertatea acordată utilizatorilor săi de a-l utiliza, copia, distribui, studia, modifica și îmbunătăți. Mai exact, e vorba de patru forme de libertate a utilizatorilor săi:

- Libertatea de a utiliza programul, în orice scop (libertatea 0).
- Libertatea de a studia modul de funcționare a programului, și de a-l adapta nevoilor proprii (libertatea 1). Accesul la codul-sursă este o precondiție pentru aceasta.
- Libertatea de a redistribui copii, în scopul ajutorării aproapelui tău (libertatea 2).
- Libertatea de a îmbunătăți programul, și de a pune îmbunătățirile la dispoziția publicului, în folosul întregii societăți (libertatea 3). Accesul la codul-sursă este o precondiție pentru aceasta.

Un program este *software liber* dacă întrunește toate aceste libertăți.

"*Software liber*" nu înseamnă "*non-comercial*". Un program liber trebuie să fie utilizabil în scop comercial, și disponibil pentru dezvoltare și distribuție comercială. Dezvoltarea comercială a software-ului liber nu mai este ceva neobișnuit; iar software-ul comercial liber este foarte important.

Discutând despre software-ul liber, este indicat a se evita termeni de genul "gra-

tuit" sau "pe gratis", întrucât acești termeni pot conduce la ideea că principala caracteristică a softului liber ar fi prețul, și nu libertatea sa.

4.3 Software VR Open Source

- Crystal Space: este un free LGPL (Lesser General Public License) și portabil motor 3Dscris în C++. Suportă: six degree of freedom, colored lighting, mipmapping, portals, mirrors, alpha transparency, reflective surfaces, 3D sprites, scripting (using Python or other languages), 8-bit, 16-bit, and 32-bit display support, Direct3D under Windows, Glide and OpenGL on Windows, Linux, OS/2, Macintosh, BeOS etc. Crystal Space este un proiect open source [24].

- DIVERSE: este un cross-platform, open source, API pentru dezvoltarea de aplicații de realitate virtuală care pot rula pe orice platformă. Versiunea curentă DIVERSE rulează pe Linux, IRIX, Mac OS X și Windows XP. Pentru a evita împiedicarea dezvoltării, DIVERSE este construit fără paradigma "center of the universe" ceea ce permite utilizarea acolo unde este necesar. Asta permite interacțiunea cu multe alte unelte cum ar fi: API, OpenGL, Open Scene Graph, SGI Open GL Performer și Coin [98].

- FastScript3D: permite a utiliza Java3D în grafica 3D. FastScript3D permite a utiliza JavaScript sau VBScript în paginile web împreună cu apleturile Java3D [66].

- Genesis 3D Engine: este un Gaming engine [32].

- OpenGL Performer: Este o puternică interfață de programare pentru creare simulări cu vizualizare real-time și alte aplicații profesionale orientate pe grafică 3D. Este disponibil pentru sistemele de operare: UNIX, Linux și Windows 2000/XP [76].

- MAVERIK: este un sistem pentru managementul display-ului și interactiunii în aplicațiile de realitate virtuală. Permite o realizare rapidă a mediilor virtuale. Este produs sub licență GNU General Public License [1].

- Mesa 3D Graphics Library: este o clona la OpenGL care permite portabilitate între multe platforme hardware [88].

- Open Scene Graph: este un open source high peformance 3D graphics toolkit, utilizat în aplicații ca: visual simulation, games, virtual reality, scientific visualization și modelling. Scris în Standard C++ și OpenGL, rulează pe toate platformele Windows, OSX, Linux, IRIX, Solaris și FreeBSD [31].

- OpenGL: High Performance 2D-3D Graphics [75].

- OpenSG: este un sistem care crează programe grafice real-time pentru aplicații de realitate virtuală. Este dezvoltat sub Open Source (LGPL). Rulează pe IRIX, Windows și Linux și este bazat pe OpenGL [30].

- PLIB: Un set de librării OpenSource (LGPL) care permit programatorilor să scrie jocuri și alte aplicații interactive real-time care rulează 100% pe următoarele platforme: UNIX (BSD, IRIX, Solaris, OS-X), MacOS-X, MacOS-9 [81].

- VR Juggler: este un proiect realizat de Dr. Carolina Cruz-Neira și o echipă de studenți de la Iowa State University's Virtual Reality Applications Center. Acest produs este open source [2].

- Virtual Rendering System: este o librărie grafică pentru construirea de aplicații 3D interactive. Este disponibilă cu o largă colecție de componente 3D care facilitează implementarea graficii 3D în aplicațiile 3D.
  Este disponibil pentru platformele: Linux, Unix, Windows și este distribui sub licență GNU Lesser General Public License [45].

- Visualization ToolKit (VTK): este un soft open source pentru 3D computer graphics, image processing și visualization. VTK constă în C++ class library și include Tcl/Tk, Java și Python. VTK este disponibil pentru orice Unix-based platform, PCs (Windows 98/ME/NT/2000/XP) și Mac OSX Jaguar or later [56].

- White_dune: este un low level VRML97 tool for Unix / Linux / MacOSX și MSWindows. Poate citi fișiere VRML97, afișa și permite userului să schimbe scenegraph/-fields. Are suport pentru stereoscopic via OpenGL "quadbuffer", pentru o varietate de echipamente de intrare 3D cum ar fi: spaceball, dialbox, joystick și Ascension Flock of Birds magnetic headtracker [80].

4.4 Software CAD Open Source

- CadSdt Lite - un produs CAD complet gratis, 255 nivele undo, formă redusă (fără "layot"-uri multiple) a produsului comercial CadStd Pro produs de firma Apperson &Daugters [53].

- CADVANCE - Produs CAD pentru Windows, produs de FIT (Furukawa Information Technology) Inc. SUA (CA) și care este distribuit gratis pe dischete. Firma FIT produce și programul comercial CADVANCE 2000 [34]. O licență educațională din acest produs se vinde cu 250 $ (sunt disponibile și licențe de campus). Produsul permite aplicații 2D și 3D, are un număr nelimitat de nivele undo, suportă 20 formate raster și DWG/DXF (translator Autocad R13).

- IntelliCAD - produs de CADopia este conceput ca un produs CAD adaptat cerințelor studenților, educatorilor, arhitecților, proiectanților, desenatorilor și inginerilor [10]. În afara produsului de bază, care este gratis, firma producătoare comercializează și un produs cu facilități avansate, dar la un preț rezonabil. Programul utilizează în mod nativ formatul DWG, permițând utilizatorilor să citească orice fisier produs cu Autocad, fără translatare. Produsul are comenzile compatibile cu cele de la Autocad (permite citirea fișierelor de meniuri și scripturi Autocad .MNU, .SCR), AutoLISP și ADS.

- Open CASCADE este cea mai bună alternativă industrială deschisă "Open Source" la nucleele de modelare 3D comerciale (proprietare) [64].
  Pachetul constă dintr-o biblioteca C++ de obiecte reutilizabile ce sunt disponibile în sursă. El este destinat producerii de aplicații grafice 3D, inclusiv din domeniile: CAD, CAE, CAM, AEC, GIS, reverse engineering, etc. Platformele suportate în prezent sunt Linux, Windows și Sun Solaris. Cu Open CASCADE se obține acces la codul sursă al diferitelor obiecte geometrice 3D, de la primitive volumetrice la suprafețe 3D avansate (NURBS, Beziers) precum și la mulți algoritmi avansați de modelare geometrică (intersections, projections, local and global properties of objects, Boolean operations, hidden line removal, fillets and chamfers, draft angles, graphic representation of 2D and 3D objects in an Open GL-based viewer, etc.). În plus, Open CASCADE citește și generează date în acord cu standardele: IGES

si STEP, asigurând comunicarea cu mediile actuale integrate de proiectare (Catia, Euclid). Mai mult, modulul Open CASCADE Application Framework permite dezvoltarea rapidă a aplicațiilor, oferind șabloane (template) de aplicații gata de utilizat, parametrizarea modelelor și posibilitatea de a atașa atribute negeometrice la geometrii. Utilizând pachetul Open CASCADE firma MAtra a dezvoltat pachetul CAD numit EUCLID. Cei 50 de membri ai echipei de dezvoltare sunt localizați la Matra Datavision's Corporate în France, în filialele din Rusia (Nijni-Novgorod) și Belarus (Minsk), dar operează și în SUA, Marea Britanie, Germania și Italia.

- SoftCAD 3D Lite - este un program de modelare 3D, editare, rendering și animație cu aplicații în special în arhitectură, design interior, peisaj, mobilier, construcții și grafică [79]. Permite generarea și editarea suprafețelor într-o manieră ușor de folosit și de învățat, annimații OpenGL și texte 3D. Obiectele 3D pot fi exportate în format DXF, sub formă de proiecții. Poate fi executat sub Windows 95/98 sau NT cu spațiu de disc liber de min. 50 Mb. Licența de utilizare are pretul de 199 $ (99$ pentru SoftCAD 2D), dar versiunea 1.16 Lite se distribuie gratis, prin Internet. Firma producătoare a donat licențe cu scop educațional institutelor de arhitectură din Romania și Ungaria.

- Solid Edge Origin - este un instrument CAD, forma limitată a pachetului complet de software Solid Edge, produs de firma UGS. El include modelarea 3D a părților componente, un sistem complet de desenare mecanică 2D (linii, arce, cote, etc.), importul și exportul datelor 2D și o instruire integrată. Modelarea solidelor 3D include proiecția automată 2D, parametrizarea și desenarea pieselor prismatice, ca elemente de bază pentru CAD 3D. Origin permite importul și exportul datelor CAD 2D în formatele DXF și DWG. Firma pune la dispoziție gratis inclusiv un pachet HTML de instruire asistată prin Internet - "Computer-Assisted Self Training (CAST) package" [78].

- Firma IMSI din SUA produce pachetul TurboCad, care oferă o soluție completă de proiectare oricui trebuie să creeze, editeze sau vizualizeze desene 2D sau 3D. Pachetul TurboCAD 2D este oferit gratis (încarcare pe web) [47]. Cu el se pot edita desene 2D care pot fi importate sau exportate în formate standard industriale,

pachetul fiind compatibil cu Autocad. IMSI a distribuit peste 1 200 000 licente.

- DeltaCad - program CAD shareware pentru Windows 95, 98, NT, 2000. Costul licentei v 4.0 este de $ 39.95 (comandă electronică), peste 100.000 licențe vândute [63]. Caracteristici: ușor de învățat, desenează puncte, linii, cercuri, elipse, arc, spline, texte, dimensiuni, dreptunghiuri, pătrate, triunghiuri, hașuri, include poze .BMP, exportă poze în editoare, permite crearea de simboluri personalizate, calculează lungimi și arii, conține un limbaj de programare de tip Basic pentru macrocomenzi , crează solide, suprafețe ascunse și secțiuni, are facilități multiple de zoom și editare grafică (move, copy, mirror, rotate, scale, change color, change line type, change cross-hatch pattern, change line weight, Undo, create a corner, radius), vine cu exemple de desene și bibliotecă de simboluri, citește și scrie fișiere .DXF compatibile cu alte programe CAD, crează listă de parti, etc.

- DesignWorkshop Lite dezvoltat de Artifice Inc. creează modele 3D, ce pot fi randate și vizualizate din diferite puncte, pornind de la schițe de prezentare (DXF), pentru construcții și arhitectura sau orice alt proiect spațial. Firma produce și o versiune "profesională", la un pret rezonabil [4].

- Minos - program freeware de modelarea solidelor realizat de francezul Regis Le Boite. El este ușor de utilizat și de învățat, extinzând facilitățile sistemelor 2D si 3D existente de tip wireframe. Pachetul permite proiectarea părților și ansamblelor 3D, pornind de la elemente simple ca: linii, curbe, cercuri, care pot fi apoi translatate, rotite sau plimbate de-a lungul unei curbe pentru a forma solide. Prin combinarea primitivelor standard: paralelipipezi, cilindri sau conuri se obțin forme complexe. Modele create pot fi vizualizate din orice unghi, cu viteza mare [69].

- Tlinea - este un program de desenare 2D din categoria shareware, dezvoltat în Spania. Importă fișiere .dxf și exportă formate ca . bmp. wmf si dxf [87].

- Varkon - este un sistem gratis de programe CAD sub Linux, dezvoltat de Microform AB Suedia, care permite desenarea, modelarea, vizualizarea precum și parametrizarea obiectelor pentru diferite aplicații CAD [89].

4.5 IDE

Un mediu de dezvoltare (*engl. development environment*, sau *integrated development environment* - "mediu integrat de dezvoltare") este un set de programe care ajută programatorul în scrierea de alte programe. Un mediu de dezvoltare combină toți pașii necesari creării unui program (exemplu: editarea codului sursă, compilarea, depanarea, testarea, generarea de documentație) într-un singur soft, care, de regulă, oferă o interfață cu utilizatorul grafică, prietenoasă.

Principalele componente ale unui mediu de dezvoltare sunt editorul de cod sursă și depanatorul. Mediile de dezvoltare apelează compilatoare sau interpretoare, care pot veni în același pachet cu mediul însuși, sau pot fi instalate separat de către programator. Printre facilitățile prezente în mediile de dezvoltare mai sofisticate se numără: exploratoare de cod sursă, sisteme de control al versiunilor, designere de interfețe grafice, sau unelte de ingineria programării (exemplu: generarea de diagrame UML) [96].

De obicei un mediu de dezvoltare este specific unui anumit limbaj de programare, însă există la ora actuală și medii de dezvoltare care pot lucra cu mai multe limbaje, de exemplu Eclipse sau Microsoft Visual Studio.

Eclipse este un mediu de dezvoltare *open-source* scris în Java, fapt ce îl face disponibil pentru majoritatea sistemelor de operare. Întreaga aplicație "VRforCAD" a fost dezvoltată utilizând Eclipse pe trei sisteme de operare diferite (FreeBSD 6.x, Windows XP, OpenSuse Linux 10.x), iar pentru partea de *User Interface* a aplicație s-a folosit *Eclipse Visual Editor*.

4.6 Aplicația "VRforCAD"

Aplicația "VRforCAD" (Virtual Reality for Computer Aided Design) dezvoltată de autorul acestei teze de doctorat, în limbajul de programare Java și API-ul Java3D, reprezintă partea software a respectivei teze.

Aplicația "VRforCAD" realizează o punte între sistemele de proiectare asistată de calculator și sistemele de realitate virtuală prin utilizarea modelelor CAD într-o aplicație de realitate virtuală. Aplicația vine în sprijinul utilizatorului cu un dezvoltat simț estetic dar mai puține cunoștințe tehnice necesare utilizării softurilor CAD.

Ideea principală a aplicației este aceea de a importa modele CAD în formatele de fișier dxf sau stl, modificarea acestora într-un mediu VR ce cuprinde vizualizare de tip stereoscopic și o mai bună interactivitate om-calculator cu ajutorul unui echipament de tip haptic intitulat "SphereDevice", echipament ce înlocuiește tradiționalul mouse, după care modelele sunt exportate în formatul CAD inițial. Schema bloc a aplicației este prezentată în Figura 4.1.

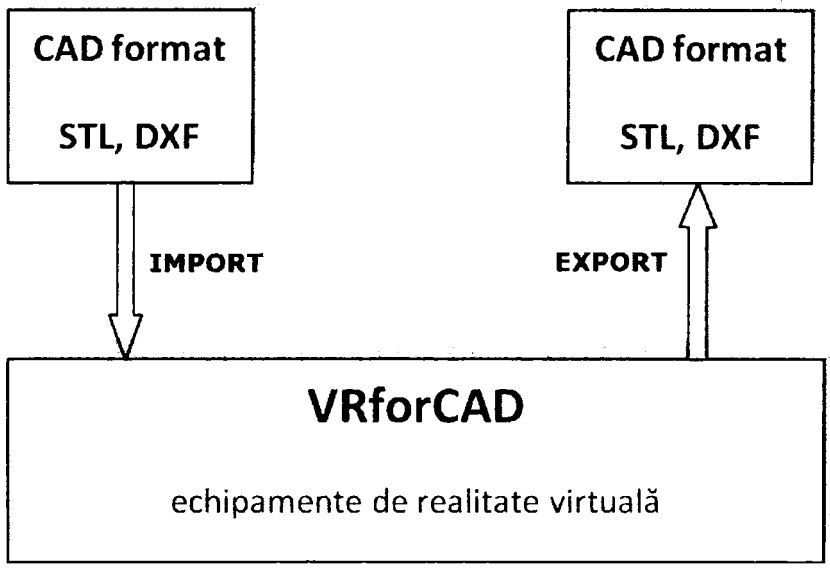


Figura 4.1: Schema bloc a aplicație "VRforCAD".

Din punct de vedere al diversității echipamentelor de realitate virtuală existente, aplicația "VRforCAD" a fost concepută modular ca în Figura 4.2. Modulele: vizualizare și manipulare au fost concepute în scopul extinderii numărului de echipamente VR, declararea aplicației sub licență GPL permițând dezvoltatorilor ce dispun de alte echipamente VR (încă neimplementate în aplicație) să adauge cod în scopul implementării respectivelor echipamente. De asemenea, numărul scăzut de aplicații - disponibile pentru sisteme de operare altele decât Windows - ce utilizează echipamente virtuale face aplicația "VRforCAD" o opțiune atractivă pentru universități în scopul extinderii funcționalității acesteia.

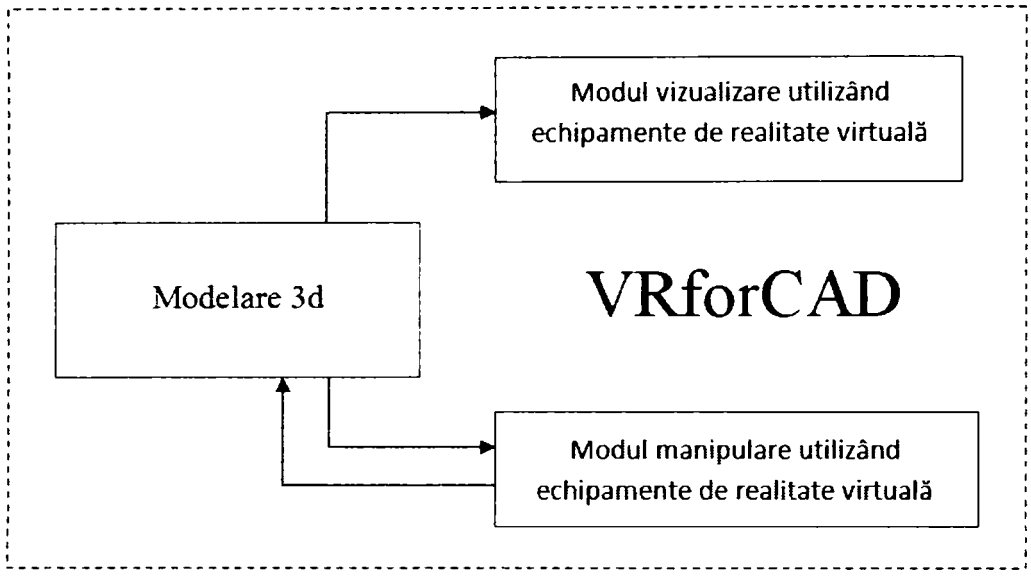


Figura 4.2: Schema modulară a aplicație "VRforCAD".

S-a ales limbajul de programare Java cu Java3D API din următoarele motive: un program Java compilat, corect scris, poate fi rulat fără modificări pe orice mașină (PC, PDA etc.) pe care e instalată o mașină virtuală Java (JVM- mediu în care se execută programele Java). Acest nivel de portabilitate (inexistent pentru limbaje mai vechi cum ar fi C) este posibil deoarece sursele Java sunt compilate într-un format standard numit cod de octeți (*byte-code*) care este intermediar între codul mașină (dependent de tipul computerului) și codul sursă.

Java 3D reprezintă o interfață de programare (API) de nivel înalt, orientată pe obiecte. Comparativ, API-urile 3D procedurale, DirectX sau OpenGL, au fost proiectate pentru a optimiza la maxim performanțele, oferind programatorilor un control total asupra aplicațiilor.

Figura 4.3 prezintă un *screenshot* al aplicației "VRforCAD".

4.7 Structura aplicației "VRforCAD"

În figura 4.5 este prezentată diagrama grafului scenei în aplicația "VRforCAD", iar în figura 4.4 este prezentată diagrama de vizualizare a aplicației.

Figura 4.3: *Screenshot* aplicație VRforCAD.

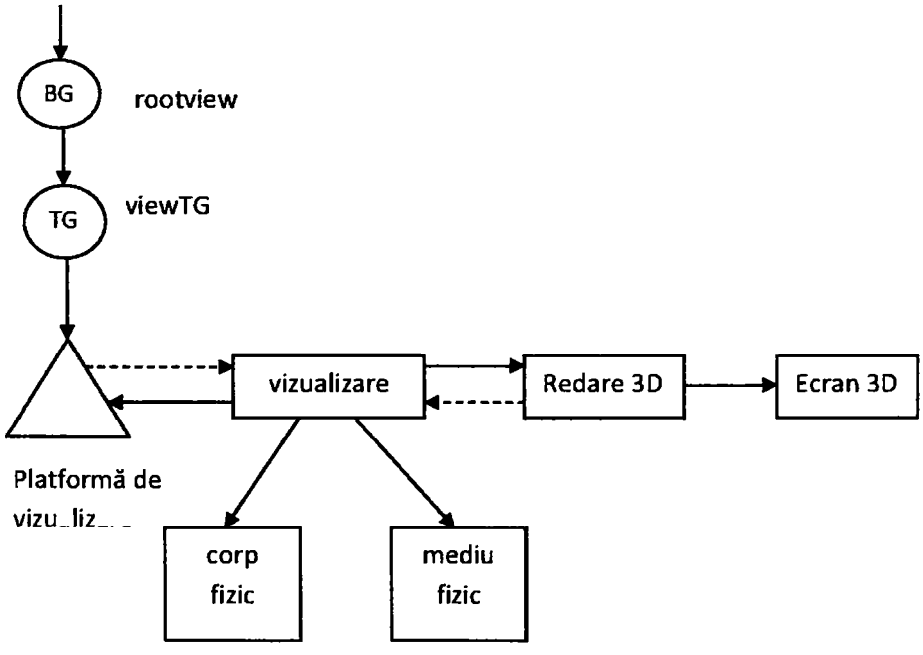


Figura 4.4: Diagrama grafului de vizualizare în aplicația "VRforCAD".

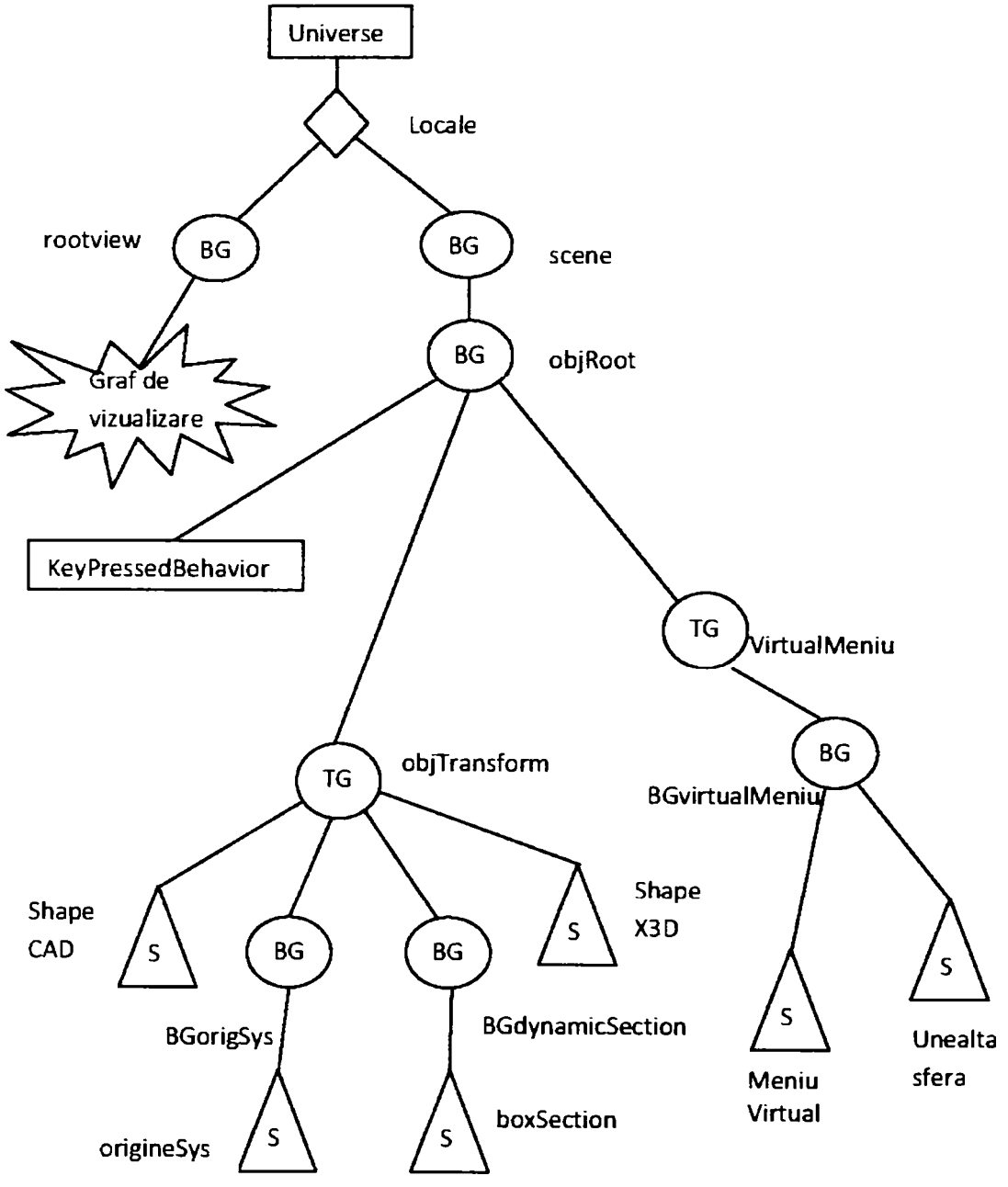


Figura 4.5: Diagrama grafului scenei in aplicatia "VRforCAD".

4.8 Modelare cu poligoane în aplicația "VRforCAD"

Modelarea poligonală, în care un obiect este reprezentat printr-o rețea de poligoane planare care aproximează suprafața de frontieră (boundary representation - B-rep), este forma "clasică" folosită în grafica pe calculator.

Motivele utilizării extinse a acestei forme de reprezentare sunt ușurința de modelare și posibilitatea de redare rapidă a imaginii obiectelor.

Pentru obiectele reprezentate poligonal s-au dezvoltat algoritmi de redare eficienți, care asigură calculul umbririi, eliminarea suprafețelor ascunse, texturare, *anti-aliasing*, frecvent implementate hardware în sistemele grafice. În reprezentarea poligonală, un obiect tridimensional este compus dintr-o colecție de fețe, fiecare față fiind o suprafață plană reprezentată printr-un poligon.

Un poligon este o regiune din plan marginită de o colecție finită de segmente de dreaptă care formează un circuit închis simplu.

Segmentele care mărginesc un poligon (linia poligonală) formează un contur închis (ciclu), deoarece segmentele sunt conectate capăt la capăt și ultimul segment conectează ultimul punct cu primul punct; ciclul este simplu deoarece segmentele neadiacente nu se intersectează.

Punctele v_i se numesc vârfurile poligonului (*vertices*); segmentele e_i se numesc muchii (sau laturi) ale poligonului. De remarcat ca un poligon conține n vârfuri și n muchii și că muchiile sunt orientate, astfel încât formează un ciclu (circuit închis). O astfel de orientare a segmentelor se numește orientare consecventă.

De regulă, se folosește sensul de parcurgere invers acelor de ceasornic: dacă se parcurg muchiile în sensul lor de definiție, interiorul poligonului este văzut întotdeauna în partea stângă (Figura 4.6).

În aplicația "VRforCAD" modelarea se face cu obiectele *TriangleArray* pentru reprezentarea geometriilor obținute din formatele de fișier vfc, dxf și stl, iar pentru reprezentarea geometriilor obținute din formatul de fișier x3d este folosit obiectul *IndexedTriangleArray*.

Următoarea secvență de cod prezintă modul de obținere a geometrie utilizând obiectul *TriangleArray*, iar pentru obiectul *IndexedTriangleArray* întreaga clasă ConvertX3D.java este listată în Anexa A.

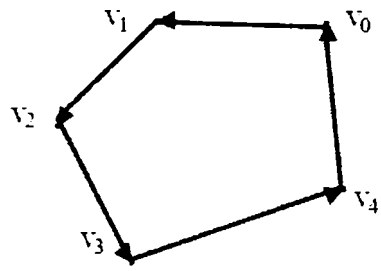


Figura 4.6: Segmentele liniei poligonale sunt orientate și nu se autointersectează.

```
...
int nrVertexi = TotalnrPuncte/3;

GeometryArray geom = new TriangleArray(nrVertexi,
      GeometryArray.COORDINATES |    GeometryArray.NORMALS |
      GeometryArray.BY_REFERENCE | GeometryArray.COLOR_3);

Shape3DCoordonates.coordonateLoad();

     geom.setCapability(GeometryArray.ALLOW_REF_DATA_READ);
  geom.setCapability(GeometryArray.ALLOW_REF_DATA_WRITE);
  geom.setCapability(GeometryArray.ALLOW_COUNT_READ);
  geom.setCapability(GeometryArray.ALLOW_COUNT_WRITE);
  geom.setCapabilityIsFrequent(GeometryArray.ALLOW_REF_DATA_WRITE);
  geom.setCapabilityIsFrequent(GeometryArray.ALLOW_REF_DATA_READ);
  geom.setCapabilityIsFrequent(GeometryArray.ALLOW_COUNT_WRITE);
  geom.setCapabilityIsFrequent(GeometryArray.ALLOW_COUNT_READ);
  geom.setCapability(GeometryArray.ALLOW_COLOR_READ);
  geom.setCapability(GeometryArray.ALLOW_COLOR_WRITE);
  geom.setCapability(Geometry.ALLOW_INTERSECT);    //for picking

     geom.setCoordRefFloat(Shape3DCoordonates.coordonate);
  geom.setColorRefFloat(Shape3DCoordonates.colors);
...
```

4.9 Deformare suprafețe în aplicația "VRforCAD"

Pentru modificarea suprafețelor modelelor 3D, pe baza index-ului poziției vertex-ului în array, obținut după operația de *picking*, se execută o căutare recursivă a vertexi-lor vecini, vertexi ce apoi sunt mutați de-a lungul direcției de deformare.

Adâncimea de căutare recursivă și deformarea sunt controlate de un factor de disipare. Figurile 4.7 și 4.8 prezintă diferența dintre utilizarea și neutilizarea acestei

metode pe un model 3D a unei componente de mașină (aripă față) [20]. Pentru o mai bună înțelegere la acest model s-a folosit modul de vizualizare wireframe.

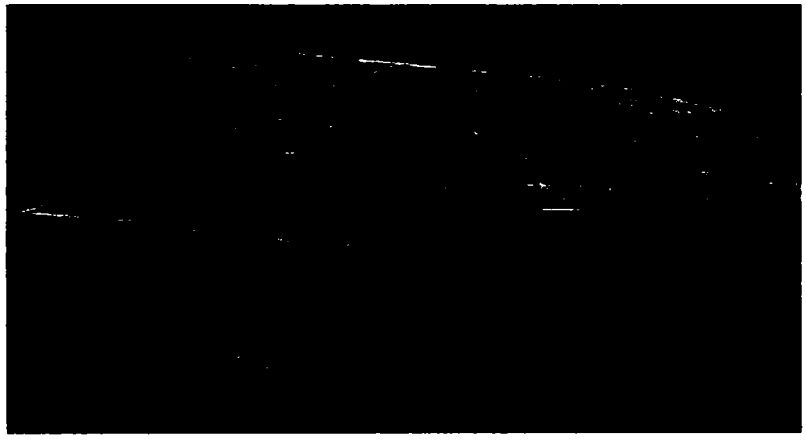


Figura 4.7: Deformare aripă de mașină fără deformare recursivă.

Figura 4.8: Deformare aripă de mașină cu deformare recursivă.

Deformarea suprafețelor se poate realiza prin două metode. Prima metodă, utilizând mouse-ul în combinație cu tasta "z" apăsată, se execută operația de picking descrisă mai sus. A doua metodă, prin deplasarea unei unelte sferă și utilizând detectarea coliziunilor așa cum este explicat mai jos. Deplasarea uneltei sferă, ce este accesată dintr-un meniu virtual, poate fi realizată prin intermediul mouse-ului în combinație cu tasta "a" apăsată (deplasarea pe axa z utilizând rotița de scroll) sau prin intermediul echipamentului "SphereDevice".

În meniul virtual din canvas-ul 3D există două sfere, prima pentru operații de deformare și a doua pentru operații de atribuire de culoare (*paint*) modelelor 3D.

4.10 Detectarea coliziunilor în aplicația "VRforCAD"

În [15], autorul prezintă metode de detectarea coliziunilor utilizate în realitate virtuală.

Detectarea coliziunilor în aplicația "VRforCAD" este realizată prin două metode. Prima prin extinderea clasei *Behavior* din API-ul Java3D așa cum este prezentat în secvența de cod de mai jos. Metoda *USE_GEOMETRY* detectează coliziuni la nivel de triunghi (la geometri cu mai mult de 4000 de triunghiuri, încărcarea procesorului devine o problemă). Marele dezavantaj al acestei metode este acela că nu se poate obține indexul în array a unuia din cei trei vectori al triunghiului cu care a intrat în coliziune sfera. Se obține eveniment de *ColisionEntry* și *CollisionExit*, dar fără posibilitatea obținerii locației exacte de coliziune. Metoda este utilă pentru reacția de atingere a modelului utilizând echipamentului "SphereDevice", dar fără posibilitatea de deformare a modelului.

```
...
public void initialize() {
     theCriteria = new WakeupCriterion[3];
     theCriteria[0] = new WakeupOnCollisionEntry(collidingShape,
               WakeupOnCollisionEntry.USE_GEOMETRY);
     theCriteria[1] = new WakeupOnCollisionExit(collidingShape,
               WakeupOnCollisionExit.USE_GEOMETRY);
     theCriteria[2] = new WakeupOnCollisionMovement(collidingShape,
               WakeupOnCollisionMovement.USE_GEOMETRY);
     oredCriteria = new WakeupOr(theCriteria);
     wakeupOn(oredCriteria);    }
public void processStimulus(Enumeration criteria) {
     WakeupCriterion theCriterion =
          (WakeupCriterion) criteria.nextElement();
     if (theCriterion instanceof WakeupOnCollisionEntry) {
```

A doua metodă, este realizată prin utilizarea obiectelor din categoria *picking* din API-ul Java3D. Următoarea clasă realizează *colision detection* utilizând obiectul *PickSegment*. Această metodă stă la baza detectării coliziunilor aplicației indiferent de dimensiunea modelelor. Totodată această clasa este una din clasele ce realizează deformarea suprafețelor în aplicația "VRforCAD".

```
/*
 * File        CollisionUsingPick.java
 *
 * Copyright (C) 2006-2007 Daniel Cioi <dan.cioi@vrforcad.org>
 *
 *                    www.vrforcad.org
 *
 *    This file is part of VRforCAD.
 *
 * VRforCAD is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * VRforCAD is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with VRforCAD.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

package org.vrforcad;

/**
 * This class detect the collision between sphere tool and model.
 *
 * @version 1.1
 * @author Daniel Cioi <dan.cioi@vrforcad.org>
 */

import javax.media.j3d.BranchGroup;
import javax.vecmath.Point3d;
import com.sun.j3d.utils.picking.PickIntersection;
import com.sun.j3d.utils.picking.PickResult;
import com.sun.j3d.utils.picking.PickTool;

public class CollisionUsingPick {

    private PickTool pickT;
    private PickResult pickR;
    private BranchGroup BGpickT;
    private Point3d closestVert;
    private int[] coordIndex;
    private int indiceTriangle;

    public CollisionUsingPick(BranchGroup objroot){
        BGpickT = objroot;
    }
```

```
public double collisionDetect(Point3d sphereCenter, Point3d sphereCenterF){

    double distanceCollisiune = 2;      // distance of detect collision
    pickT = new PickTool(BGpickT);
    pickT.setMode(PickTool.GEOMETRY);
    pickT.setShapeSegment(sphereCenter, sphereCenterF);
    pickR = pickT.pickClosest();

    if (pickR != null) {
        PickIntersection pi =
            pickR.getClosestIntersection(sphereCenterF);
        closestVert = pi.getClosestVertexCoordinates();

        if(closestVert!=null)
            distanceCollisiune = pi.getDistance();
        }

    return distanceCollisiune;
    }

public int[] getCoordsIndex(){      // return the indicies of
            // clossest vertex coordonate
    return coordIndex;
    }

public int getIndiceTriangle(){     //return the vector's indicies
            //in triangle (0, 1 or 2);
    return indiceTriangle;
    }

public Point3d getClosestVertex(){// return the closest vertex
            // in contact with Segment;
    return closestVert;
    }

}
```

Unealta sferă și emiterea segmentului de rază este exemplificată în figura 4.9. Astfel, când sfera este mutată pe una din cele șase direcții, pentru fiecare pas este emis un segment de rază în direcția de deplasare. Când segmentul de rază intersectează un vertex, returnează poziția vertexului în tabelul coordonate. Prin modificarea lungimii segmentului de rază se pot obține date despre detectarea coliziunii în momentul atingerii sferei sau mai departe de suprafața sferei (util pentru simularea suprafețelor poroase).

Deoarece geometria este neindexată, există vertecși comuni. Pentru a nu deplasa doar un triunghi la operația de deformare, se execută o cautare în tabelul coordonatelor

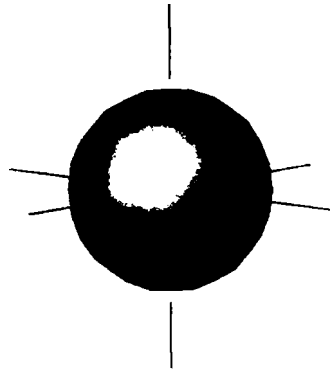


Figura 4.9: Unealta sferă și segmentele de rază.

pentru identificarea acestor vertecși. La deformare recursivă, apare o structură arbore de căutare ceea ce duce la căutări repetate (în funcție de nivelul de deformare recursivă) în același interval de timp. Aplicația se comportă corespunzător la deformarea modelelor cu până la 10.000 vertecși și 5 niveluri de deformare recursivă.

Din acest motiv, pentru modele mai mari de 10.000 vertecsi, se utilizează o metodă ce împarte scena 3D în unități de volum (cub) 5x5x5, de dimensiuni mai mici. Astfel, în loc de un singur tablou cu coordonatele vertecșilor geometriei, sunt 125 de tablouri, căutarea vertecșilor efectuându-se în tablouri de 125 ori mai mici. Localizarea tabloului în care se va efectua căutarea se face după coordonatele vertexului obținut la momentul coliziunii.

4.11 Vizualizarea Stereoscopică în aplicația "VRforCAD"

În [16] autorul prezintă o comparație între sistemele de vizualizare stereoscopică existente, justificând astfel de ce utilizatorul a ales folosirea sistemul de vizualizare activă cu ochelari *shutter*.

Testarea aplicației "VRforCAD" în modul stereoscopic s-a realizat pe următoarele plăci video: ATI FireGL 2, ATI FireGL X3-256, Nvidia Quadro FX 1400, Nvidia Quadro FX 3500; monitoare CRT: Compaq 22" și FujitsuSiemens 21"; ochelari *shutter wirelles*: Nuvision 60GX. Din punct de vedere al sistemului de operare, aplicația funcționează corect în modul stereoscpic pe sistemele Linux și Unix. La Windows XP efectul stereoscopic afectează și partea de SWING a alicației (nu doar Canvas-ul 3D),

efectul rezultat fiind butoane și meniuri dublate cu un offset.

Activarea modului stereoscopic se face din *CheckBox*-ul *Stereoscopic* din figura 4.10. Setarea distanței de *offset* în modul stereoscopic se face cu tastele: PageUP pentru valori pozitive și PageDown pentru valori negative. Valoarea default este 6.9 mm. iar valoarea modificată de către utilizator este afișată în *Panel South* (Figura 4.11)

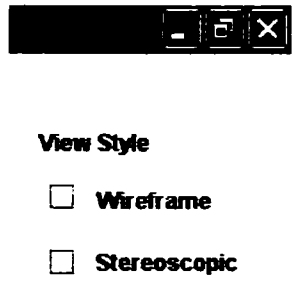


Figura 4.10: *CheckBox* pentru activare dezactivare mod stereoscopic.

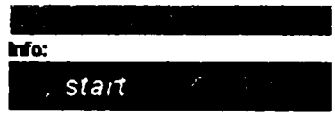


Figura 4.11: Afișarea valorii de *offset* în modul stereoscopic.

Următoarea secvență de cod din clasa *J3Dinterface.java* prezintă setările de vizualizare în aplicația "VRforCAD".

```
...
ViewPlatform viewPlatform = new ViewPlatform();
    double frustumfar = 40.0;
    view.setFieldOfView(Math.PI    2.0);
    view.setFrontClipPolicy(View.VIRTUAL_EYE);
    view.setBackClipPolicy(View.VIRTUAL_EYE);
    view.setFrontClipDistance(0.5);
    view.setBackClipDistance(frustumfar);
    view.setScreenScalePolicy(View.SCALE_EXPLICIT);
    view.addCanvas3D(workSpace);
    view.attachViewPlatform(viewPlatform);
    PhysicalBody physBody = new PhysicalBody();
    view.setPhysicalBody(physBody);
    PhysicalEnvironment physEnv = new PhysicalEnvironment();
    view.setPhysicalEnvironment(physEnv);
    eyesSeparation(eyesSeparationDistance);
```

4.12 Comunicare CAD - VR, VR - CAD în aplicația "VRforCAD"

Așa cum este prezentat în figura 4.1, comunicare între sistemele CAD și sistemul VR este asigurată de schimbul de fișiere. Modulul import/export al aplicației "VRforCAD", asigură importul formatelelor de fișier dxf, stl, vfc și x3d (x3d doar pentru modulul de vizualizare) și exportul formatelor de fișier dxf, stl, vfc. De asemenea, modelele pot fi convertite între cele trei formate de fișier.

În continuare este listat un exemplu de cod *convertDXFtoJ3D.java*, program ce citește fișiere în format dxf, după care le convertește în formatul de fișier vfc după care modelul 3D este încărcat într-o scenă 3D și redat pe dispozitivul de afișsare (monitor 2D sau sistem stereoscopic).

```
/*
 * File     : convertDXFtoJ3D.java
 *
 * Copyright (C) 2007 Daniel Cioi <dan.cioi@vrforcad.org>
 *
 *                    www.vrforcad.org
 *
 *   This file is part of VRforCAD.
 *
 * VRforCAD is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * VRforCAD is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with VRforCAD.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

package org.vrforcad;

/**
 * This class read the DXF file and convert it in VFC file format.
 * The VFC file format is an own format for VRforCAD application.
 *
 * @version 1.1
```

```
 * @author Daniel Cioi <dan.cioi@vrforcad.org>
 */

import java.io.*;
import java.util.Scanner;

public class convertDXFtoJ3D {

     public static void ReadCADFile(File DXF_File){

          Scanner dataDXF = null;
          PrintWriter result;  // output stream for writing data.
        String text;
        float[] array3DFACE = new float [24];

        try {  // Create the input stream.

            dataDXF = new Scanner(new BufferedReader(
                    new FileReader(DXF_File)));
         }
         catch (FileNotFoundException e) {
            System.out.println("Can't find file object.vfc!");
            return;             // End the program
         }

         try {  // Create the output stream.
             result = new PrintWriter(new FileWriter("object.vfc"));
             // .vfc from "vr for cad"
          }
          catch (IOException e) {
             System.out.println("Can't open file object.vfc!");
             System.out.println(e.toString());
             return;         // End the program.
          }

         try {
             while (dataDXF.hasNext()) {    // Read until end-of-file.

                if (dataDXF.hasNext()) {
                     text = dataDXF.next();

                     if (text.equals("AcDbFace")){ //scan MY3DLAYER
                     for (int ia = 0; ia < 24; ia++){
                          array3DFACE[ia] = dataDXF.nextFloat();
                                        }

                     for (int ia = 1; ia < 18; ia=ia+2){
                          result.println(array3DFACE[ia]);
                                        } //end for

                     result.println(array3DFACE[13]);
```

```
                    result.println(array3DFACE[15]);
                    result.println(array3DFACE[17]);

                    result.println(array3DFACE[19]);
                    result.println(array3DFACE[21]);
                    result.println(array3DFACE[23]);

                    result.println(array3DFACE[1]);
                         result.println(array3DFACE[3]);
                         result.println(array3DFACE[5]);

                    } //end if scan MY3DLAYER
                        }//end if hasNext
                 else {
                    dataDXF.next();
                 }
              }//end while
           }//end try
              finally {
                  dataDXF.close();
                  result.close();
                  // Finish by closing the files

                  J3Dinterface.geometrieNoua = true;
                  // the new geometry indicator
               }
            } // end of main()

} // end class
```

4.13 Formatul de fișier vfc al aplicației "VRforCAD"

Aplicația permite de asemenea salvarea modelelor într-un format de fișier propiu cu extensia vfc, structura datelor fiind de tip XML. În continuare este prezentată o scurtă descriere a ceea ce înseamnă XML [2].

Extensible Markup Language, abreviat XML, descrie o clasă de obiecte numite documente XML și descrie parțial comportamentul unor programe de computer care le procesează. XML este o aplicație profil sau o formă restrictivă a SGML-ului, *Standard Generalized Markup Language [ISO8879].* Prin construcție, documentele XML se conformează documentelor SGML.

Documentele XML sunt realizate din unități de stocare numite entități, ce conțin date parsate sau neparsate. Datele parsate sunt realizate din caractere, unele dintre ele

formând date caracter iar altele ca marcaje. Marcajele codifică o descriere a schemei de stocare a documentului și structura logică. XML furnizează un mecanism pentru a impune constrângeri asupra schemei de stocare și a structurii logice.

Un modul software numit procesor XML este utilizat pentru a citi documente XML și pentru a da acces la structura și conținutul lor.

Structura fișierului .vfc este următoarea:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<root>
<!--  .vfc file for "VRforCAD" Application-->
<ID system="RT56C788HJ">
<TRIANGLE T="nr. triunghi" V1="x z y" V2="x z y" V3="x z y"
ColorV1="r g b" ColorV2="r g b" ColorV3="r g b"
...
</root>
```

Astfel, fișierul de tip vfc permite salvarea atributelor de culoare pentru fiecare vertex al poligonului. Geometria modelelor este descrisă cu ajutorul poligoanelor de tip *triangle*. Pentru formatele CAD importate, normalele sunt generate de către aplicație, (dxf nu conține atribute de normale, iar stl include normală pentru fiecare triunghi nu pentru fiecare vertex al triunghiului așa cum este cerut de obiectul *GeometryInfo* din Java3D API).

O secvență dintr-un fișier cu extensia vfc este prezentat în cele ce urmează.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<root>
<!--  .vfc file for "VRforCAD" Application-->
<TRIANGLE ColorV1="0.6 0.6 0.6" ColorV2="0.6 0.6 0.6"
ColorV3="0.6 0.6 0.6" T="1" V1="-2.25 3.11 -0.35"
V2="-2.27 3.23 -0.5" V3="-2.17 3.07 -0.52"/>
<TRIANGLE ColorV1="0.6 0.6 0.6" ColorV2="0.6 0.6 0.6"
ColorV3="0.6 0.6 0.6" T="2" V1="-2.25 3.11 -0.35"
V2="-2.34 3.2 -0.27" V3="-2.27 3.23 -0.5"/>
...
</root>
```

4.14 Păstrarea setărilor modificate de către utilizator în aplicația "VRforCAD"

Setările aplicației ce pot fi modificate de către utilizator sunt salvate în fișierul cfg.xml. Strunctura datelor din fișier este de tip XML așa cum se poate observa mai jos.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<root>
<!--Config file for "VRforCAD" Application-->
<Background B="1.0" G="0.73" R="0.45"/>
<SphereDevice SDDX="1.5" SDDY="1.5" SDDZ="1.5"
SDLDX="40.0" SDLDY="40.0" SDLDZ="40.0"
SDWSXm="-10.0" SDWSXp="10.0"
SDWSYm="-10.0" SDWSYp="10.0"
SDWSZm="-5.0" SDWSZp="10.0"/>
<TypeParallelPort offsetValue="0"/>
<Database databaseName="cadmodels" password="cadmodelspass"
url="localhost" userName="cadmodels"/>
<UserAccount UserAbout="first user" UserName="danny"
UserPassword="qUqP5cyxm6YcTAhz05Hph5gvu9M="/>
...
</root>
```

În continuare este prezentată clasa java care citește și scrie în fișierul cfg.xml. Este evidențiată secvența de cod ce salvează setările de culoare a fundalului (*Background*) scenei 3D.

```
/*
 * File     : IniCfgFile.java
 *
 * Copyright (C) 2006-2007 Daniel Cioi <dan.cioi@vrforcad.org>
 *
 *                    www.vrforcad.org
 *
 *   This file is part of VRforCAD.
 *
 * VRforCAD is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * VRforCAD is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
```

```
 * along with VRforCAD.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

package org.vrforcad;

/**
 * This class read and write the cfg.xml file.
 * The cfg.xml file keep the settings made by user.
 *
 * @version 1.1
 * @author Daniel Cioi <dan.cioi@vrforcad.org>
 */

import java.io.File;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.Result;
import javax.xml.transform.Source;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Comment;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

public class IniCfgFile {

static DocumentBuilderFactory factory =
                    DocumentBuilderFactory.newInstance();
static DocumentBuilder docCfg;

static String BackgroundR = "0.0";
static String BackgroundG = "0.0";
static String BackgroundB = "0.0";
...

public IniCfgFile(){          }

public static void SetDefaultSetting(){
          BackgroundR = "0.4";
          BackgroundG = "0.4";
          BackgroundB = "0.4";


     }

public static void WriteFile(){
     try{
```

```
        //Get the DocumentBuilder
            DocumentBuilder docBuilder = factory.newDocumentBuilder();
            Document doc = docBuilder.newDocument();

            //create the root element
            Element root = doc.createElement("root");
            doc.appendChild(root);

            //create a comment
            Comment comment =
            doc.createComment("Config file for \"VRforCAD\" Application");
            //add in the root element
            root.appendChild(comment);

            //create child element
            Element childElement = doc.createElement("Background");
            //Add the atribute to the child
            childElement.setAttribute("R",BackgroundR);
            childElement.setAttribute("G",BackgroundG);
            childElement.setAttribute("B",BackgroundB);
            root.appendChild(childElement);



            TransformerFactory transFactory =
            TransformerFactory.newInstance();
            Transformer aTransformer = transFactory.newTransformer();

            Source src = new DOMSource(doc);
            //Result dest = new StreamResult(System.out);
            Result dest = new StreamResult(new File("cfg.xml"));
            aTransformer.transform(src, dest);
        }

        catch(Exception e){
            System.out.println(e.getMessage());
            }
        }

public static void ReadData(){
        try {
            docCfg = factory.newDocumentBuilder();
            Document document = docCfg.parse("cfg.xml");

            NodeList elementRead =
            document.getElementsByTagName("Background");
            for (int nr = 0; nr < elementRead.getLength(); nr++) {
                BackgroundR =
                ((Element)elementRead.item(nr)).getAttribute("R");
                BackgroundG =
                ((Element)elementRead.item(nr)).getAttribute("G");
```

```
                BackgroundB =
                ((Element)elementRead.item(nr)).getAttribute("B");
                    }

            }
        catch (Exception e) {
            e.printStackTrace();
            }
    }
}
```

4.15 Utilizarea unei server de bază de date pentru stocarea fișierelor CAD

În vederea stocării fișierelor CAD online pe un server de bază de date a fost dezvoltat un modul inclus în aplicația "VRforCAD" (Figura 4.12). Ca server de bază de date s-a ales MySQL (un sistem de gestiune a bazelor de date relațional, produs de compania suedeză MySQL AB și distribuit sub Licență Publică Generală GNU, fiind cel mai popular SGBD open-source la ora actuală).

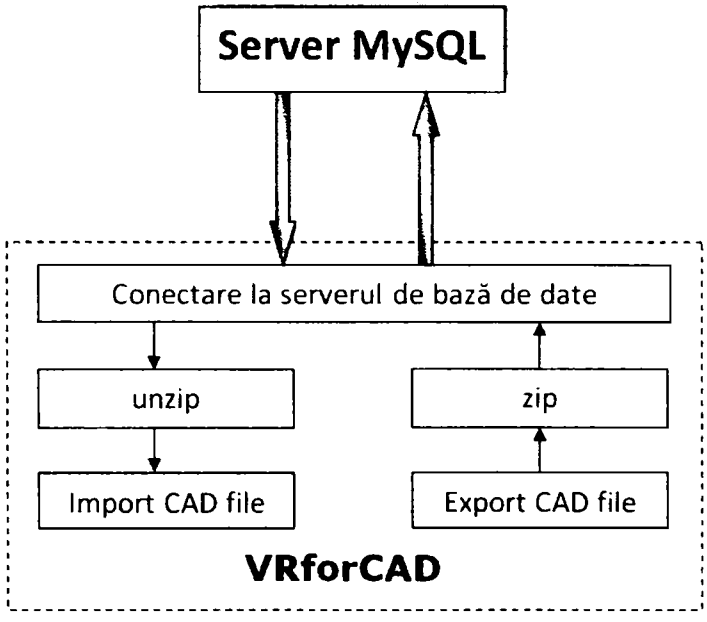
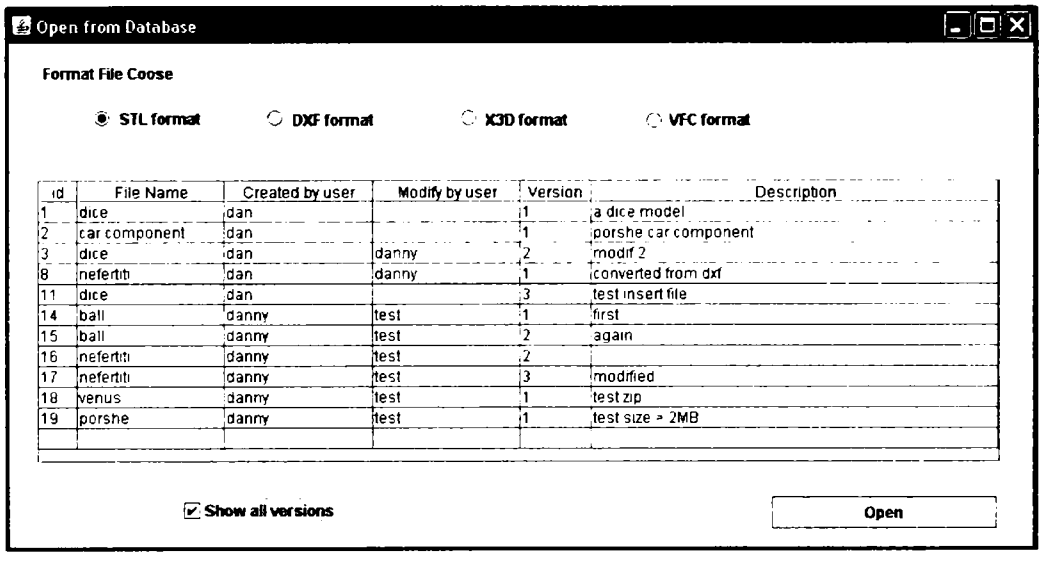


Figura 4.12: Schemă bloc - Import/Export modele CAD utilizând un server MySQL.

Astfel, utilizatorul poate salva modele CAD local pe hard disk sau online utilizând un astfel de server. Utilizând optiunea online, se elimină problemele apărute de așezarea geografică a utilizatorilor. Un alt avantaj este acela că în baza de date fișierele nu sunt rescrise (toate fișierele sunt păstrate diferența facându-se prin indexarea numărului de versiune a respectivelor fișiere). De asemenea corespunzător fiecărui fișier se salvează numele utilizatorului care a creat fișierul și numele utilizatorilor care modifică respectivul fișier. Pentru un transfer cât mai rapid al fișierelor, acestea sunt comprimate utilizând arhive în format zip.

Pentru interfața cu utilizatorul au fost adăugate următoarele două dialoguri prezentate în: figura 4.13 pentru importul modelelor CAD și figura 4.14 pentru salvarea modelelor CAD.



id	File Name	Created by user	Modify by user	Version	Description
1	dice	dan		1	a dice model
2	car component	dan		1	porshe car component
3	dice	dan	danny	2	modif 2
8	nefertiti	dan	danny	1	converted from dxf
11	dice	dan		3	test insert file
14	ball	danny	test	1	first
15	ball	danny	test	2	again
16	nefertiti	danny	test	2	
17	nefertiti	danny	test	3	modified
18	venus	danny	test	1	test zip
19	porshe	danny	test	1	test size > 2MB

Figura 4.13: Dialog - *Open from Database*

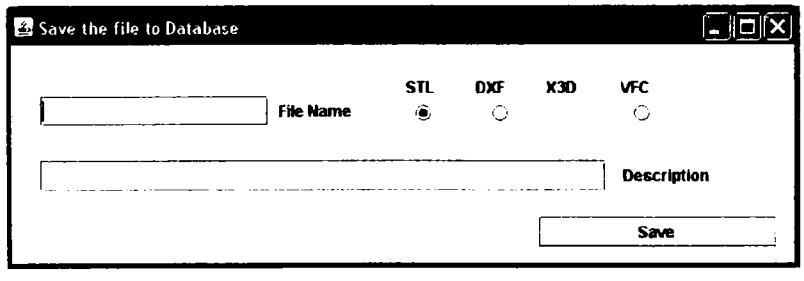


Figura 4.14: Dialog - *Save to Database*

Utilitatea unei server de bază de date pentru stocarea fișierelor CAD derivă din:

- unicitatea informației - clienții renunță la replici ale fișierelor;
- administrarea și accesul clienților controlat;
- accesibilitate permanentă și controlată;
- posibilități de securizate a informației prin implementarea sistemelor de securizare IT.

4.16 Realizarea unui mediu de lucru colaborativ

În cele ce urmează este prezentată o soluție originală de integrare a aplicației "VRforCAD" într-un mediu de lucru colaborativ. În acest scop a fost dezvoltată aplicația server (aplicație Java stand-alone) "VRforCAD CWES" (CWES - *Collaborative Work Environment Server*), aplicație care împreună cu VRforCAD permite manipularea modelelor 3D de către mai mulți utilizatori din locații diferite folosind arhitecturi informatice de tip client - server peste rețele interconectate. Astfel, participanții în mod colectiv pot vizualiza, manipula și modifica un model CAD concomitent.

Schema bloc a întregului sistem format din "VRforCAD CWES" instalat pe un server Linux sau Unix și n clienți ce utilizează aplicația "VRforCAD" este prezentată în figura 4.15.

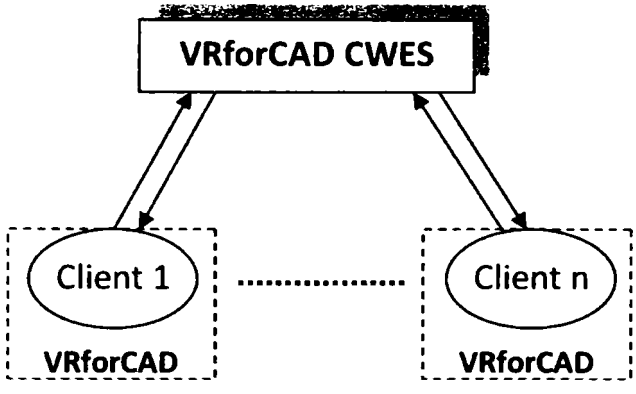


Figura 4.15: Schema bloc a arhitecturi de tip client - server

Aplicația server "VRforCAD CWES" este conectată la un server de bază de date (MySQL) ce conține o bază de date cu numele dbcwe și 3 tabele. Aceste tabele conțin următoarele coloane:

1. tabela *users* - '*id*' int, '*username*' varchar, '*password*' varchar, '*description*' varchar;
2. tabela *files* - '*id*' int, '*created by user*' varchar, '*name file*' varchar, '*content file*' longblob, '*users allowed*' varchar;
3. tabela *messages* - '*id*' int, '*user send*' varchar, '*user receive*' varchar, '*message*' varchar.

Pentru fiecare model 3D care este inițiat în mediul de lucru colaborativ, se crează câte un tabel cu numele fișierului obținut din tabelul *files*, coloana *name file*. Tabelul conține coloanele: '*id*' int, '*username*' varchar, '*operation*' varchar.

Figura 4.16 ilustrează tabelele din baza de date dbcwe.

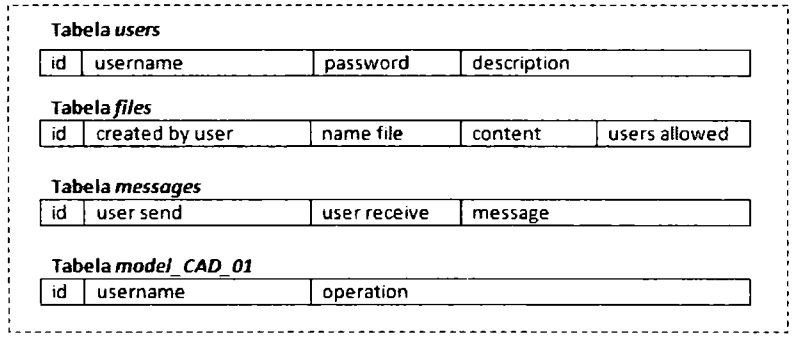


Figura 4.16: Tabele din baza de date: dbcwe

Comunicarea între client și server se realizează folosind socket-uri. Un socket furnizează facilități pentru crearea de fluxuri de intrare și ieșire, care permit datelor să fie schimbate între client și server.

Din motive de securitate, parola userului de conectare la serverul "VRforCAD CWES", este criptată folosind algoritmul de criptare SHA (*Secure Hash Algoritm*).

Conținutul tabelelor din baza de date dbcwe și scopul lor este descris în cele ce urmează.

Tabela *users* conține date despre utilizatori: nume de utilizator, parolă și o scurtă desciere a utilizatorului. Tabela *files* conține date despre fișierul CAD cum ar

fi: utilizatorul care inițiază mediul de lucru colaborativ, utilizatorii care au permisiunea de a accesa respectivul mediu colaborativ, inclusiv conținutul fișierului în câmpul de tip *longblob* (blob - *binary large object* permite stocarea datelor în format binar). Tabela *messages* asigură comunicarea sub formă de mesaje între clienții conectați la server. Tabelele cu numele fișierelor CAD conțin date de tipul: numele utilizatorului și comanda (într-un format definit) ce a avut ca acțiune modificarea modelului 3D.

Figura 4.17, prezintă schema bloc a clientului care inițiază modelul CAD în mediul de lucru colaborativ, iar figura 4.18 prezintă schema bloc a clienților care participă la respectivul mediu. După cum se poate observa din figurile 4.17 și 4.18, în urma stabiliri conexiunii cu serverul și validarea utilizatorului, clientul 1 (cel care inițiază modelul CAD) trimite fișierul la server, iar restul clienților încarcă respectivul fișier de pe server, restul operațiilor se execută în buclă. Valoarea din variabila statică ID.old este comparată cu ultima valoare a câmpului ID din tabela "nume fișier". Dacă ID.old < ID se citește câmpul *operation* și se execută modificarea modelului 3D în funcție de comandă. Dacă respectivul user execută o modificare a modelului 3D, comanda (formatată) și numele utilizatoului sunt trimise către server pentru a fi salvate în tabela "nume fișier", id-ul fiind automat indexat. Câmpul *username* este util pentru funcția *undo* și *history*.

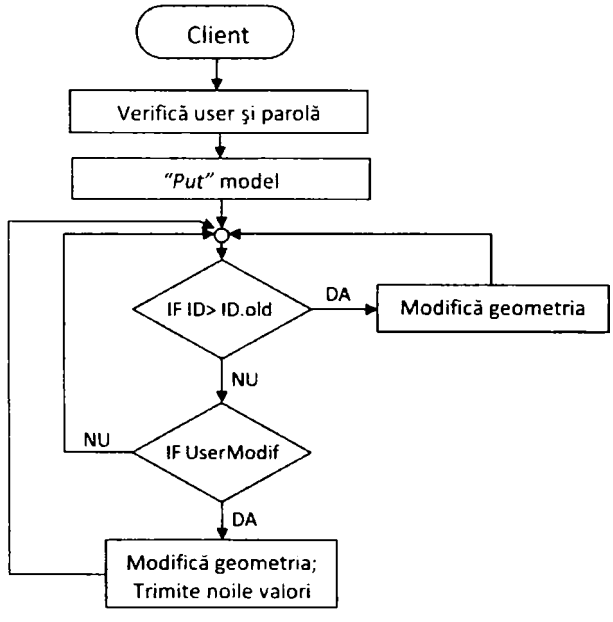


Figura 4.17: Schema bloc client - CWES

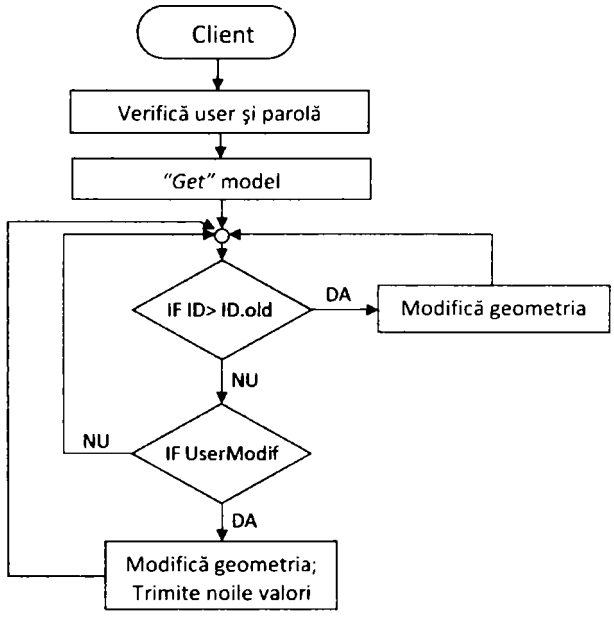


Figura 4.18: Schema bloc client - CWES

4.17 Interfața cu utilizatorul în aplicația "VRforCAD"

Interfața cu utilizatorul (*User Interface*), reprezintă totalitatea mecanismelor care permit interacțiunea dintre o aplicație și utilizatorii ei. O particularitate a interfeței cu utilizatorul o reprezintă interfața grafică - GUI (*Graphic User Interface*), care se referă strict la comunicarea vizuală dintre utilizator și aplicație.

În cadrul aplicației "VRforCAD", pentru o gestionare cât mai ușoară și eficentă, autorul a considerat necesar proiectarea unei interfețe grafice accesibilă majorității utilizatorilor (care, în general, nu posedă cunoștințe avansate de operare a calculatorului). Interfața realizată a fost astfel concepută, încât toate operațiile pe care utilizatorul le poate efectua la un moment dat sunt disponibile direct pe ecran.

Interfața grafică cu utilizatorul pentru aplicația "VRforCAD" a fost dezvoltată utilizând biblioteca SWING.

Biblioteca SWING este partea JFC (*Java Foundation Classes*) care oferă componentele necesare programatorilor pentru crearea de interfețe grafice moderne și complexe aplicațiilor Java. SWING aduce îmbunătățiri calitative față de mai vechiul pachet

AWT (*Abstract Window Toolkit*) folosit în trecut pentru realizarea interfețelor grafice, completându-i neajunsurile.

Diferența majoră dintre SWING și AWT este aceea că, spre deosebire de AWT, componentele SWING sunt scrise 100% în Java având ca bază API-ul JDK 1.1, neconținând cod nativ (dependent de sistemul de operare). Această înseamnă că butoanele SWING, spre exemplu, vor arăta și se vor comporta identic pe platformele Macintosh, Solaris, Linux, sau Windows. Spre deosebire, butoanele AWT luau mereu aspectul platformei pe care rula aplicația. De asemenea, prin această independență față de platformă se obține o îmbunătățire a vitezei de execuție a aplicațiilor având interfețe realizate în SWING.

În cadrul interfeței cu utilizatorul a aplicației "VRforCAD", autorul a creat mai multe meniuri, fiecare conținând mai multe funcții de operare asupra scenei 3D.

Meniul *File* (Figura 4.19) cuprinde următoarele opțiuni considerate de autor ca fiind necesare pentru gestionarea eficientă a fișierelor:

- *Open* - încărcare fișier în formatul vfc;
- *Save* - salvare fișier în formatul vfc;
- *Save As* - salvare fișier în formatul vfc cu o altă denumire;
- *Import CAD file* - importă fișiere în formatele: dxf, stl și x3d. Extensia fișierului este detectată automat de aplicație;
- *Export CAD file* - exportă modelul în formatle: dxf și stl.

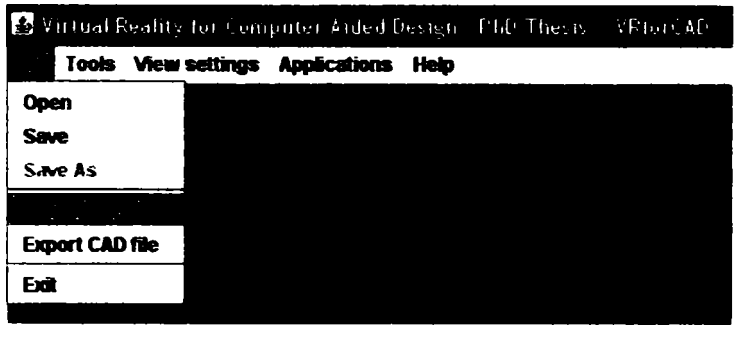


Figura 4.19: Meniul *File*.

Meniul View permite utilizatorului să folosească modul de vizualizare *wire-frame*, respectiv *stereosopic* a scenei 3D virtuale. Accesarea acestor optiuni se face prin activarea componentelor *CheckBox* disponibile in panoul (*Panel*) din dreapta interfetei (Figura 4.20).

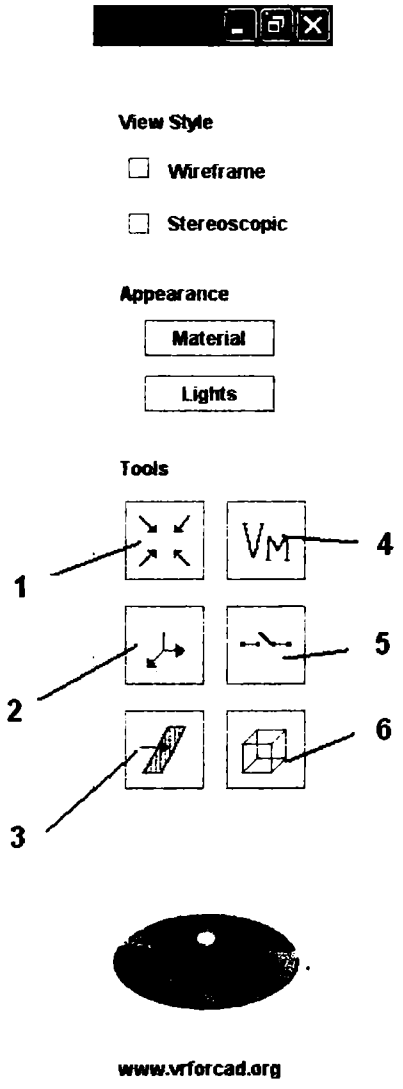


Figura 4.20: Panel dreapta.

Pentru accesarea opțiunilor existente în cadrul meniului *Tools* sunt folosite componente *JButton*. Pentru a descrie funcțiile butoanelor din figura 4.20, fiecărui buton i s-a asociat un număr:

- activarea butonului 1 permite utilizatorului aducerea în prim plan a modelului;
- activarea butonului 2 atașează sistemul de coordonate al scenei virtuale;
- activarea butonului 3 deschide fereastra de dialog *Dynamic Section* (4.21) ce permite utilizatorului alegerea planelor pe care se realizează secțiunea dinamică. Modificarea poziției planelor de secționare se realizează cu componenta *JScrollBar*;
- butoanele 4, 5, 6 se vor prezenta în capitolul 4.

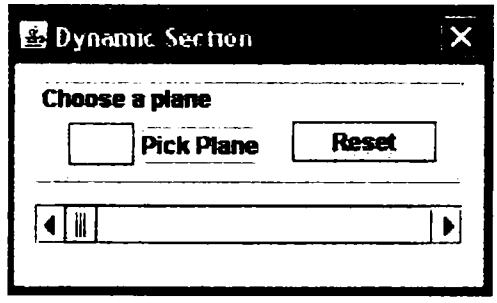


Figura 4.21: Fereastra de dialog *Dynamic Section*

Activarea butonului *Lights* din figura 4.20 deschide fereastra de dialog *Lights Settings* (Figura 4.22). În cadrul acestui dialog, utilizatorul poate modifica o serie de parametrii cu privire la:

- amplasare surselor de lumină în scena 3D. Sunt disponibile două surse de lumină direcționale și două surse de lumină de tip *Point*;
- culoarea surselor de lumină.

Utilizatorul are posibilitatea salvării noilor setări prin activarea butonului *Save new Settings*, sau încărcarea setărilor predefinite de autor prin activarea butonului *Load Default Settings*. Modificarea parametrilor se realizează în mod dinamic (utilizatorul observă imediat rezultatul modificărilor în scena virtuală).

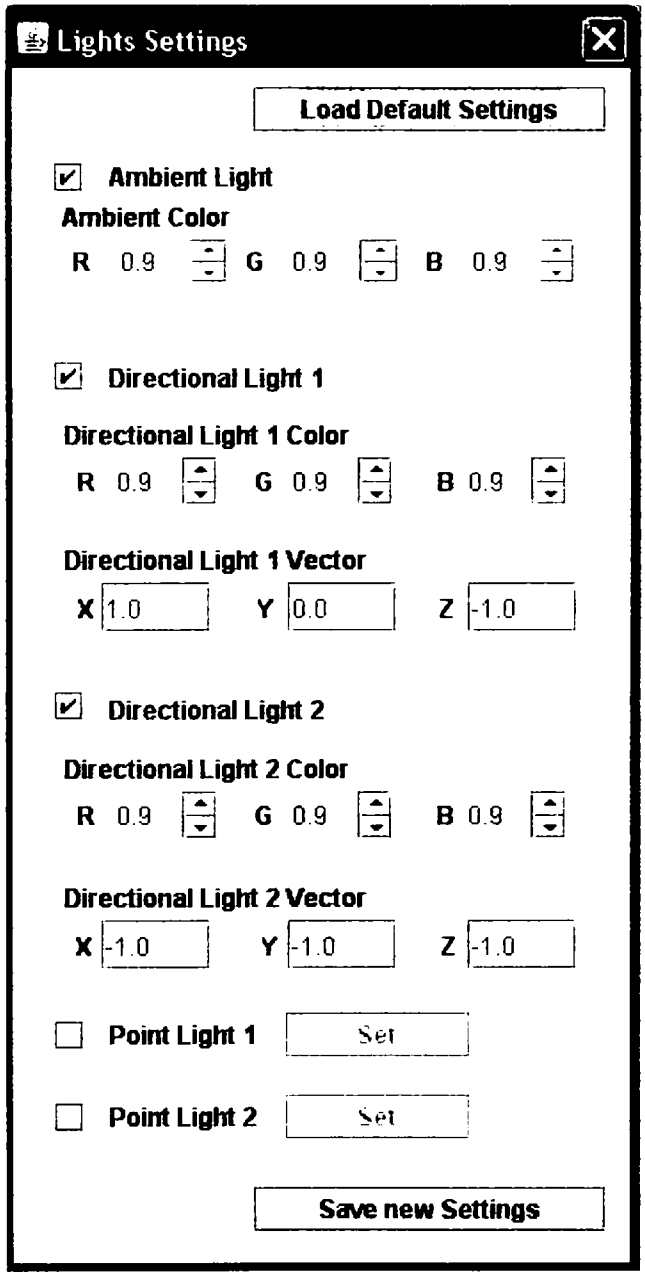


Figura 4.22: Fereastra de dialog *Lights Settings*.

Activarea butonului *Material* din figura 4.20, deschide fereastra de dialog *Material* (Figura 4.23). Se observă că utilizatorul poate modifica în mod dinamic parametrii *Appearance* ai modelului.

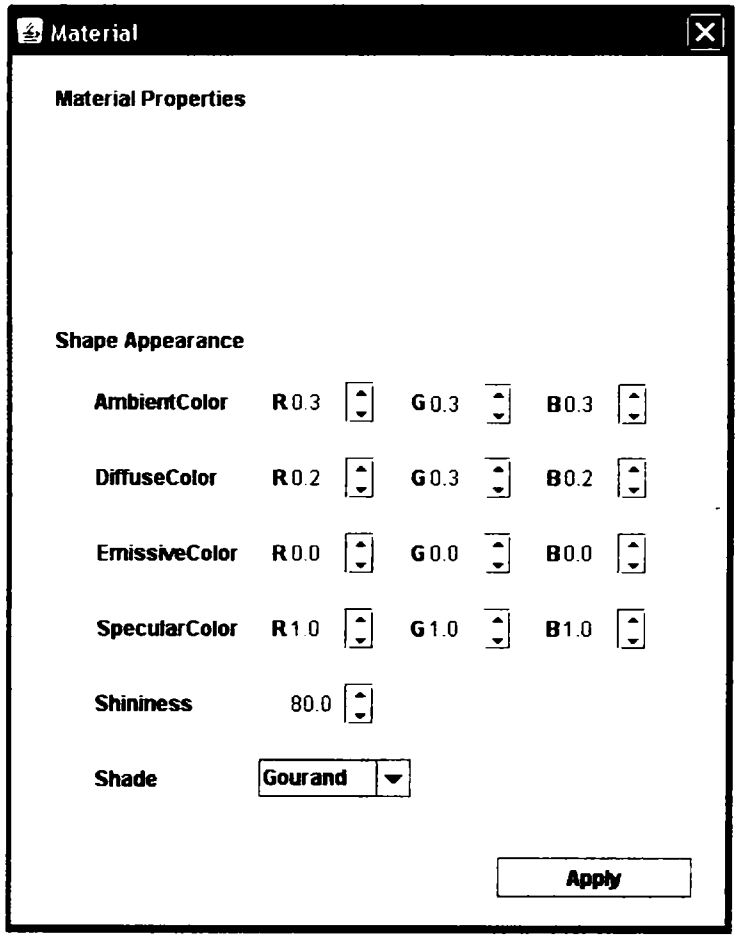


Figura 4.23: Fereastră de dialog *Material*.

Selectarea opțiunii *Preference* din meniul *Settings* are ca efect deschiderea ferestrei de dialog ce conține trei tab-uri. În figura 4.24 este prezentată fereastra de dialog *Preference* corespunzătoare tab-ului *General*. În acest caz utilizatorul poate seta culoarea de *Background* a canvas-ului 3D. Se poate alege între un *background* personalizat (prin modificare dinamică a parametrilor de culoare) sau utilizarea unui *background* predefinit de autor. Activarea butonului *Apply* salvează noile setări în fișierul de configurare a aplicației.

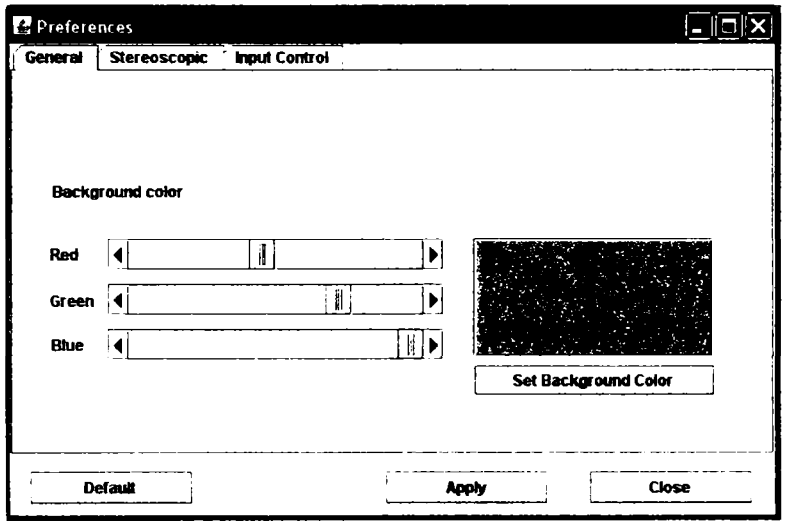


Figura 4.24: Fereastră de dialog *Preferences* tab *General*.

Figura 4.25 prezintă fereastra de dialog *Preference* corespunzătoare tab-ului *Stereosopic*. Prin intermediul acestei ferestre, utilizatorul are posibilitatea de a selecta modul stereoscopic de vizualizare (doar modul *Active Stereo* este disponibil momentan).

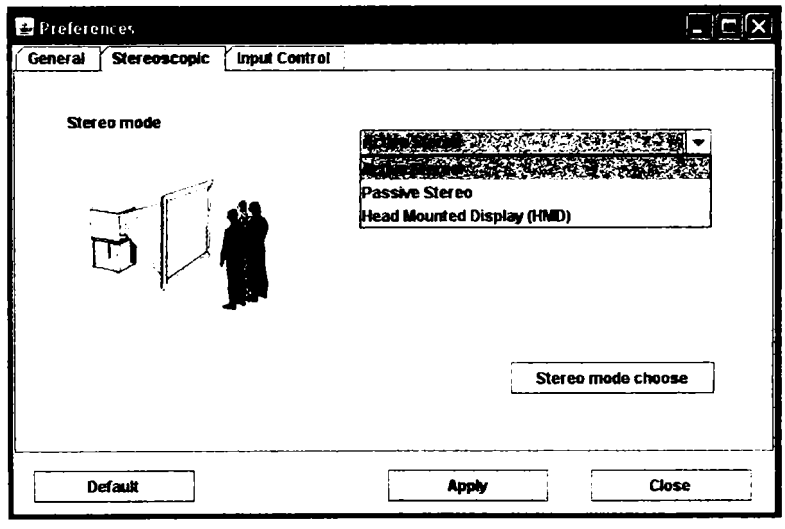


Figura 4.25: Fereastră de dialog *Preferences* tab *Stereoscopic*.

Fereastra de dialog *Preference* corespunzătoare tab-ului *Input Control* este prezentată detaliat în capitolul 4.

Nu în ultimul rând, fereastra de dialog *About* (Figura 4.26), ce poate fi accesată din meniul *Help*, oferă utilizatorului informații cu privire la aplicația "VRforCAD" (detalii despre autorul aplicației și tipul de licență sub care este declarată aplicația).

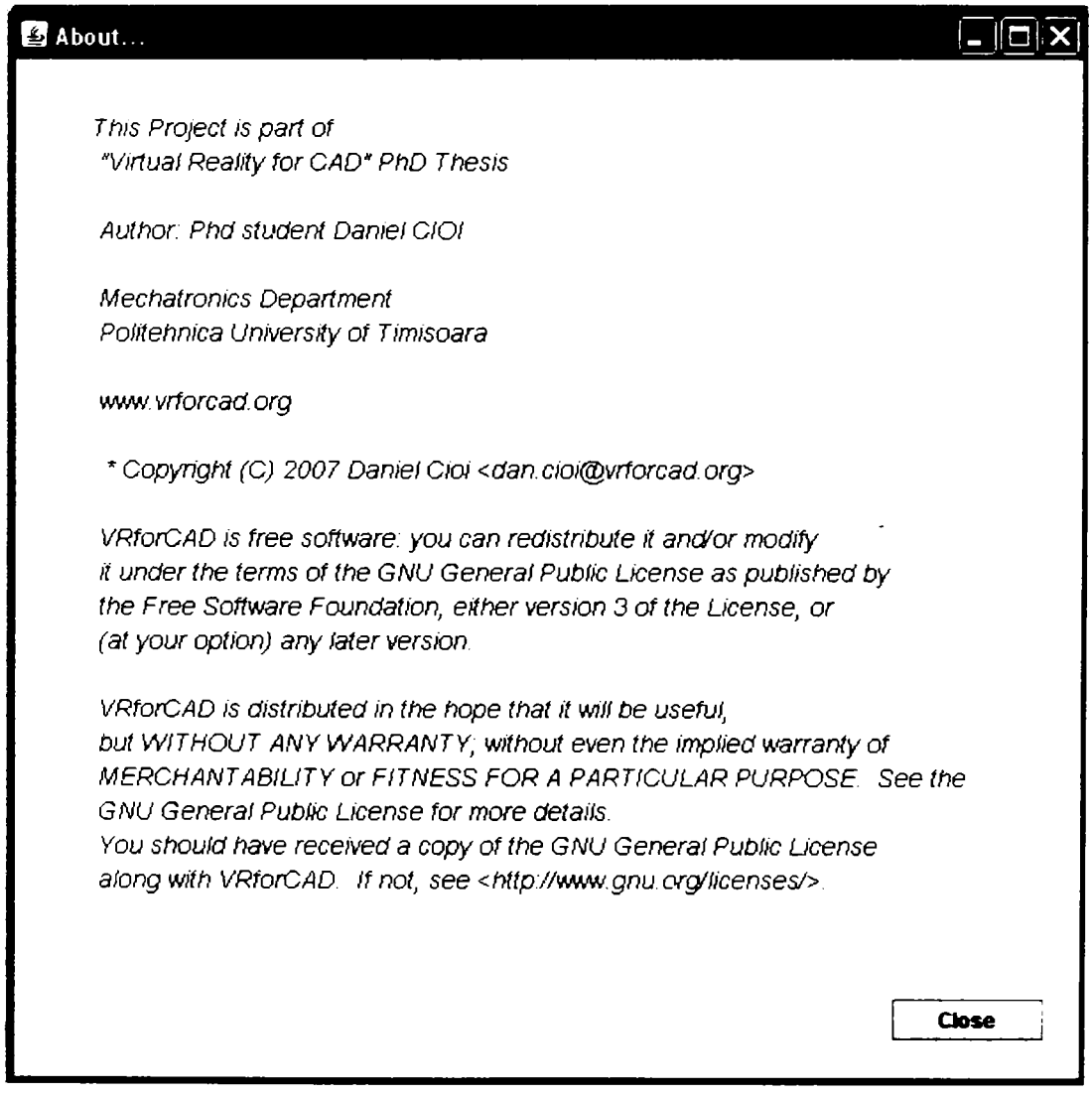


Figura 4.26: Fereastra de dialog *About*.

4.18 Domenii de utilizarea a aplicației "VRforCAD"

În prezent aplicația "VRforCAD" cuprinde 4 moduri de operare, selectabile din meniul *Applications* (Figura 4.27):

- CAD: import export modele CAD cu posibilitatea modificării suprafețelor în mod freeform, modificarea atributelor de culoare, opțiune pentru vizualizare de tip wire-frame, vizualizare de tip stereoscopic (activ); exemplu în figura 4.28 - cutie de bere din aluminiu (model în format stl);

- *Visualization*: permite încărcarea modelelor CAD și modelelor X3D în scopul vizualizării cu posibilitate de manipulare: *rotate, zoom and translate*; modificarea iluminării scenei 3D (activare, poziție, intensitate, culoare); scena conține o sursă de lumină ambientală, două surse de lumină direcționale și două surse de lumină punctiforme: exemplu în figura 4.29 - o mașină marca Porsche (model în format stl);

- *Robotics*: permite realizarea unei celule de lucru robotizată (în mediu virtual) prin încărcarea în scena 3D a roboților (în format dxf, stl, x3d) cu specificarea poziție în scenă ajutând astfel la găsirea soluției optime pentru amplasarea roboților reali; exemplu în figura 4.30 - doi roboți marca ABB amplasați într-un spațiu limitat de 4 stâlpi de susținere (modele CAD ale roboților în format stl sunt disponibile pe site-ul producătorului);

- *Medicine*: permite vizualizarea modelelor obținute de la tomograf, digitizor sau scaner (în format stl) cu opțiune de secțiune dinamică pe mai multe plane concomitent; exemplu în figura 4.31 - un craniu uman obținut de la tomograf, prezentat în secțiune dinamică (model în format stl).

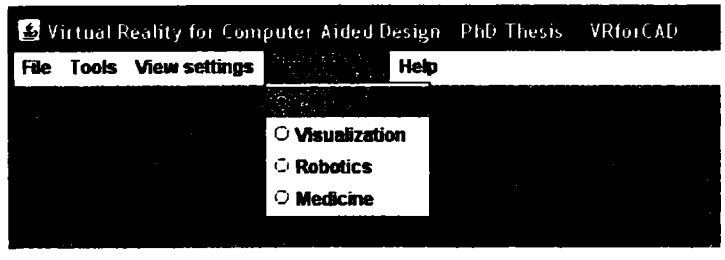


Figura 4.27: Meniul *Applications*.

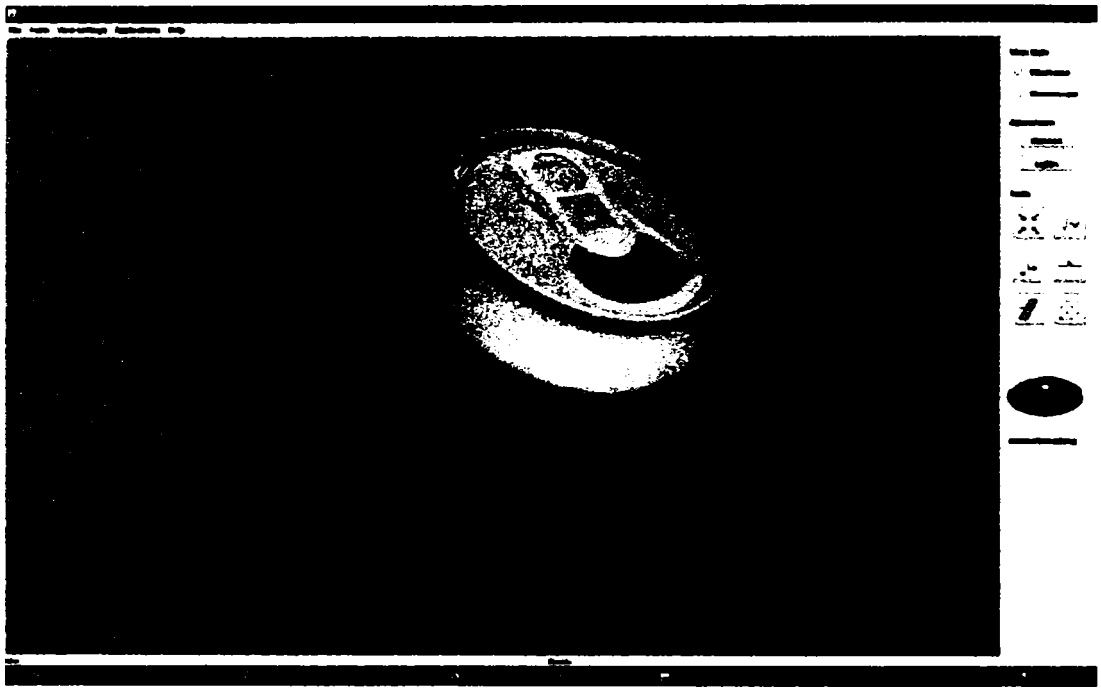


Figura 4.28: Exemplu de operare în modul CAD.

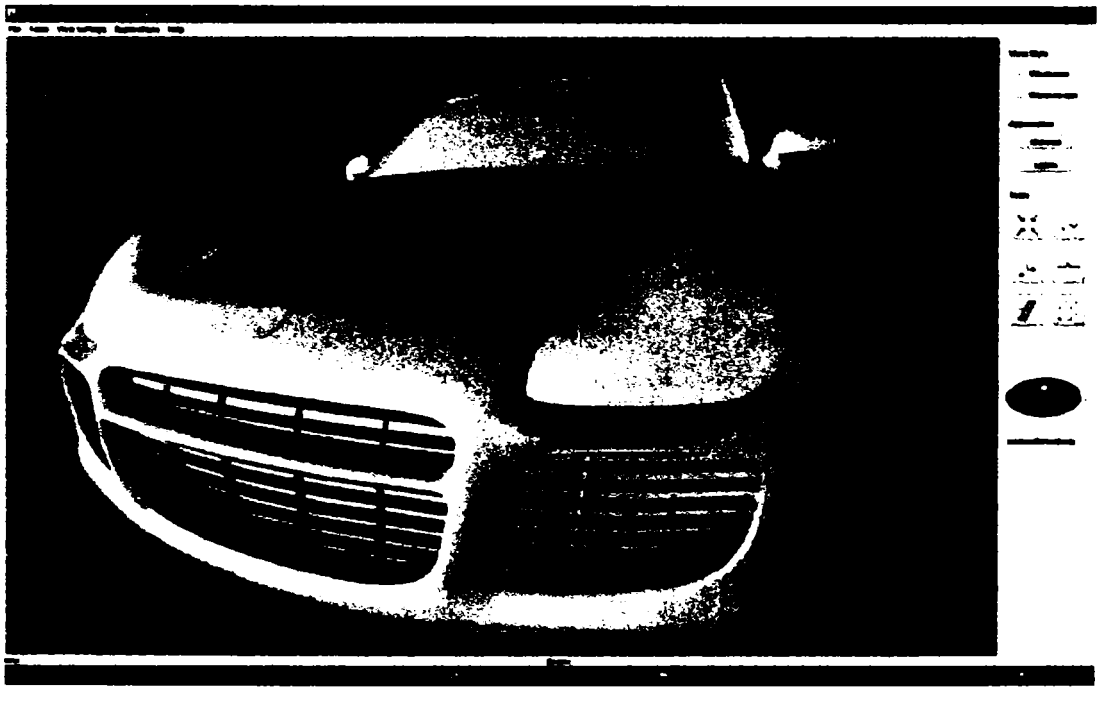


Figura 4.29: Exemplu de operare în modul *Visualization.*

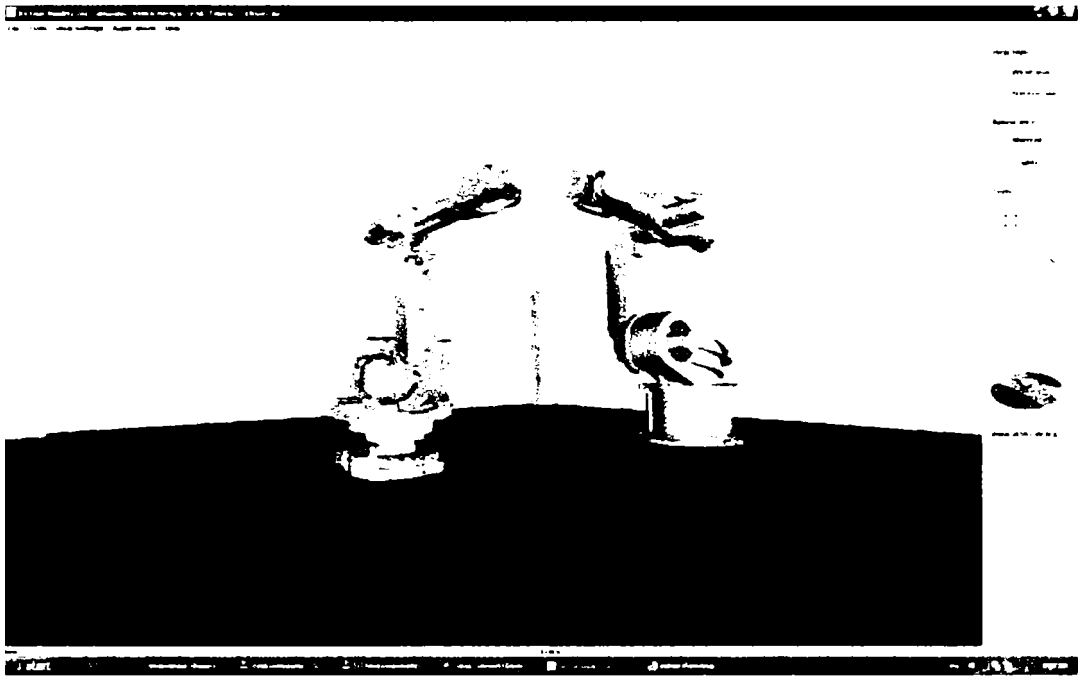


Figura 4.30: Exemplu de operare în modul *Robotics*.

Figura 4.31: Exemplu de operare în modul *Medicine*.

4.19 Concluzii

În acest capitol a fost prezentată aplicația "VRforCAD", dezvoltată în Java și Java3D. Aplicația conține 75 clase ce însumează peste 12000 linii de cod.

Se evidențiază locul aplicației în categoria aplicațiilor Open Source.

La inceputul capitolului se trec în revistă softuri CAD și softuri VR din categoria *Open Source*. Astfel se evidențiază originalitatea aplicației "VRforCAD" prin portabilitatea (independența platformei pe care rulează) asigurată de programarea în limbajul Java și API-ul Java3D ("*write once, compile it once, and run it anywhere*"). De asemenea, faptul că prețul de dezvoltare este zero (IDE (*Integrated Development Environment*) *Freeware*), reprezintă un alt avantaj al programări în Java.

Aplicația permite convertirea formatelor CAD din stl în dxf și invers. De asemenea, aplicația are inclus un format de fișier propriu intitulat vfc. Astfel modele aflate în lucru pot fi păstrate în acest format până la definitivarea prelucrărilor, urmând apoi să fie exportate intr-unul din cele două formate neutre de fișier CAD. Fișierele CAD pot fi păstrate local pe *hard disk* sau *online* folosind un server MySQL de bază de date.

Utilizarea unui server de bază date MySQL permite folosirea comună a bazelor de date CAD, ceea ce are următoarele avantaje: clienți diferiți lucrează asupra aceleiași bază de date CAD situată în spațiu comun; clienții importă baza de date pentru a o dezvolta (continuarea proiectului); clienții importă componente diferite, de exemplu pentru le integra într-o nouă entitate.

În continuare, este prezentată o soluție originală proiectată, dezvoltată și implementată de către autor pentru realizarea unui mediu de lucru colaborativ.

Interfața grafică cu utilizatorul a aplicației "VRforCAD", a fost proiectată sub o formă "prietenoasă", ușor de înțeles de către utilizator.

Capitolul se încheie cu o scurtă prezentare a celor 4 moduri de operare ale aplicației "VRforCAD". Aceste moduri de operare sunt: *CAD, Visualization, Robotics* și *Medicine*.

Aplicația "VRforCAD" este înregistrată sub licență GPL v3 în scopul disponibilității pentru o continuă dezvoltare.

Capitolul 5

Contribuții la proiectarea, realizarea și testarea unui echipament hardware pentru sistemul VR-CAD

5.1 Introducere

În acest capitol se prezintă echipamentul hardware intitulat "SphereDevice", conceput și realizat practic de către autorul acestei teze.

Noutatea acestui echipament constă în:

- spațiul de lucru al echipamentului este în corelanță cu percepția de spațiu obținută prin proiecție stereoscopică;

- portabilitatea (funcționează pe sisteme Unix, Linux și Windows) oferită de limbajul Java (driver-ul este dezvoltat în Java) permite integrarea ușoară a echipamentului în orice aplicație dezvoltată în Java (multe echipamente de tip *haptic device* au probleme legate de suportul pentru platforme Linux și Unix);

- nu în ultimul rând, prețul de producție al echipamentului (în jur de 300 euro) reprezintă un important atu.

În cele ce urmează, se face o scurtă prezentare a noțiunii *haptic feedback* după care este descris echipamentul "SphereDevice", respectiv principiul de funcționare al acestuia.

5.2 Haptic feedback

Haptic feedback ("haptic" în limba greacă înseamnă "a putea atinge") este o modalitate senzorială crucială în interacțiile din realitatea virtuală. *Haptic* reprezintă ambele: reacție de forță "*force feedback*" simularea durității obiectelor, greutatea și inerția și reacție tactilă "*tactile feedback*" (simulează contactul cu suprafața geometriei, netezimea, alunecarea și temperatura). Asigurarea unor astfel de senzații necesită un echipament special *desk-top* sau portabil cu denumirea de interfață *haptic* [42].

În [17], autorul face o prezentare a echipamentelor cu *feedback* haptic care pot fi utilizate în proiectarea asistată de calculator.

5.3 Ce este "SphereDevice"?

"SphereDevice" este un echipament cu feedback haptic dezvoltat de autorul acestei teze, echipament folosit împreună cu aplicația "VRforCAD" (Figura 5.1, 5.2, 5.3). Echipamentul este compus din: un cub (doar muchii) din aluminiu, trei transmisii liniare și o sferă ce se deplasează pe cele 3 axe (x, y, z). Scopul realizării acestui echipament este înlocuirea tradiționalului mouse (cu doar două coordonate (x, y)). Folosirea rotiței de scroll de la mouse pentru deplasare pe axa Z în scena virtuală fiind incomodă.

Interacțiunea cu scena virtuală se realizează prin intermediul sferei - parte componentă a echipamentul "SphereDevice". Sfera este realizată din plastic prin procedeu de *rapid prototyping* (Figura 5.4, 5.5) și are în componență șase microswitch-uri (câte două pentru fiecare axă), ce comandă prin intermediul aplicației "VRforCAD" cele trei motoare pas cu pas ale echipamentului. Sfera din componența echipamentului are corespondent virtual în aplicația "VRforCAD" tot o sferă.

Interfațarea echipamentului cu calculatorul este realizată prin portul paralel, iar

lista de materiale necesare realizării fizice a echipamentului se regăseste in Anexa B.

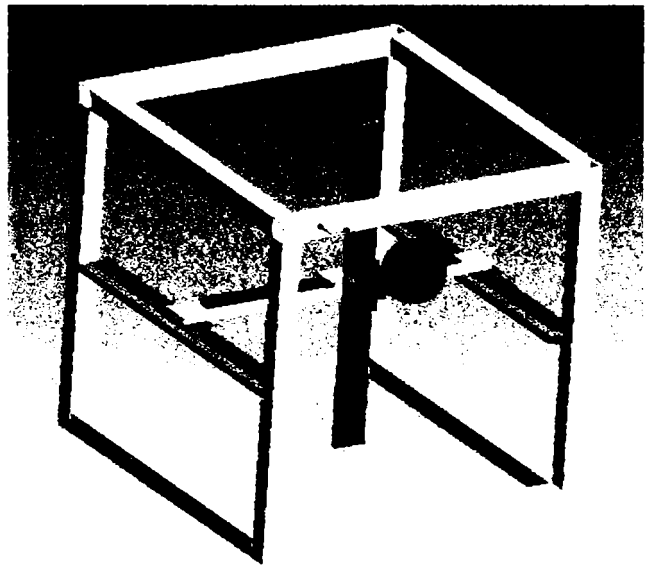


Figura 5.1: SphereDevice - modelul CAD in ProE.

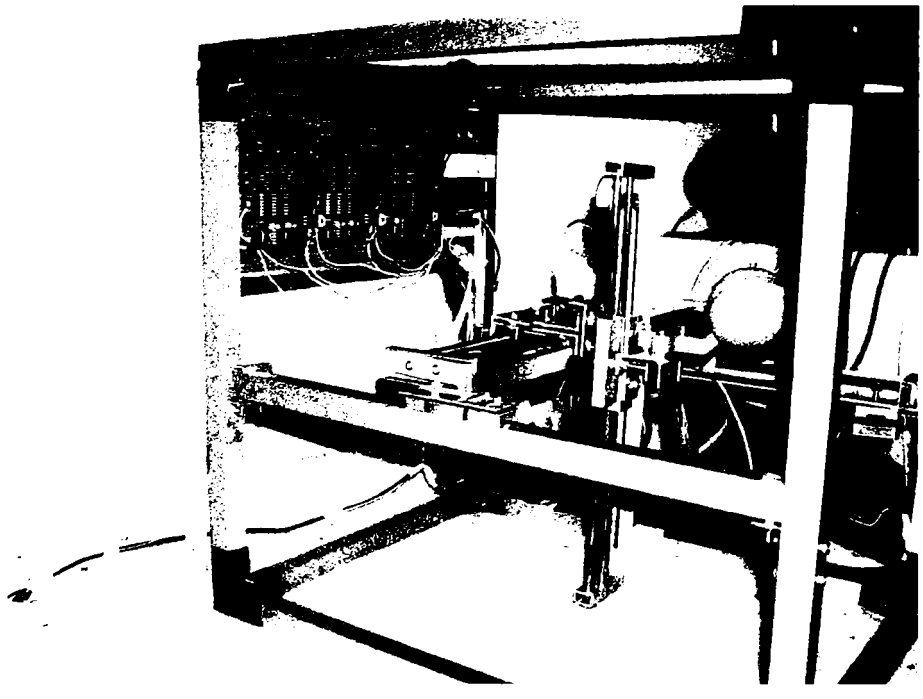


Figura 5.2: Echipamentul "SphereDevice" - realizat practic.

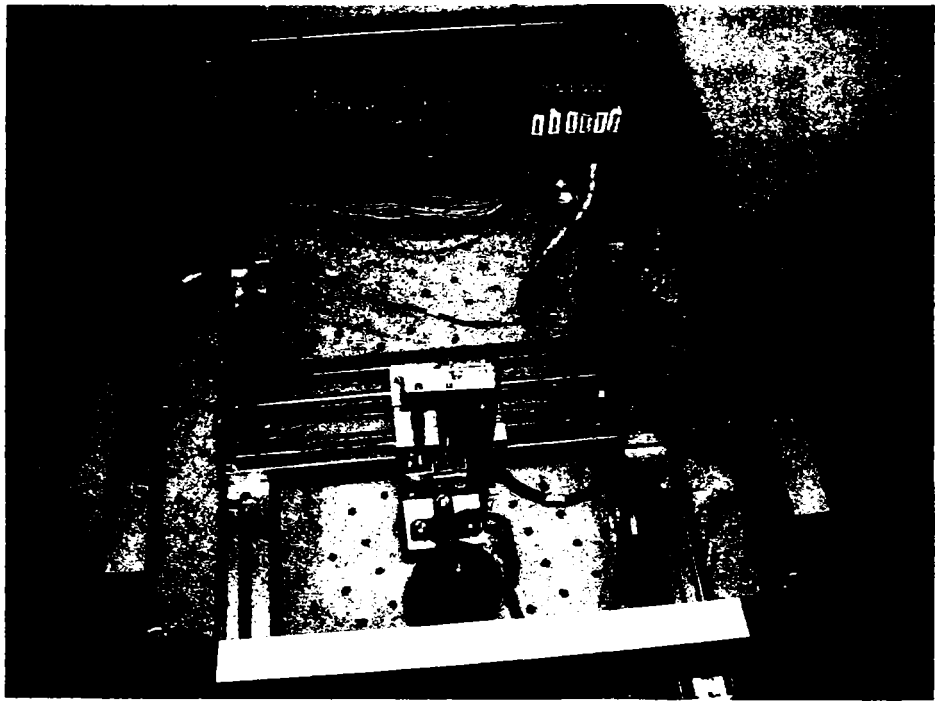


Figura 5.3: Echipamentul "SphereDevice" - realizat practic.

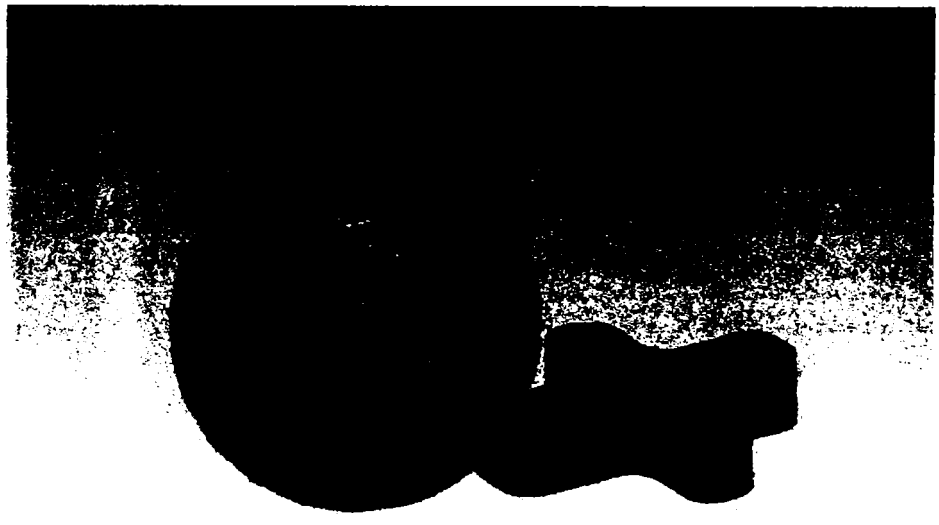


Figura 5.4: Modelul CAD in ProE.

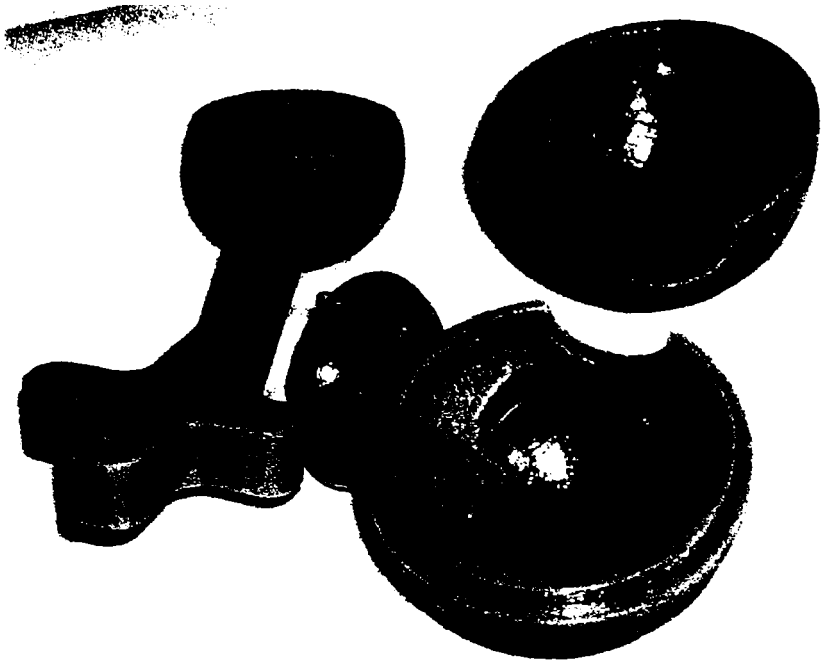


Figura 5.5: După prototipare rapidă.

5.4 Principiul de funcționare al echipamentului "SphereDevice"

Schema electronică redată în figura 5.7, realizată de către autor în Protel 99 SE, prezintă principiul de funcționare din punct de vedere electronic al echipamentului "SphereDevice". Interfațarea echipamentului cu calculatorul se efectuează prin portul paralel cu conector de tip DB25.

În schemă se observă trei circuite de comandă cu optocuploare a trei relee. Tensiunea de alimentarea a acestor circuite este de 5V. Cel din stânga primește semnal de la pinul 8 al portului paralel și are rol de a alimenta cu tensiune întregul circuit. Functiile de power on si power off sunt realizate din aplicație. Celelalte două circuite, au rolul de a compensa lipsa a încă două linii de semnal pe registrul de stare a portului paralel.

Unealta sferă are 6 microswitch-uri (Figura 5.6 a) folosite pentru comanda a trei axe (două microswitch-uri pentru fiecare axă). Portul paralel are doar 5 linii de semnal

pentru registrul de stare. O linie de semnal (pinul 15) este utilizată pentru aducerea în poziția de acasă a echipamentului. Astfel pentru primire semnal circuit închis de la microswitch-urile sferei s-au folosit următorii pini (S7. S6. S5. S4. S3 sunt liniile registrului de stare a portului paralel):

- axa Z – - pinii 10 (S7) și 11 (S6) ai portului paralel:
- axa X – - pinii 12 (S5) și 13 (S4) ai portului paralel:
- axa Y – - împreună pinii 15 (S3) și 10 (S7) pentru deplasare pozitivă și pinii 15 (S3) și 11 (S6) pentru deplasare negativă.

Pentru comanda controlerelor (circuite de comandă a motoarelor pas cu pas). s-au folosit următoarele linii de semnal a registrului de date: axa Z. pinul 2 (D0) pentru pas și pinul 3 (D1) pentru sens: axa Y. pinul 4 (D2) pentru pas și pinul 5 (D3) pentru sens: axa X. pinul 6 (D4) pentru pas și pinul 7 (D5) pentru sens.

La pinul 15 (S3) sunt conectate (în paralel) alte 6 switch-uri (Figura 5.6 b)(două pentru fiecare axă) folosite pentru aducerea în poziția de acasă a echipamentului. Deoarece echipamentul nu are traductoare de poziție. la pornire se execută o deplasare pe fiecare axă (în sens pozitiv) până la primirea semnalului circuit închis pe pinul 15 al portului paralel. Cunoscându-se lungimea fiecărei axe. se comandă aducerea uneltei sferă la poziția de mijloc a fiecărei axe. De asemenea. switch-urile au si rol de limitare de cursă.

Tensiunea de alimentare a circuitelor de comandă a motoarelor pas cu pas este de 12V. Ambele tensiuni (12V și 5V) sunt obținute de la o sursă de calculator de tip ATX.

Deplasarea ansamblului de axe nu este realizată prin împingere de către utilizator a săniilor axelor ci doar prin simpla atingere a sferei. softul comandând motoarele ce execută deplasarea. De asemenea softul (aplicația "VRforCAD") stabilește viteza de deplasare. sensul deplasării. când trebuie să se deplaseze etc.

Figura 5.6: Switch-uri folosite în construcția echipamentului.

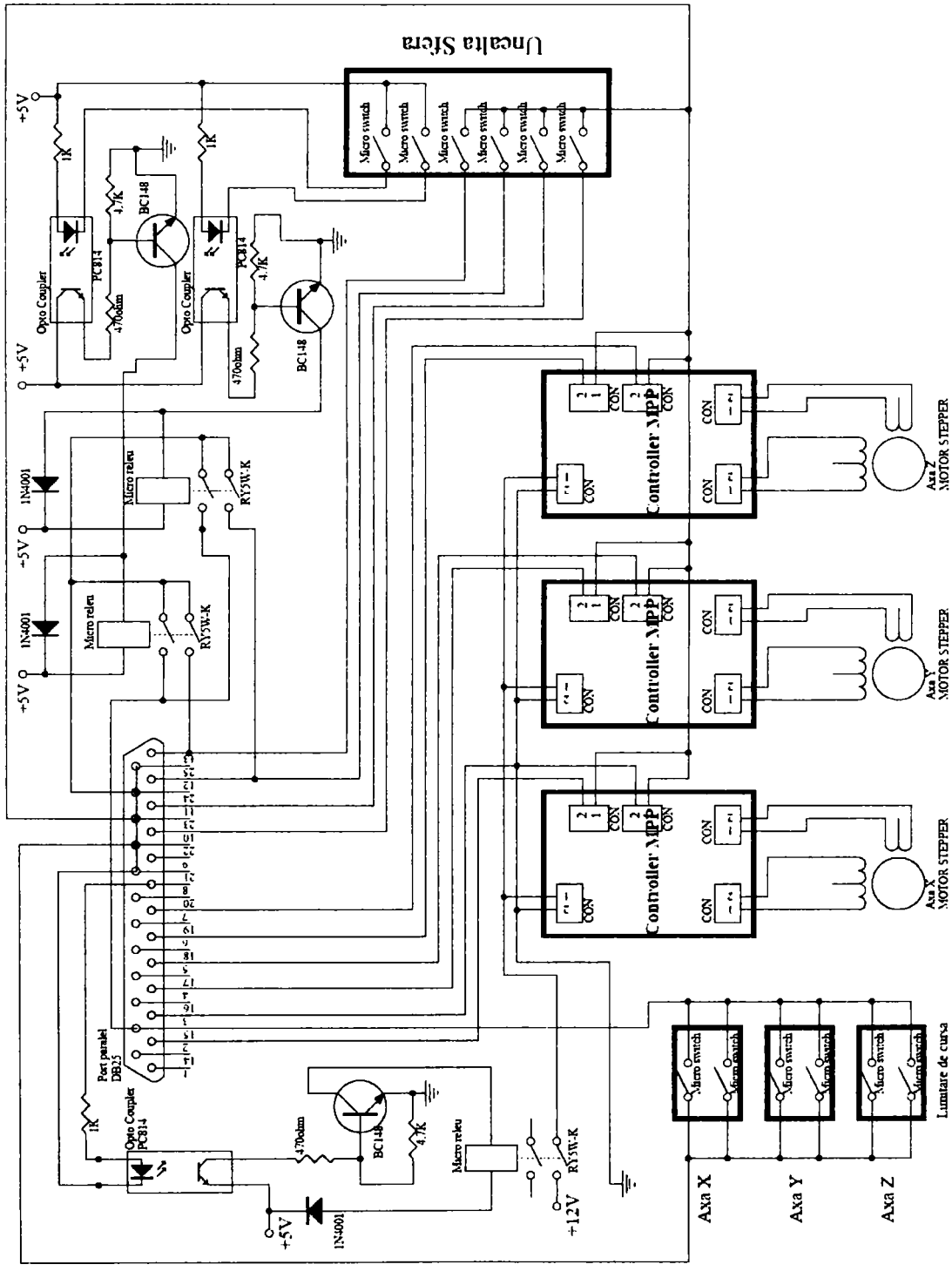


Figura 5.7: Schema electronică a echipamentului "SphereDevice".

Motoare pas cu pas

În cele ce urmează se prezintă o scurtă introducere în domeniul motoarelor pas cu pas [29].

Motorul pas cu pas (MPP) este un convertor electromecanic care realizează transformarea unui tren de impulsuri digitale într-o mișcare proporțională a axului său. Mișcarea rotorului MPP constă din deplasări unghiulare discrete. succesive. de mărimi egale și care reprezintă pașii motorului. În cazul unei funcționări corecte. numărul pasilor efectuați trebuie să corespundă cu numărul impulsurilor de comandă aplicate motorului. Deplasarea unghiulară totală. constituită dintr-un număr de pași egal cu numărul de impulsuri de comandă aplicate motorului. determină poziția finală a rotorului. Această poziție se păstrează. adică este memorată. până la aplicarea unui nou impuls de comandă. Proprietatea de univocitate a conversiei impulsuri - deplasare. asociată cu aceea de memorare a poziției. fac din MPP un excelent element de execuție. integrat în sistemele de reglare a poziției în circuit deschis. MPP mai prezintă proprietatea de a putea intra în sincronism față de impulsurile de comandă chiar din stare de repaus. funcționând fără alunecare. frânarea efectuându-se. de asemenea. fără ieșirea din sincronism. Datorită acestui fapt se asigură porniri. opriri si reversări bruște fără pierderi de pași în tot domeniul de lucru.

Viteza unui MPP poate fi reglată în limite largi prin modificarea frecvenței impulsurilor de intrare. Astfel. dacă pasul unghiular al motorului este 1.8° numărul de impulsuri necesare efectuării unei rotații complete este 200. iar pentru un semnal de intrare cu frecvența de 400 impulsuri pe secundă turația motorului este de 120 rotații pe minut. MPP pot lucra până la frecvențe de 2000 pași secundă. având pași unghiulari cuprinși între 180° si 0.3°.

Construcția motoarelor pas cu pas este de mai multe tipuri: rotative sau liniare. numărul înfășurărilor de comandă variind între unu și cinci. Din punct de vedere al construcției circuitului magnetic sunt trei tipuri principale:

- cu reluctanță variabilă (de tip reactiv):
- cu magnet permanent (de tip activ):
- hibride.

Utilizarea MPP conferă următoarele avantaje:

- asigură univocitatea conversiei număr de impulsuri - deplasare și pot fi utilizate în circuit deschis;
- gamă largă a frecvențelor de comandă;
- precizie de poziționare și rezoluție mari;
- permit porniri, opriri, reversări fără pierderi de pași;
- memorează poziția;
- sunt compatibile cu comanda numerică.

Motoare pas cu pas utilizate în construcția echipamentului "SphereDevice"

Pentru construirea echipamentului s-au folosit 3 motoare pas cu pas bipolare cu 4 fire. Caracteristicile celor 3 motoare sunt descrise în Tabela 5.1

Recuperat din:	**Model**	**Unghiul de pas**	**Rezistenta înfășurărare**	**Tensiune nominală**
Canon BJC 1000	Mitsumi M42SP-4	1.8°	11Ω	12V
Epson Staylus color 640	EM-257	1.8°	8Ω	39.9V
-	Teco 4H4018X0711	1.8°	7Ω	5.1V

Tabela 5.1: Caracteristicile motoarelor pas cu pas folosite la contruirea echipamentului "SphereDevice"

Circuite integrate destinate comandării MPP

Numărul foarte mare de produse și echipamente, care utilizează motoare pas cu pas, a impus proiectarea și realizarea unor circuite integrate specializate, care să includă o parte sau totalitatea funcțiilor, necesare comenzii MPP. În majoritatea cazurilor se tinde spre simplificarea rolului procesorului, încât acesta să genereze numai semnalul de tact (frecvență), corespunzător numărului de pași/secundă ai MPP și un semnal binar pentru selectarea sensului de rotație (orar - CW sau anti-orar -CCW) a axului MPP. Cele mai

adecvate sunt procesoarele care au ieșiri PWM, întrucât simpla programare a registrelor interne care comandă aceste ieșiri asigură semnalele de frecvențe dorite. Numărul de ieșiri PWM ale unui procesor coincide cu numărul de MPP care pot fi comandate (de exemplu, 2 la microcontrollerul 80552, 6 la familia C166 produs de Siemens etc.)

Gama de circuite integrate, destinate comenzii MPP este extrem de amplă, iar aceste circuite, în funcție de complexitatea lor, pot asigura parțial sau integral toate problemele pe care această comandă le implică: distribuirea impulsurilor pe faze pentru diferite moduri de comandă - în secvență simplă, dublă, mixtă sau în micropășire, în mod unipolar sau bipolar; comanda de putere (la curenții nominali) a fazelor; forțarea; supresarea. Există și circuite integrate care au capacitatea computațională de a comanda accelerarea, mersul uniform și decelerarea MPP, conform unor profile de viteză impuse, eliberând astfel procesorul de aceste operații.

Pentru comanda motoarelor pas cu pas s-a ales folosirea controlerului (circuit de comandă a motoarelor pas cu pas) de tip ep0090 produs de firma EPSICOM [33], câte un controler pentru fiecare motor. Următoarele două figuri prezintă: schema electrică (Figura 5.8) și amplasarea componentelor (Figura 5.9). Fiecare motor este comandat pe 2 biți (sens și pas).

Circuitul de comandă are următoarele caracteristici:

- Finali cu tranzistori MOSFET;
- Cuplu constant de la 1-290 RPM;
- Comanda motoare cu 4 sau 6 terminale de la 5 la 50V Imax. 14A;
- Izolare galvanică;
- Răcire uniformă a finalilor.

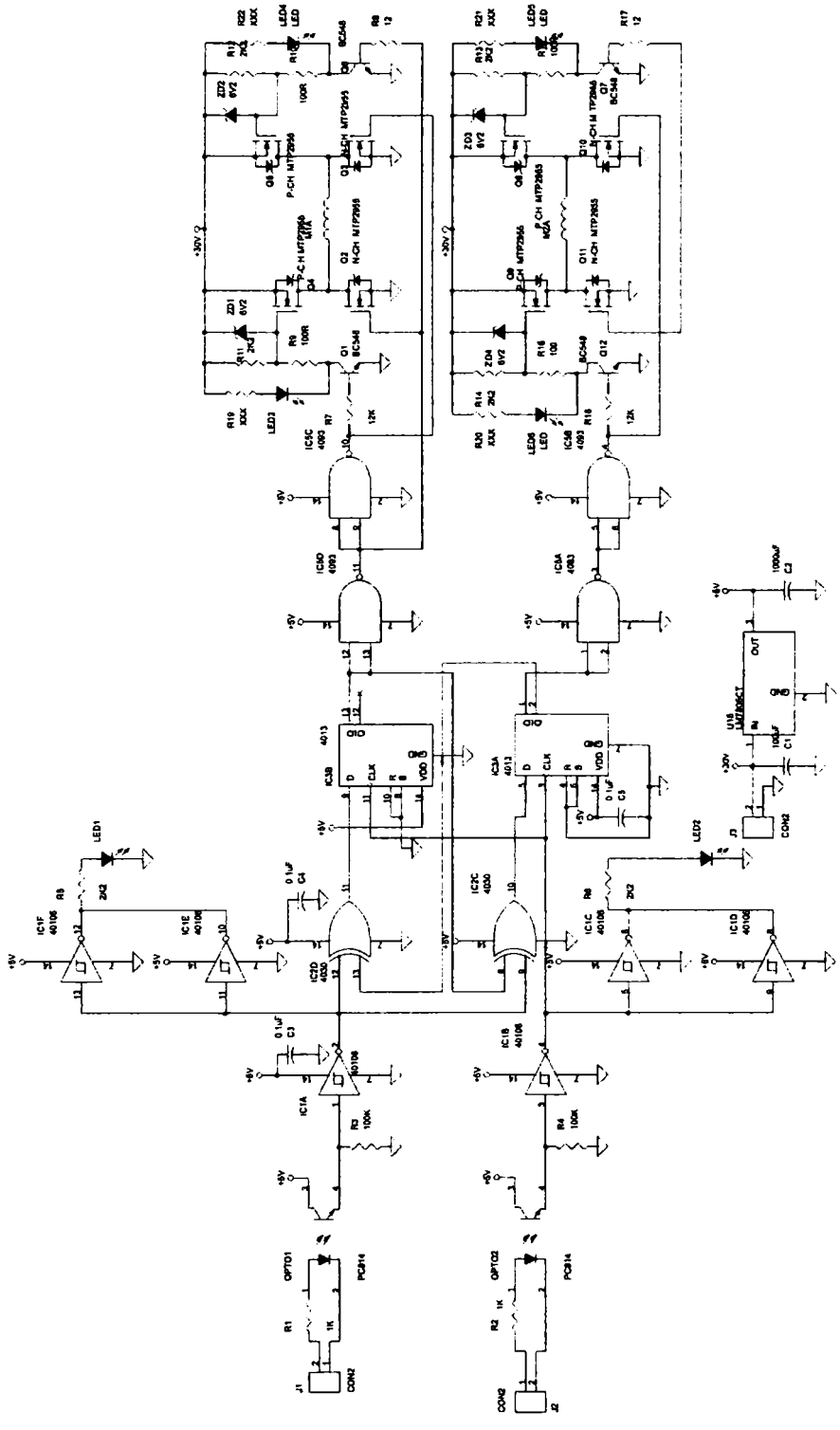


Figura 5.8: Schema electrică *drive stepper*.

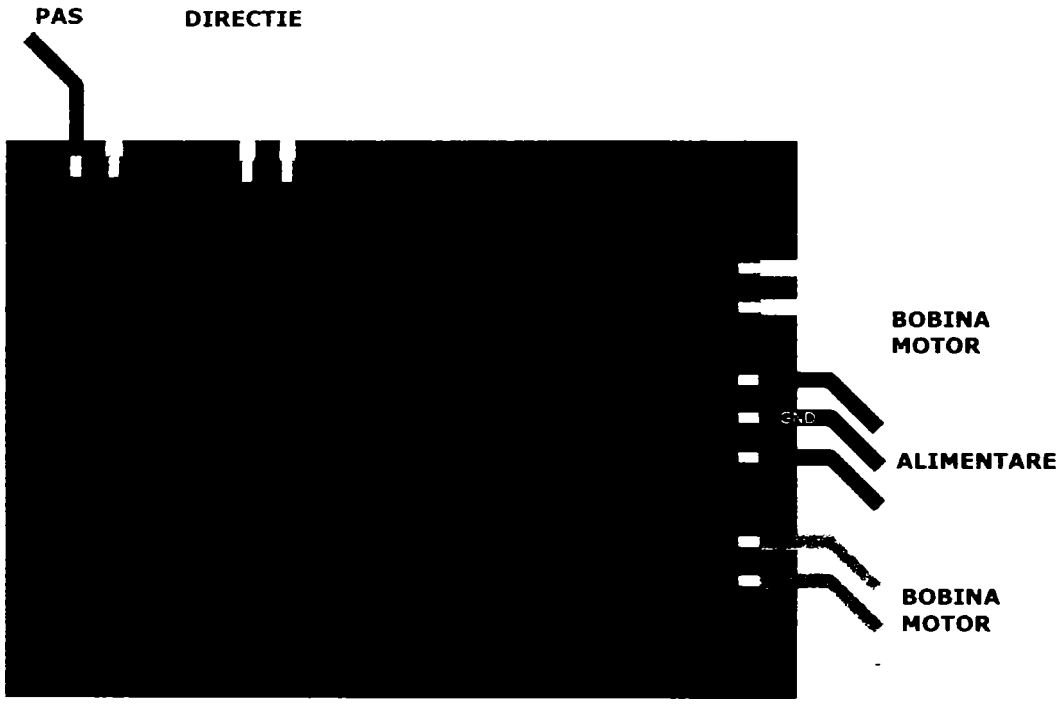


Figura 5.9: Amplasarea componentelor.

Portul paralel

Portul paralel, așa cum este implementat în PC, constă într-un conector cu 17 linii de semnal și 8 linii de masă. Liniile de semnal sunt clasificate în 3 categorii (Figura 5.10):

- linii de control (4);
- linii de stare (5);
- linii de date (8).

În varianta originală a portului paralel, liniile de date au fost prevăzute pentru a transfera date către imprimantă, ca urmare erau unidirecționale. În variantele ulterioare, aceste linii au devenit bidirecționale. Liniile de control au fost prevăzute pentru dialog și control al imprimantei, iar liniile de stare au fost prevăzute pentru dialog și indicare a stării imprimantei. La variantele ulterioare, liniile de stare au fost folosite și pentru a transmite date către port.

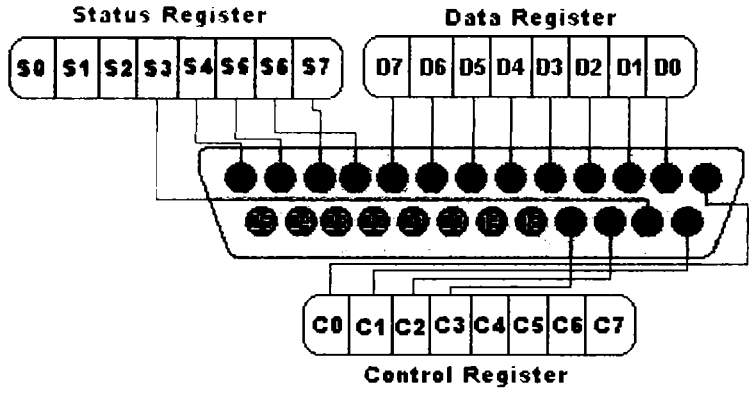


Figura 5.10: Conectorul DB25.

Tabelul 5.2 conține simbolul, descrierea și alocarea la conectorii tipici ai fiecărui semnal al portului paralel.

Semnalul nSTROBE este emis de calculator pentru a comunica perifericului că există informație validă pe liniile de date. Pentru aceasta se generează un impuls negativ care apare după 0.5 μS de la apariția datelor și durează cel puțin 0.5 μS și cel mult 50 μS.

Semnalul BUSY comunică cu nivel HIGH că nu este pregătit să preia date, altele decât cele curente sau, că pur și simplu nu poate prelua date dintr-un motiv sau altul (eroare, nu este on-line, nu are hârtie). Semnalul BUSY este emis de periferic și comută în "1" imediat după detectarea semnalului nSTROBE activ. Rămâne în această stare până la terminarea recepției datelor semnalizate valide.

Semnalul nACK este emis de periferic și comunică printr-un impuls negativ că ultimul Byte a fost recepționat. Durata impulsului este de 0.5 μS - 10 μS.

Semnalele nSTROBE, BUSY și nACK controlează fluxul de date, celelalte semnale de interfață sunt ajutătoare în dialogul dintre calculator și o imprimantă.

Portul paralel este mapat în spațiul I/O al PC-ului. Fiecărei grupe de semnale îi este asociat un registru; în cadrul fiecărui registru, fiecărui semnal îi este precizată poziția. Fiecare registru este apelabil cu o adresă de port. Adresa de bază a portului paralel poate fi 3BCh, 378h sau 278h. La această adresă se apelează registrul de date, iar celelalte (stare și control) se apelează cu un *offset*. Tabelul 5.3 conține informații legate de registrele portului paralel.

Grupa	Semnal SPP	In/Out	Descrierea semnalului	DB (25)
Control	nSTROBE	Out	Activ LOW. Sunt date valide pe liniile de date	1
	nAUTOFEED	Out	Activ LOW. Imprimanta va include LF la fiecare CR.	14
	nSELECTIN	Out	Activ LOW. Spune imprimantei că este selectată.	17
	nINIT	Out	Activ LOW. Resetează imprimanta.	16
Stare	nACK	In	Un puls negativ pentru a semnaliza că ultimul byte a fost recepționat.	10
	BUSY	In	Nivel HIGH, anunța calculatorul că imprimanta nu poate prelua date.	11
	PE	In	Nivel HIGH, imprimanta nu are hârtie.	12
	SELECT	In	Nivel HIGH, imprimanta este ON LINE.	13
	nERROR	In	Nivel LOW, la imprimantă există o condiție de eroare.	15
Date	DATA [1:8]	Out	8 linii de date (numai ieșire)	2-9
GND		-	masă de semnal	18-25

Tabela 5.2: Semnalele portului paralel standard

Observații:

1. Offset indică deplasamentul față de adresa de bază a portului paralel.

2. Registrul de date poate fi citit în cazul tuturor porturilor, dar se poate utiliza pentru o operație de intrare numai în cazul porturilor bidirecționale.

3. Registrul de control poate fi citit în scopuri de testare. Valoarea citită trebuie să fie egală cu cea înscrisă anterior în acest registru.

* Cu fiecare bit din aceste registre poate fi controlat nivelul unui semnal al interfeței. Specificația inversat neinversat are următoarea semnificație:

- inversat: setarea unui "1" are ca rezultat generarea unui nivel LOW pentru semnalul respectiv.

- neinversat: setarea unui "1" are ca efect generarea unu nivel HIGH pentru semanlul respectiv.

Offset	Nume	Mod de operare	Descriere
0	Registru de date DB0-DB7	 neinversat *	Conține biții de date; R/W DATA [1:8]
1	Registru de stare DB3 DB4 DB5 DB6 DB7	 inversat * neinversat * neinversat * inversat * inversat *	Conține biți de stare; R nERROR SELECT PE nACK BUSY
2	Registru de control DB0 DB1 DB2 DB3 DB4	 neinversat * neinversat * inversat * neinversat * neinversat *	Conține biți de control; W nSTROBE nAUTOFEED nINIT nSELECTIN IRQE **

Tabela 5.3: Registre port paralel

Spre exemplu, nSTROBE este activ LOW; pentru a activa acest semnal se înscrie un "0" în registrul de control, pe rangul D0. În schimb, pentru a activa nINIT, care este activ tot LOW, se va inscrie un "1" în registrul de control pe ragul D2.

** Bitul alocat rangului D4 în registrul de control, IRQE, nu activează vreun semnal de ieșire; înscrierea unui "1" pe această poziție validează acceptarea activării semnalului nACK ca o cerere de întrerupere venită din partea echipamentului periferic. Această facilitate nu este utilizată în general de imprimante, ea a fost prevăzută pentru aplicații de transfer de date pe la portul paralel.

5.5 Testarea echipamentului "SphereDevice"

Pentru accesarea portului paralel din Java (la nivel de bit) se folosește o bibliotecă *free* ("ParallelPort") disponibilă pentru platforma Windows și Linux [54]. Biblioteca este dezvoltată în limbajul C și este accesată din Java prin Java Native Interface (JNI). ParallelPort este o clasă Java ce permite citirea și scrierea biților către și de la portul parallel al calculatorului

Metodele clasei ParallelPort sunt descrise în tablelul următor:

Prototipul metodei	Descrierea metodei
int read()	Citește un bit de la pinii STATUS ai obiectului port paralel.
static int readOneByte(int address)	Citește un bit de la adresa specificată.
void write(int oneByte)	Scrie un bit la pinii DATA ai obiectului port paralel.
static void writeOneByte(int address, int oneByte)	Scrie un bit la o adresă specificată.

În aplicația "VRforCAD", comanda motoarelor pas cu pas pe portul paralel se face într-un *thread* (fir de execuție) separat. Un pas este executat prin trimiterea unei valori de tip int la portul paralel folosind methoda *write(int oneByte)* pentru a seta semnal activ *high* (+2.4 ... +5 V, current max 14mA) pe bitul de step al controlerului de motor pas cu pas timp de 1ms, apoi se trimite o alta valoare pentru a seta semnalul activ *low* (0 ... +0.8 V).

Următoarea secvență de cod Java prezintă cum se realizează pași într-o buclă while.

```
while(steps){
     lpt1_output.write(2);
          try{
               Thread.sleep(1);
               }
          catch(InterruptedException e){}
     lpt1_output.write(0);
          try{
               Thread.sleep(1);
               }
          catch(InterruptedException e){}
}
```

- linia de cod: lpt1_output.write(2); setează bitul registrului de date D2 în stare electrică *high.*

- linia de cod: lpt1_output.write(0); setează biții registrului de date în stare electrică *low.*

- linia de cod: *Thread.sleep(1)*; întrerupe pentru 1ms execuția thread-ului, astfel se crează timpul necesar setării stări electrice *high* sau *low.*

Valorile de tip int ai biților registrului de date sunt următoarele:

```
D2=1 D3=2 D4=4 D5=8 D6=16 D7=32 D8=64 D9=128
```

Conform documentației de la SUN, "*The sleep times are not guaranteed to be precise, because they are limited by the facilities provided by the underlying OS. In any case, you cannot assume that invoking sleep will suspend the thread for precisely the time period specified*".

Pentru evidențierea problemelor ce apar la comanda controlelor de motoare pas cu pas (utilizând cod Java) s-au executat următoarele 6 teste:

- Testul nr.1 (Figura 5.11), 100 pași cu *delay* de 1ms pentru semnal *high* și 1ms pentru semnal *low*. Concomitent cu acest *thread* rulează un alt *thread* ce ascultă registrul de stare de pe portul paralel. Din grafic se observă menținerea aproximativ constantă a timpului de execuție a metodei *sleep()* în jurul valorii de 8ms cu aparitia unor valori extreme. Analizând eșantionul experimental, s-a utilizat ca estimator al valorii centrale a șirului de date modulul acestora (valoarea cu cea mai mare frecvență de apariție):

- Testul nr.2a (Figura 5.12), 100 pași cu *delay* de 1ms pentru semnal *high* și 1ms pentru semnal *low* (un singur *thread* în execuție). Din grafic se observă menținerea aproximativ constantă a timpului de execuție a metodei *sleep()* în jurul valorii de 8ms cu aparitia unor valori extreme.

- Testul nr.2b (Figura 5.13), același ca și testul nr.2a dar rulat a doua oară. Comparând cele două teste se observă un comportament diferit prin repartizarea și frecvența valorilor extreme, dar se mentine valoarea centrală la valoarea 8ms;

- Testul nr.3 (Figura 5.14), 100 pași fără *delay* pentru semnal *high* și 1ms pentru semnal *low* (două thread-uri în execuție). Pe grafic este reprezentat doar semnal *low* deoarece timpul pentru semnal *high* este de 0 ms. Prin eliminarea delay-ului se dublează viteza de execuție obținându-se o stabilitate mai bună (vârfurile sunt mult mai mici comparativ cu situațiile anterioare).

- Testul nr.4 (Figura 5.15), 100 pași folosind intructiunea *while* în loc de metoda *sleep()*, se obține astfel un *delay* de 3ms. Concomitent cu acest *thread* rulează un alt *thread* ce ascultă registrul de stare de pe portul paralel. Instructiunea *while* este executată până la parcurgerea a 3ms, condiție obținută prin diferența de timp

folosind metoda *System.currentTimeMillis()*. Thread-ul este setat pe nivel de prioritate *default* (5). Deci sistemul răspunde cu precizie comenzii utilizatorului.

- Testul nr.5 (Figura 5.16). aceleași condiții ca la testul nr. 4 cu setarea priorității thread-ului la nivel 10 (maxim). Se observă că frecvența de apariție a valorilor extreme scade.

- Testul nr.6 (Figura 5.17). 100 pași folosind intructiunea *while* în loc de metoda *sleep()* pentru semnal *low* (fără while pentru semnal *high*). Instructiunea *while* este executată până la parcurgerea a 2ms. Prioritatea thread-ului setată la nivel 10 și concomitent un alt *thread* în execuție. Se obține un răspuns foarte bun al sistemului realizându-se cu exactitate timpul impus simultan cu diminuarea substanțială a amplitudinii vârfurilor. De asemenea și frecvența lor de apariție se diminuează.

La testele: 1. 2a. 2b. 3 - încărcarea procesorului este de aproximativ 5%. pe când la testele: 4. 5. 6 - încărcarea procesorului este de 100%.

Valoarea medie a timpul de acces a portului paralel din java. pe configurația hardware și software pe care s-au efectuat testele anterioare. este de 16512ns.

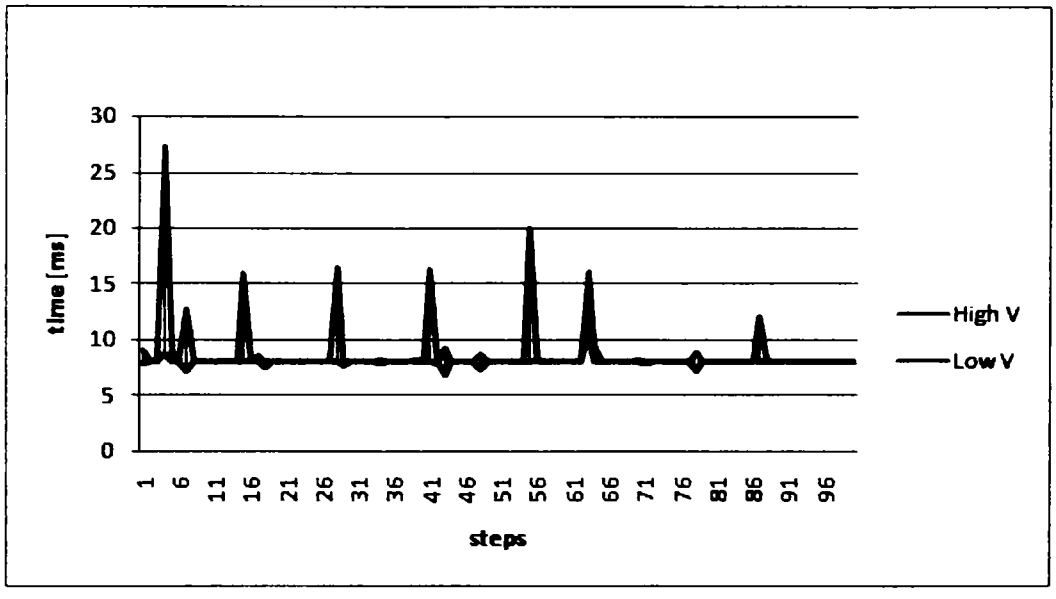


Figura 5.11: Linux - testul nr.1

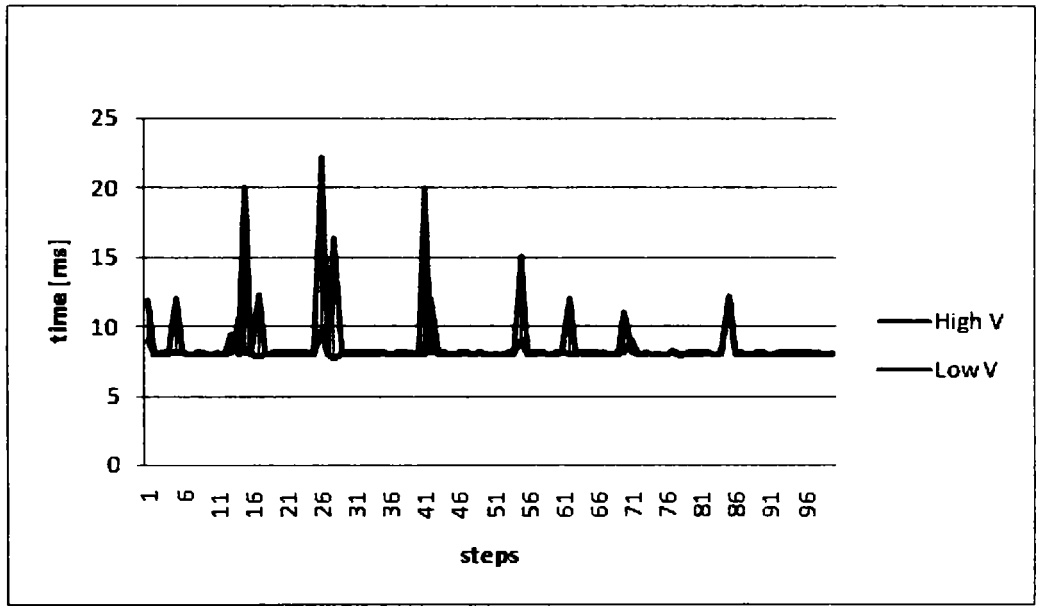


Figura 5.12: Linux - testul nr.2a

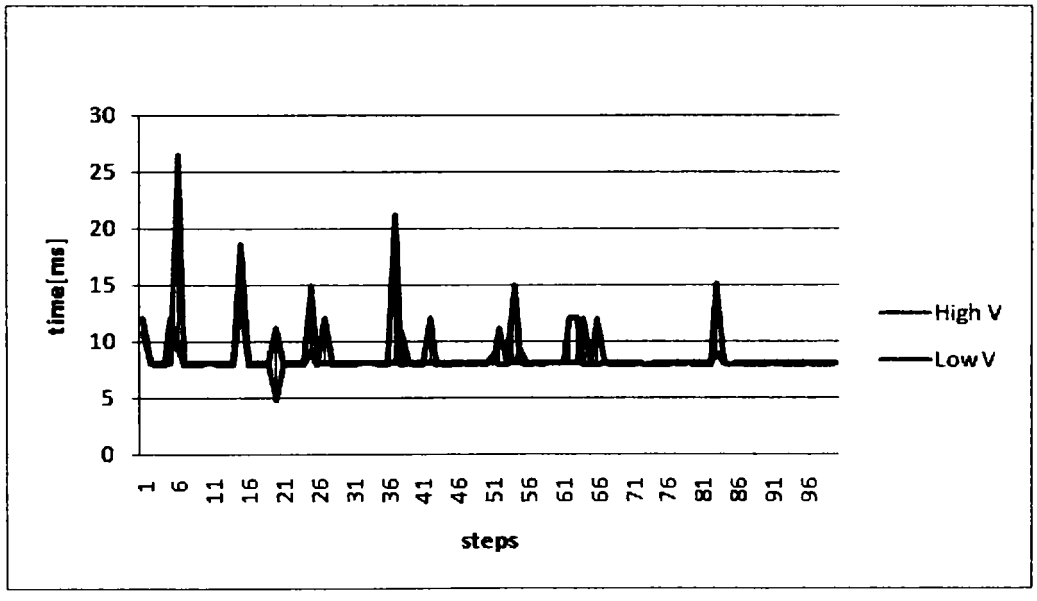


Figura 5.13: Linux - testul nr.2b

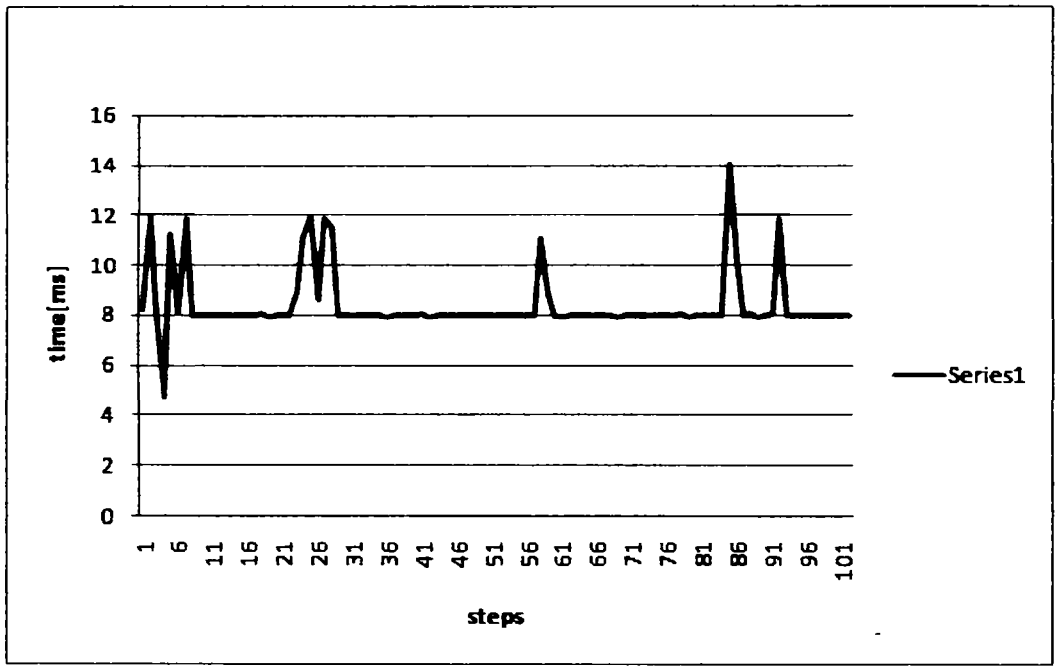


Figura 5.14: Linux - testul nr.3

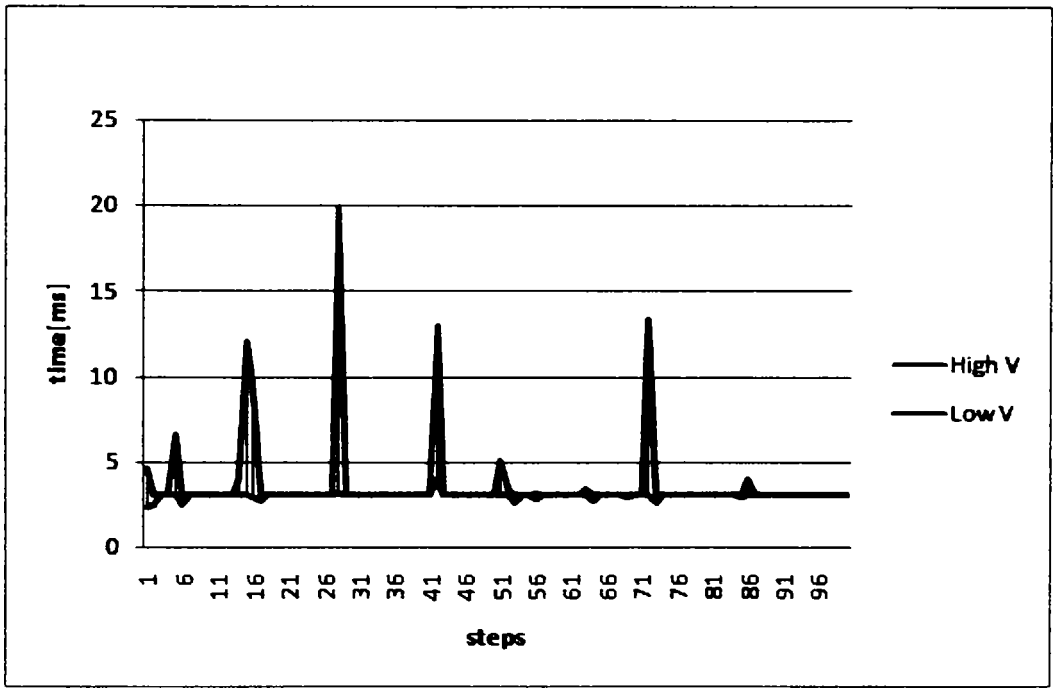


Figura 5.15: Linux - testul nr.4

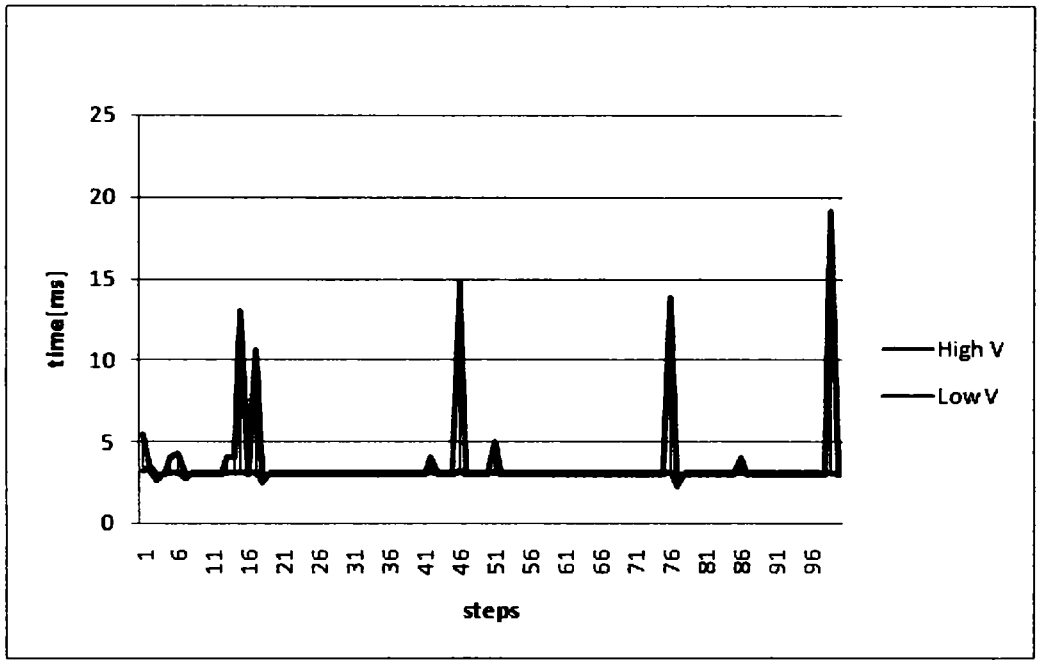


Figura 5.16: Linux - testul nr.5

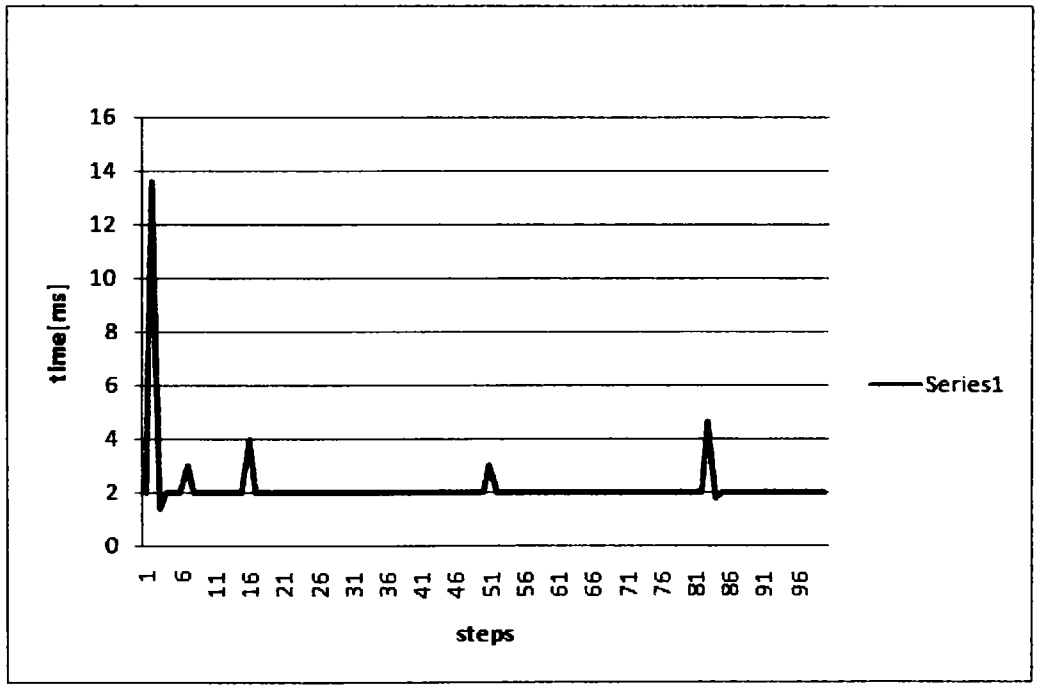


Figura 5.17: Linux - testul nr.6

Configurația hardware a calculatorului pe care s-au executat testele este: CPU AMD 3500, PB ASUS A8V, 1GB Memory (Corsair), PV NVIDIA 5200FX (Leadtek) iar ca sistem de operare: OpenSUSE Linux 10.02.

Următoarele teste au fost efectuate pe aceeași configurație hardware pe care s-au efectuat testele anterioare, dar cu sistem de operare WindowsXP.

- Testul nr.1 (Figura 5.18), 100 pași cu *delay* de 1ms pentru semnal *high* și 1ms pentru semnal *low*. Concomitent cu acest *thread* rulează un alt *thread* ce ascultă registrul de stare de pe portul paralel. Din grafic se observă menținerea aproximativ constantă a timpului de execuție a metodei *sleep()* în jurul valorii de 2ms cu apariția unor valori extreme. Uneori se atinge valoarea de 1ms;

- Testul nr.2a (Figura 5.19), 100 pași cu *delay* de 1ms pentru semnal *high* și 1ms pentru semnal *low* (un singur *thread* în execuție). Din grafic se observă menținerea aproximativ constantă a timpului de execuție a metodei *sleep()* în jurul valorii de 2ms, vârfurile fiind mult mai mici comparativ cu situația anterioară.

- Testul nr.2b (Figura 5.20), același ca și testul nr.2 dar rulat a doua oară. Comparând cele două teste se observă un comportament diferit prin repartizarea și frecvența valorilor extreme, dar se menține valoarea centrală la valoarea 2ms;

- Testul nr.3, 100 pași fără *delay* pentru semnal *high* și 1ms pentru semnal *low* (două thread-uri în execuție). Test nefuncțional, timpul pentru semnal *high* este insuficient pentru comanda controller-ului;

- Testul nr.4 (Figura 5.21), 100 pași folosind intrucțiunea *while* (2ms) în loc de metoda *sleep()*. Test nesatisfăcător deoarece execuția unui astfel de program încarcă procesorul 100%.

Valoarea medie a timpul de acces a portului paralel din java, folosind sistem de operare WindowsXP, este de 1961ns.

În concluzie, folosind sistem de operare Windows se obțin timpi de control mai mici, deci viteze mai mari de deplasare pe axe. Diferența mare a timpilor obținuți în teste, folosind cele două sisteme este justificată de implementare diferită a conceptului de multitasking pe fiecare sistem.

Diferențele de *multitasking* între sistemele de operare Windows și Linux sunt reprezentate în Figurile 5.22, 5.23, 5.24, 5.25. S-au efectuat câte două teste pe fiecare sistem de operare. Testul a constat în rularea concomitentă a două thread-uri, unul ce încarcă procesorul 100% (o buclă *while*) și thread-ul folosit la testele anterioare.

Windows: modulul eșantionului experimental este 31 pentru semnal *high* și 31 pentru semnalul *low* la testul cu prioritate *default*: valoarea 2 pentru semnal *high* și 2 pentru semnalul *low* la testul cu prioritate 10.

Linux: modulul eșantionului experimental este 8 pentru semnal *high* și 8 pentru semnalul *low* la ambele teste:

Ambele sisteme de operare sunt versiuni pe 32biți.

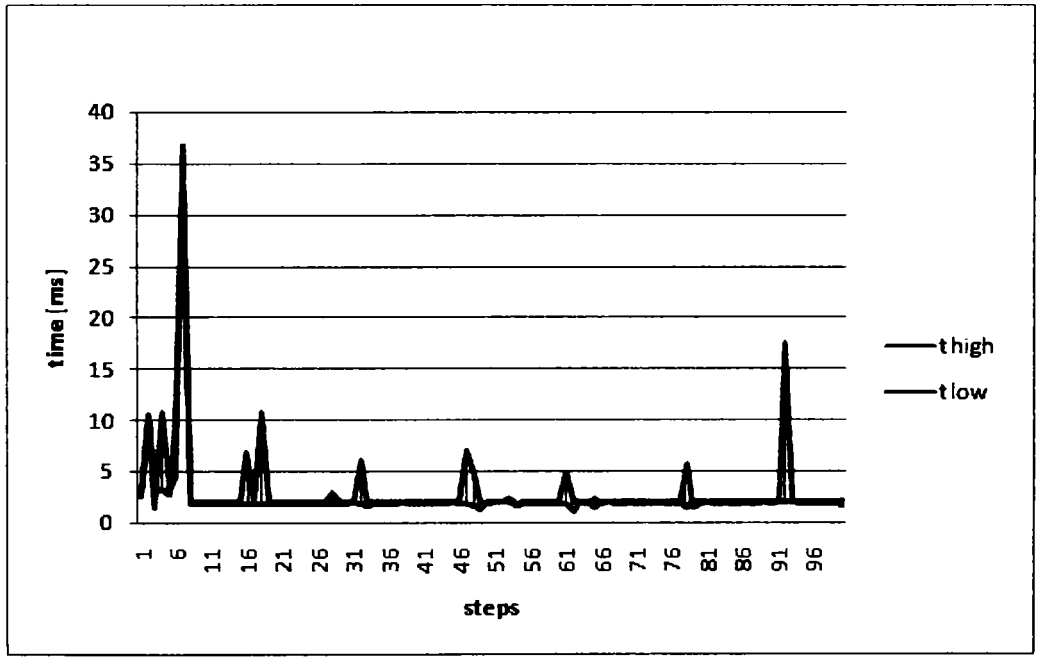


Figura 5.18: Windows - testul nr.1

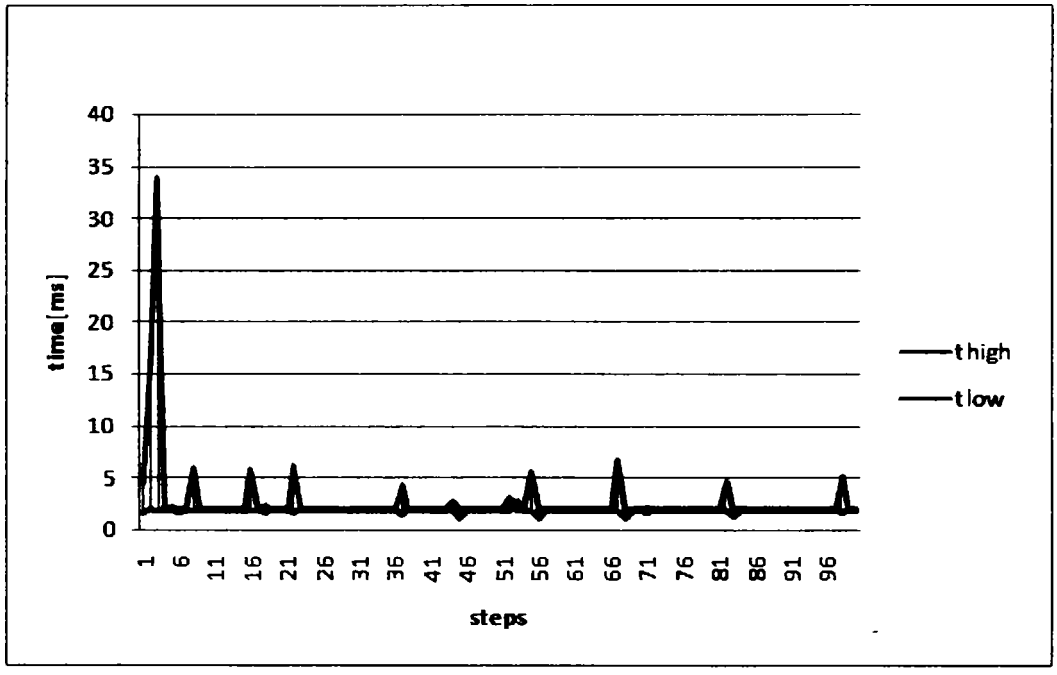


Figura 5.19: Windows - testul nr.2a

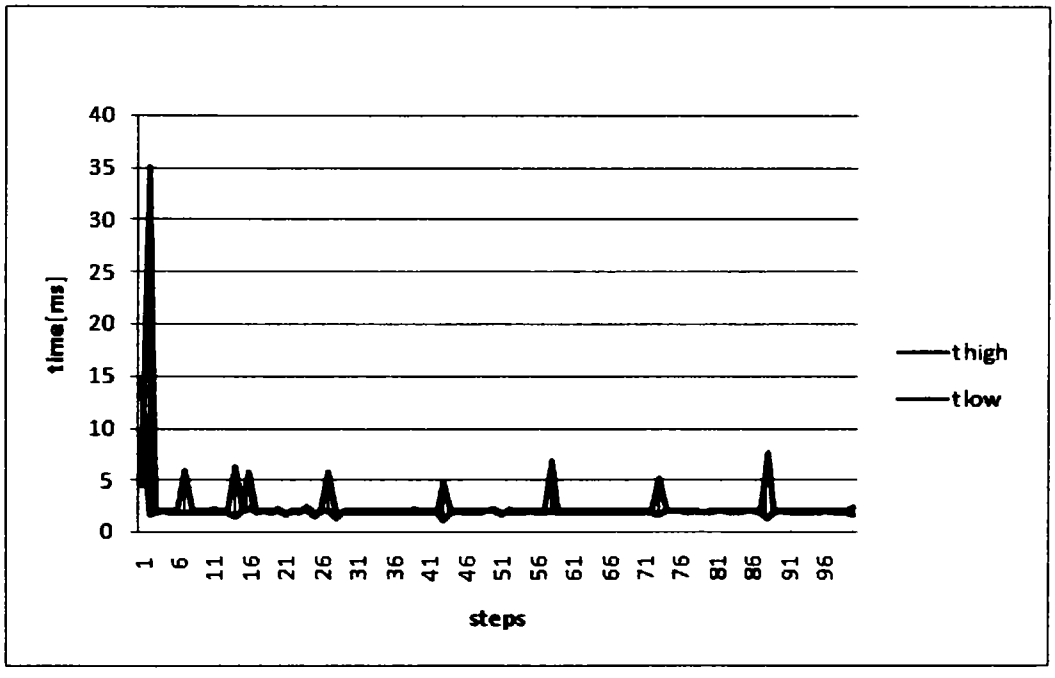


Figura 5.20: Windows - testul nr.2b

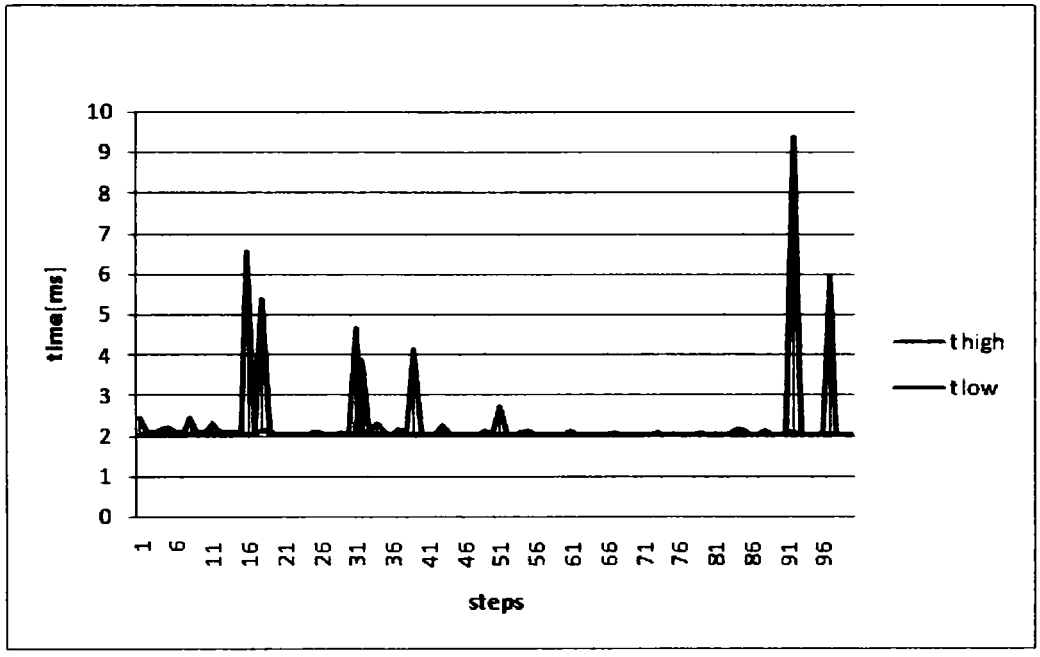


Figura 5.21: Windows - testul nr.4

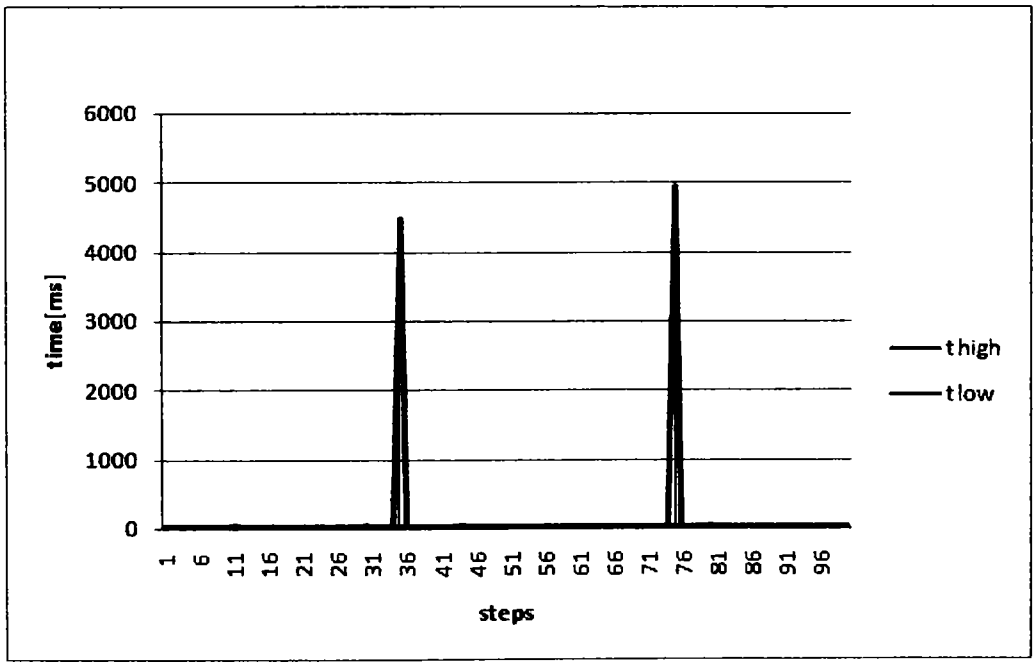


Figura 5.22: Windows 100% CPU, prioritate default

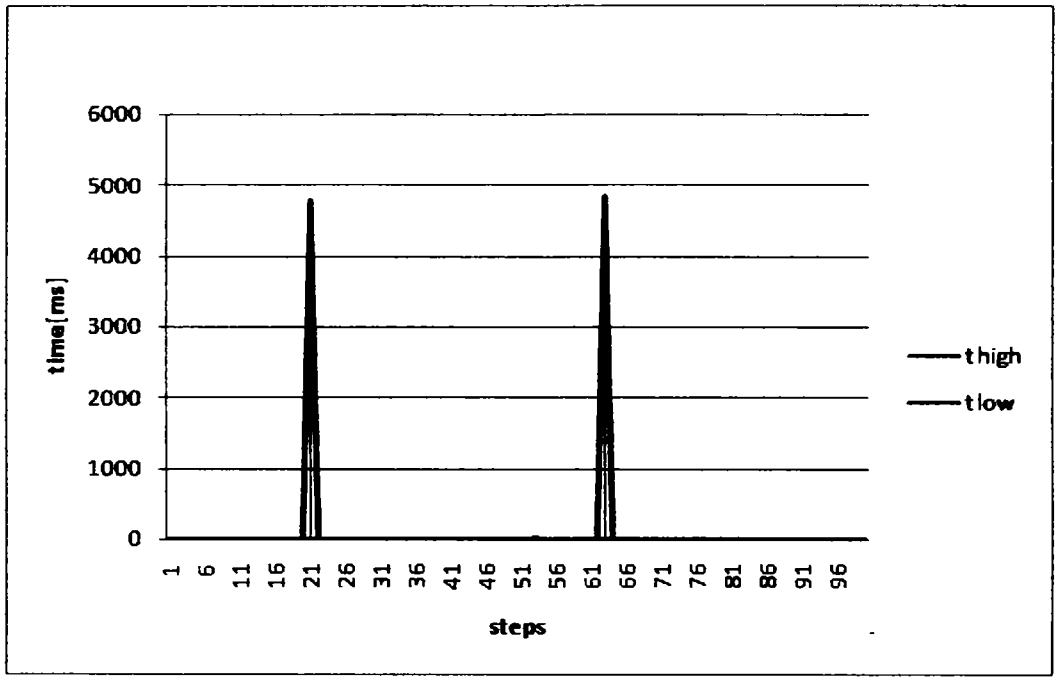


Figura 5.23: Windows 100% CPU. prioritate 10

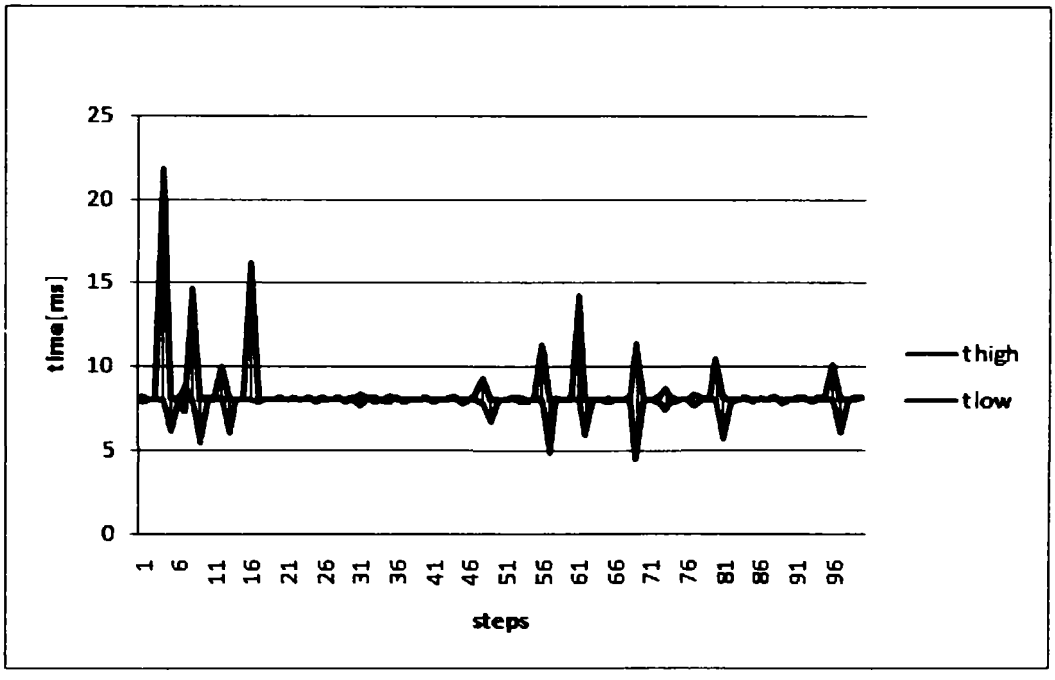


Figura 5.24: Linux 100% CPU. prioritate default

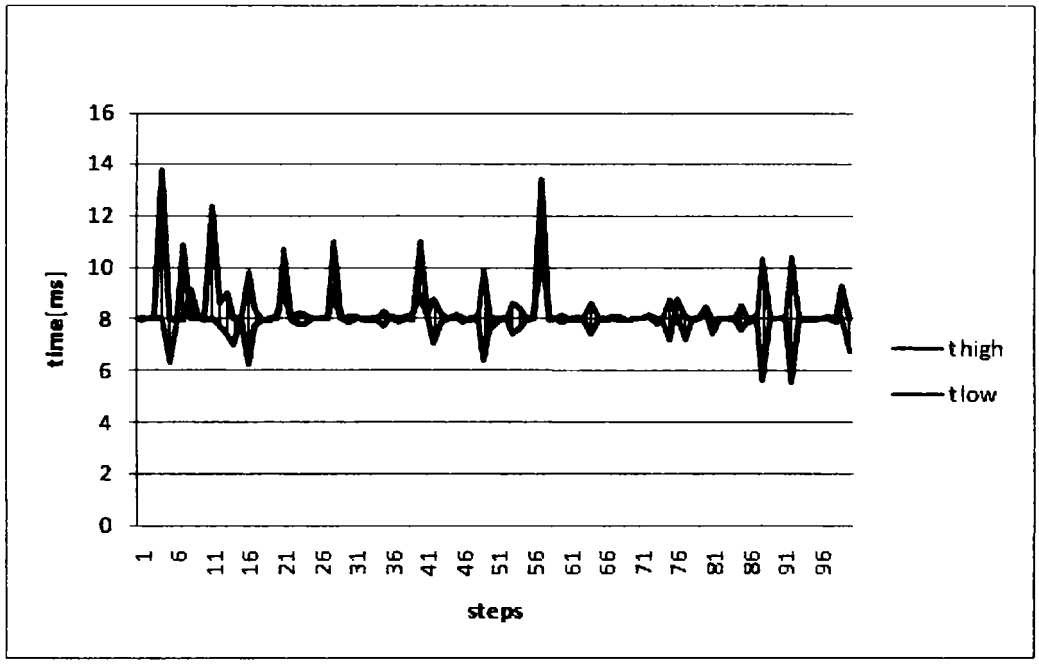


Figura 5.25: Linux 100% CPU, prioritate 10

O soluție pentru rezolvarea acestei probleme (obținerea constant a valorii de *delay* impuse) este folosirea unei extensi cum ar fi: *Sun Java Real-Time System 2.0 (RTSJ)*, extensie ce este compatibilă cu Java SE 5.0. *RTSJ* introduce conceptul a două noi fire de execuție (*threads*): fire de execuție în timp real (*real time thread*) și fire de execuție în timp real "*no-heap*" (un fir de execuție ce nu poate fi întrerupt de sistemul de eliberare a memorie (*garbage collection*)). Aceste fire de execuție au 28 de niveluri de prioritate și contrar Java standard, prioritatea firelor de execuție este strict constrânsă.

Cum scopul dezvoltării aplicației "VRforCAD" și a echipamentului "SphereDevice" este acela de a fi folosite pe calculatoare uzuale (fără harware și software special), o altă soluție de rezolvare a problemei de mai sus este aceea de a folosi motoare pas cu pas cu un unghi pe pas mai mare (ex. 7.5°) și controlere cu micropășire.

O a treia soluție (în dezvoltare) este aceea de a folosi un microcontroller intercalat între portul paralel și echipament. Astfel secvențele de cod ce realizează pașii MPP sunt implementate în microcontroler rezolvând astfel problema timpilor de comandă a celor trei controlere MPP. Pentru acest scop s-au luat în considerare trei microcontrolere:

- JStamp - produs de Systronix Inc. [84] - programabil în RealTime Java (preț

aproximativ 300 euro). JStamp execută aproximativ 3.000.000 *Java byte codes per second.*

- BASIC Stamp 2e - *powered by the Ubicom SX microcontroller* produs de Parallax Inc. [65] - programabil în PBASIC (*Parallax BASIC*) (pret aproximativ 80 euro). BASIC Stamp 2e execută aproximativ 4.000 instructiuni sec.

- BasicAtom Pro 28-M, produs de Basic Micro [6] - programabil în Mbasic, montat pe un *board* Mini-ABB (Atom Bot Board) produs de Lynxmotion [59] (pret total, aproximativ 70 euro).

Din considerente legate de pret, autorul acestei teze a dezvoltat cea de-a treia variantă: BasicAtom Pro 28-M. În figura 5.26, este prezentat microcontrolerul intercalat între echipamentul "SphereDevice" și calculator (montajul are în componentă doi conectori de tip DB25, mamă și tată).

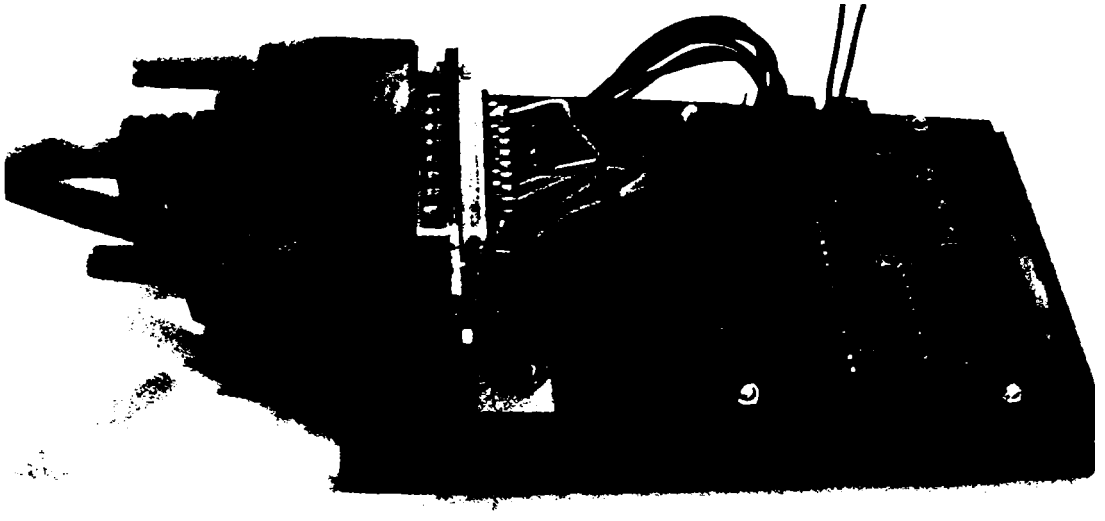


Figura 5.26: Montaj cu microcontroller de tip BasicAtom Pro 28-M conectat la echipamentul "SphereDevice".

Următorul cod sursă (programat în limbajul Atom Pro, limbaj bazat pe Mbasic (BASIC Micro)) este cel folosit la programarea microcontrolerului. Astfel pașii motoarelor pas cu pas sunt realizați de microcontroler eliminând problemele ce apar utilzând metoda *sleep()* în Java.

```
;  * File     : spheredevice.bas
;
; Copyright (C) 2007 Daniel Cioi <dan.cioi@vrforcad.org>
;
;                                www.vrforcad.org
;
;   This file is part of VRforCAD.
;
; VRforCAD is free software: you can redistribute it and/or modify
; it under the terms of the GNU General Public License as published by
; the Free Software Foundation, either version 3 of the License, or
; (at your option) any later version.
;
; VRforCAD is distributed in the hope that it will be useful,
; but WITHOUT ANY WARRANTY; without even the implied warranty of
; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
; GNU General Public License for more details.
;
; You should have received a copy of the GNU General Public License
; along with VRforCAD.  If not, see <http://www.gnu.org/licenses/>.
stepdelay var word
stepdelay = 1 ; X 2 = delay per step
pinnr var nib
input P0
...
input P5
output P6
...
output P11
loop
        IF IN5 = 1 AND IN4 = 0 then
                low P10
                pinnr = 11 ; z minus
                gosub step
        endif
        IF IN5 = 1 AND IN4 = 1 then
                high P10
                pinnr = 11 ; z plus
                gosub step
        endif
        IF IN3 = 1 AND IN2 = 0 then
                low P8
                pinnr = 9 ; y minus
                gosub step
        endif
        IF IN3 = 1 AND IN2 = 1 then
                high P8
                pinnr = 9 ; y plus
                gosub step
        endif
        IF IN1 = 1 AND IN0 = 0 then
```

```
                low P6
                pinnr = 7 ; x minus
                gosub step
        endif
        IF IN1 = 1 AND IN0 = 1 then
                high P6
                pinnr = 7 ; x plus
                gosub step
        endif
goto loop
step
        high pinnr
        pause stepdelay
        low pinnr
        pause stepdelay
return
END
```

5.6 Haptic feedback cu echipamentul "SphereDevice"

Autorul a menționat la începutul acestui capitol că echipamentul "SphereDevice" este un echipament *haptic* cu *feedback*. Feedback-ul se obține în urma modificării timpului de păsire a motoarelor pas cu pas. Următoarea secvență de cod prezintă algoritmul conceput de autor pentru obținerea feedback-ului. Autorul menționează că algoritmul este de complexitate redusă, urmând ca pe viitor să se realizeze un alt algoritm mai complex. În figura 5.27 este reprezentat graficul variației timpului de păsire, valori obținute cu clasa de mai jos.

```
/*
 * File     : DeformationValues.java
 *
 * Copyright (C) 2006-2007 Daniel Cioi <dan.cioi@vrforcad.org>
 *
 *                  www.vrforcad.org
 *
 *   This file is part of VRforCAD.
 *
 * VRforCAD is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * VRforCAD is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with VRforCAD.  If not, see <http://www.gnu.org/licenses/>
 *
 */
package org.vrforcad;

import java.util.ArrayList;
/**
 * This class return the array for steps delay values.
 *
 * @version 1.1
 * @author Daniel Cioi <dan.cioi@vrforcad.org>
 */
public class DeformationValues {

    private int lenghtDeformation;      //mm
    private float stepDelay;            //time step ms
    private int liniarDeplasation = 40;         //mm
    private float degreePerStep = 1.5f;     //degree
    private float liniarDeplasationPerStep;
    private int incrementstep;          //size time interval
    private float velocityTouch;    //viteza de coliziune
    private int timeIncrem;
    private ArrayList<Integer> delays = new ArrayList<Integer>();
    private int[] delaysArray;

    public DeformationValues(int lenghtDeformation, int stepDelay){
        this.stepDelay = stepDelay;
        this.lenghtDeformation = lenghtDeformation;
    }
    int[] getValues() {
    liniarDeplasationPerStep = (liniarDeplasation*degreePerStep)/360;
    incrementstep = (int)(lenghtDeformation/liniarDeplasationPerStep);
    velocityTouch = 10/(stepDelay*60);
    float velocityTouchIncrem = velocityTouch/incrementstep;
    for(int ii=0; ii<incrementstep; ii++){
        float velocityTouchTemp = velocityTouch - (velocityTouchIncrem*ii);
        timeIncrem = (int)(liniarDeplasationPerStep/velocityTouchTemp);
        delays.add(timeIncrem);
        }
    delaysArray = new int[delays.size()];
    for(int i=0; i<delays.size();i++){
        delaysArray[i] = delays.get(i);
        }
        return delaysArray;
    }
}
```

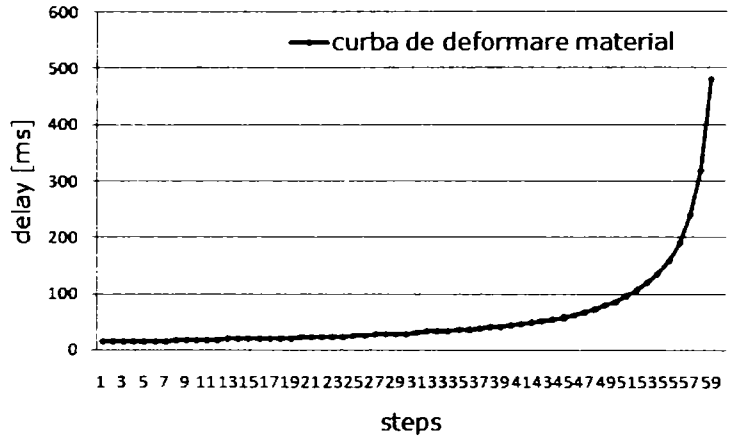


Figura 5.27: Valori pentru timpii de păsire în contact cu o suprafață.

5.7 Configurarea echipamentului "SphereDevice" în aplicația "VRforCAD"

Din meniul Aplicației "VRforCAD", se alege fereastra de dialogul "*Preference*" (Figura 5.28). Acest dialog permite utilizatorului alegera dispozitivului de lucru (*Input device*). Aplicația "VRforCAD" oferă posibilitatea utilizării comcomitent a mouse-ului și a echipmanmetului "SphereDevice".

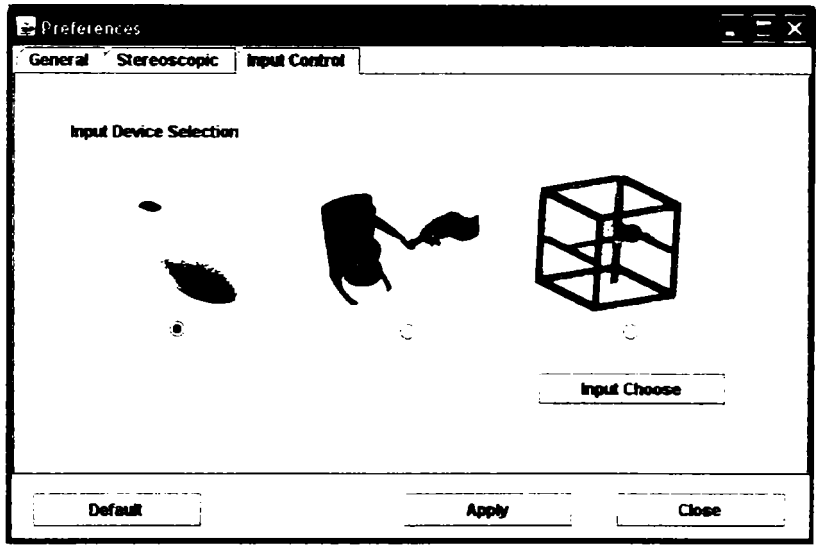


Figura 5.28: Fereatra de dialog *Preference* a aplicației "VRforCAD"

În figura 5.29 sunt identificate butoanele care sunt folosite împreună cu echipamentul "SphereDevice" . Apăsarea butonului 1 are ca scop deschiderea meniului virtual din care se alege prin operația de picking sfera pentru paint sau cea pentru deformare suprafețe. Sfera aleasă, poate fi controlată cu ajutorul mouse-ului sau prin intermeniul echipamentului "SphereDevice". Butonul 2 activează echipamentul (inclusiv alimentarea electrică). Automat se execută operația de aducere în poziția de acasă după care echipamentul este pregătit pentru utilizare. O a doua apăsare pe butonul 2 are ca rezultat oprirea echipamentului (inclusiv alimentarea electrică). De asemenea dacă echipamentul nu este folosit, după o perioadă de 10 min se execută comanda pentru deschiderea circuitului de alimentare.

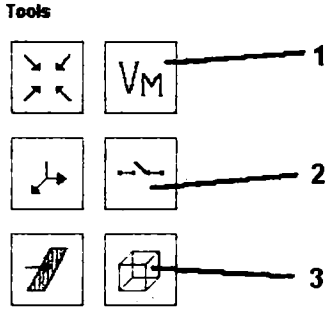


Figura 5.29: Butoane din meniul aplicației "VRforCAD"

Apăsarea butonului 3 are ca răspuns la evenimentul generat, deschiderea meniului de configurare a echipamentului. Următoarea figură (Figura 5.30) prezintă un *screenshot* a meniului de configurarea a echipamentului "SphereDevice" în aplicația "VRforCAD".

Configurarea spațiului de lucru are ca scop limitarea deplasării uneltei sferă doar în spațiul definit. De asemenea există și o opțiune automată (Auto definition) care setează automat spațiul de lucru al echipamentului în funcție de dimensiunile modelului încărcat în scena virtuală.

Setarea motoarelor pas cu pas este utilă atunci când echipamentul nu este construit cu trei motoare identice (așa cum este și cazul de față).

Butonul *Home position*, permite executarea manual a operației de aducere în poziția de acasă (operația este executată automat la pornirea echipamentului). Această

optiune este utilă în momentul în care unul din motoare pierde pași din diverse motive cum ar fi opunerea de forță din partea utilizatorului.

Butonul *Save* salvează în fișierul de configurare al aplicației datele introduse în scopul disponibilității acestora la o nouă pornire a aplicației.

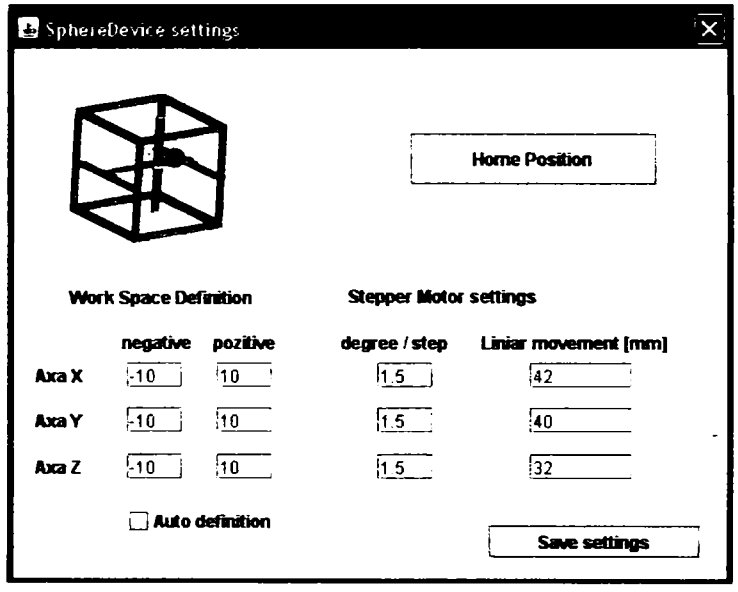


Figura 5.30: Fereastra de dialog pentru configurare a echipamentului "SphereDevice" în aplicatia "VRforCAD"

5.8 Concluzii

În acest capitol sunt prezentate contribuțiile autorului tezei la dezvoltarea unui echipament cu *feedback* haptic. Astfel pe parcursul capitolului s-au prezentat detalii constructive, principiul de funcționare din punct de vedere electronic, respectiv avantajele și dezavantajele acestui echipament.

Spațiul de lucru al echipamentului este în concordanță cu spațiul virtual. Aplicația "VRforCAD" permite utilizatorului să folosească mouse-ul pentru manipularea scenei virtuale concomitent cu utilizarea echipamentului "SphereDevice" pentru operații de deformare a suprafețelor respectiv operații de atribuire de culoare modelelor.

În final se precizează faptul că autorul a dorit realizarea unui echipament cu un cost de producție scăzut și o mare flexibilitate în utilizare.

Capitolul 6

Concluzii finale

În cele ce urmează se prezintă concluziile finale care se pot trage din precedentele capitole ale acestei teze. De asemenea se prezintă un rezumat sistematizat al capitolelor anterioare. În acest sens este expus din diferite abordări conținutul prezentei teze, precum și activitatea autorului.

Tratarea corespunzătoare a subiectului tezei "Contribuții la utilizarea Realității Virtuale în Proiectarea Asistată de Calculator" solicită acoperirea unui domeniu vast de cunoștințe. Contribuțiile și aplicațiile autorului sunt rezultatul unei activități de cercetare focalizate pe tema tezei în ultimii patru ani în cadrul colectivului Departamentului de Mecatronică al Facultății de Mecanică din Timișoara.

Din parcurgerea lucrării se desprind următoarele:

- după cum rezultă din dispunerea în timp a publicațiilor, la tema prezentată s-a lucrat pe parcursul a mai multor ani. Studiind temele se poate observa că informațiile din atâtea domenii au necesitat parcurgerea bazelor de cunoștințe din diferite puncte de vedere pentru a reuși sistematizarea și structurarea lor;

- autorul s-a familiarizat cu domenii foarte diverse ce țin de realitate virtuală, proiectarea asistată de calculator și limbaje de programare;

- pe parcursul tezei sunt prezentate o serie de contribuții teoretice și aplicative;

- lucrarea are o structură coerentă ce servește scopului propus, de a prezenta contribuțiile

autorului la utilizarea realității virtual în proiectarea asistată de calculator.

6.1 Structura tezei

În capitolul 1, se prezintă problematica abordată în prezenta teză precum și principalele informații legate de aceasta. Sunt prezentate definiții și noțiuni introductive a ceea ce înseamnă Realitate Virtulală (VR) și Proiectare Asistată de Calculator (CAD). De asemenea se evidențiază beneficiile utilizării realității virtuale împreună cu sistemele de proiectare asistată de calculator. În continuare se prezintă stadiul actual al sistemelor VR - CAD. Capitolul se încheie prin prezentarea obiectivelor prezentei teze de doctorat.

În capitolul 2, sunt descrise limbajele de realitate virtuală ce permit dezvoltarea de aplicații *Open Source*. În urma acestei prezentări, autorul a ales să folosească limbajul Java și API-ul Java3D în scopul dezvoltării unei aplicații *software* intitulată "VRforCAD", parte a sistemului VR-CAD.

Capitolul 3, prezintă noțiuni despre sistemele VR-CAD. În cadrul capitolului se face o trecere în revistă a celor mai utilizate formate de fișier pentru schimburi între sistemele CAD. Conform celor prezentate în acest capitol, autorul tezei a ales folosirea fișierelor neutre DXF și STL în scopul interschimbabilității modelelor între sistemele CAD-VR.

Capitolul 4, prezintă contribuțiile majore ale autorului la dezvoltarea și testarea unui *software* pentru sistemul VR-CAD. Astfel după o scurtă prezentare a aplicațiilor VR și CAD *Open Source*, se prezintă pe larg aplicația "VRforCAD" dezvoltată de autor utilizând limbajul de programare Java și API-ul Java3D. Autorul menționează că aplicația "VRforCAD" este de asemenea Open Source sub licență GPL v3.

Capitolul 5, prezintă contribuțiile majore ale autorului la proiectarea și realizarea unui echipament de tip haptic cu *feedback* pentru sistemul VR-CAD. Astfel sunt prezentate detalii constructive, principiul de funcționare, modul de comandă al motoarelor (cu și fără microcontroler) respectiv avantajele și dezavantajele echipamentului "SphereDevice" conceput de autor.

Ultimul capitol al lucrării, prezintă concluziile finale, precum și un rezumat al problemelor abordate în prezenta teză.

6.2 Contribuții teoretice și aplicative

- analiza critică a limbajelor de realitate virtuală ce permit dezvoltarea de aplicații *Open Source*, urmărește din perspectiva subiectului abordat, realizarea unei fundamentări a limbajului de realitate virtuală selectat pentru dezvoltarea aplicației "VRforCAD".

- prezentarea critică a aspectelor ce privesc comunicarea între CAD și VR, în contextul schimbului de fișiere.

- reliefarea necesității folosirii unui format de fișier neutru pentru interschimbabilitatea modelelor CAD, l-a determinat pe autor să realizeze un studiu amplu cu privire la formate de export modele 3D existente. Finalitatea acestui studiu îl constituie alegerea a două formate de fișier neutru: DXF și STL pentru dezvoltarea modulului de import/export a aplicației "VRforCAD".

- sintetizarea unor aspecte teoretice și practice legate de implementarea sistemelor VR-CAD în activitatea de proiectare, vizualizare și manipulare a obiectelor tridimensionale.

- studiul realizat de către autor cu privire la aplicațiile VR și CAD *Open Source*, a scos în evidență lipsa unei aplicații *Open Source* în scopul realizării unui *bridge* între aplicații CAD și aplicații VR. În acest scop, autorul a dezvoltat aplicația "VRforCAD" utilizând limbajul de programare Java și API-ul Java3D (aplicația conține 75 clase ce însumează peste 12000 linii de cod), ce prezintă un avantaj major din punct de vedere al portabilității pe diferite sisteme de operare (Windwos, Unix, Linux, Macintosh). Interfața grafică cu utilizatorul a aplicației "VRforCAD", a fost proiectată sub o formă "prietenoasă", ușor de înțeles de către utilizator.

- din punct de vedere al implementării unui mediu de lucru colaborativ, aplicația "VRforCAD" cuprinde un modul de comunicare prin internet cu o bază de date (MySQL) în vederea păstrării modelelor 3D în respectiva bază de date cu scopul accesibilității modelelor 3D din mai multe locații internet; de asemenea a fost proiectat, realizat și implementat un server "VRforCAD_CWES" cu facilități ce permite

mai multor utilizatori modificarea concomitent a aceluiași model CAD, realizându-se operarea într-un mediu virtual colaborativ.

- aplicația "VRforCAD" împreună cu echipamentul "SphereDevice" constituie un sistem original VR-CAD. Cele două componente software și hardware ale sistemului VR-CAD, reprezintă contribuțiile originale majore ale autorului în atingerea scopului propus în ceea ce privește utilizarea Realității Virtuale în Proiectarea Asistată de Calculator.

- construția echipamentului "SphereDevice" a evidențiat problemele ce apar la comanda motoarelor pas cu pas utilizând limbajul de programare Java. Astfel, referitor la comanda motoarelor pas cu pas, autorul a implementat două metode de comandă:

  - prima, utilizând cod java și evidențiind soluțiile ce pot rezolva problemele de temporizare între pași;

  - a doua, prin adăugarea unui microcontroler, rezolvând astfel problemele de temporizare, obținându-se rezultatele dorite. Programarea microcontrolerului a fost făcută în limbajul Mbasic.

- referitor la varietatea domeniilor de aplicabilitate se remarcă faptul că aplicația "VRforCAD" include module de utilizare în robotică, medicină și vizualizare.

- o parte a rezultatelor cercetării prezentate în această teză se găsesc și la adresa `http://www.vrforcad.org`.

- codul sursă al aplicației "VRforCAD" (inclusiv codul sursă pentru comanda echipamentului "SphereDevice") este înregistrat sub licență *GNU GENERAL PUBLIC LICENSE Version 3* și poate fi obținut de la adresa `http://www.vrforcad.org` din secțiunea Download sau de la adresa `http://sourceforge.net/projects/vrforcad/`

Se remarcă interesul științific și economic al sistemului VR-CAD realizat, identificându-se domenii de cercetare și soluții personale.

Pentru realizarea acestei teze s-au folosit următoarele softuri open source: MikTex și TeXnicCenter (IDE) pentru tehnoredactarea tezei; Gimp pentru prelucrarea imaginilor; limbajul de programare Java și API-ul Java3D, Eclipse (IDE) pentru dezvoltarea aplicație "VRforCAD".

Activitatea de cercetare în acest domeniu a autorului tezei nu se oprește aici, urmând ca în viitor aplicația VRforCAD să conțină noi funcționalități și extinderea suportului pentru alte echipamente de realitate virtuală. Un prim pas va fi adăugarea de suport pentru mouse-uri din categoria *3DConnection* produși de Logitech. Circuitele de comandă a motoarelor pas cu pas din componența echipamentului "SphereDevice" vor fi înlocuite cu circuite de comandă cu micropășire.

ANEXA A

```
/*
 * File     : ConvertX3D.java
 *
 * Copyright (C) 2006-2007 Daniel Cioi <dan.cioi@vrforcad.org>
 *
 *                    www.vrforcad.org
 *
 *   This file is part of VRforCAD.
 *
 * VRforCAD is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * VRforCAD is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. - See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with VRforCAD.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

package org.vrforcad;

/**
 * This class parse the X3D file and build the geometry.
 *
 * @version 1.1
 * @author Daniel Cioi <dan.cioi@vrforcad.org>
 */

import java.io.File;
import java.util.ArrayList;
import javax.media.j3d.IndexedTriangleArray;
import javax.media.j3d.Material;
import javax.media.j3d.PolygonAttributes;
import javax.media.j3d.Shape3D;
import javax.media.j3d.Transform3D;
import javax.media.j3d.TransformGroup;
import javax.vecmath.Vector3f;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
```

```
import com.sun.j3d.utils.geometry.GeometryInfo;
import com.sun.j3d.utils.geometry.NormalGenerator;


public class ConvertX3D {

    static String group = "Group";
    static String transform = "Transform";
    static String shape = "Shape";
    static String appearance = "Appearance";
    static String material = "Material";
    static String indexedFaceSet = "IndexedFaceSet";
    static String coordinate = "Coordinate";
    static String normal = "Normal";

    public static ArrayList<Float> coodonates = new ArrayList<Float>();
    public static ArrayList<Float> normals = new ArrayList<Float>();
    public static ArrayList<Integer> indices = new ArrayList<Integer>();
    public static ArrayList<Integer> normalindices = new ArrayList<Integer>();
    public static ArrayList<Float> transformPosition = new ArrayList<Float>();
    public static ArrayList<Float> rotationPosition = new ArrayList<Float>();
    public static ArrayList<Float> scalePosition = new ArrayList<Float>();
    public static ArrayList<Float> tempValuesFloat = new ArrayList<Float>();
    public static ArrayList<Integer> tempValuesInt = new ArrayList<Integer>();

    public static ArrayList<Float> ambientIntensity = new ArrayList<Float>();
    public static ArrayList<Float> diffuseColor = new ArrayList<Float>();
    public static ArrayList<Float> shininess = new ArrayList<Float>();
    public static ArrayList<Float> specularColor = new ArrayList<Float>();

    static String shapeName;

    public static CADmodelAppearance shapeAppearance =
        new CADmodelAppearance();;
    static float colortest=0;

    static IndexedTriangleArray geom;
    static Shape3D result;
    static Transform3D shapePosition;
    static TransformGroup transformPos;

    static TransformGroup tg;

    static int max = 900000;
    static float[] coordonateShapeLoad = new float[max];
    static float[] coordonateShapeLoadNormals = new float[max];
    static int[] coordonateShapeLoadIndices = new int[max];
    static int[] coordonateShapeLoadNormalIndices = new int[max];

    static File x3dFilePath = null;
```

```
public ConvertX3D(File filePath){

      x3dFilePath = filePath;
      scan();
      }


static void splitstringfloat(String tosplit){
      tosplit=tosplit.trim();
      tosplit=tosplit.replaceAll(",", "");
      String[] scanstring=tosplit.split(" ");

      for (int itr=0; itr< scanstring.length;itr++){
            tempValuesFloat.add(itr, Float.parseFloat(scanstring[itr]));
            }
}

static void splitstringint(String tosplit){
      tosplit=tosplit.trim();
      String[] scanstring=tosplit.split(" ");

      for (int itr=0; itr< scanstring.length;itr++){
            int itrif=0;
            if (Integer.parseInt(scanstring[itr])!=-1){
            tempValuesInt.add(itrif, Integer.parseInt(scanstring[itr]));
            itrif++;
            }
      }
}

static void recursivelysearch(Node node){

      Node no = node.getParentNode();
      String asd = no.getNodeName();

      if ( asd==group){
            shapeName=((Element) no).getAttribute("metadata");
      }

      if ( asd==transform){
//translation
            String trans = ((Element) no).getAttribute("translation");
            if(trans.isEmpty()){
      }
      else{
            splitstringfloat(trans);
            for(int i=0;i<tempValuesFloat.size();i++){
                  transformPosition.add(i, tempValuesFloat.get(i));
            }
            tempValuesFloat.clear();
```

```
            cleanTempValues();
        }
//rotation
        String rot = ((Element) no).getAttribute("rotation");
        if(rot.isEmpty()){
            }
        else{
            splitstringfloat(rot);
            for(int i=0;i<tempValuesFloat.size();i++){
            rotationPosition.add(i, tempValuesFloat.get(i));
        }
            tempValuesFloat.clear();
            cleanTempValues();
        }
//scale
        String scale = ((Element) no).getAttribute("scale");
        if(scale.isEmpty()){
            }
        else{
            splitstringfloat(scale);
            for(int i=0;i<tempValuesFloat.size();i++){
                scalePosition.add(i, tempValuesFloat.get(i));
            }
            tempValuesFloat.clear();
            cleanTempValues();
            }
        }

// material
        if ( asd==shape){
    NodeList Material = node.getChildNodes();
        String asdMat = Material.item(1).getNodeName();

        if ( asdMat==material){
        //ambient intensity
            String ambientIntens =
            ((Element) no).getAttribute("ambientIntensity");
            if(ambientIntens.isEmpty()){
                }
            else{
                splitstringfloat(ambientIntens);
                for(int i=0;i<tempValuesFloat.size();i++){
                    ambientIntensity.add(i, tempValuesFloat.get(i));
                }
            tempValuesFloat.clear();
            cleanTempValues();
        }
        //diffuseColor
            String diffuseCol = ((Element) no).getAttribute("diffuseColor");
            if(diffuseCol.isEmpty()){
                }
```

```
            else{
                splitstringfloat(diffuseCol);
                for(int i=0;i<tempValuesFloat.size();i++){
                    diffuseColor.add(i, tempValuesFloat.get(i));
                }
            tempValuesFloat.clear();
            cleanTempValues();
        }
        //shininess
            String shin = ((Element) no).getAttribute("shininess");
            if(shin.isEmpty()){
                }
            else{
                splitstringfloat(shin);
                for(int i=0;i<tempValuesFloat.size();i++){
                    shininess.add(i, tempValuesFloat.get(i));
                }
            tempValuesFloat.clear();
            cleanTempValues();
        }
        //specularColor
            String specularC = ((Element) no).getAttribute("specularColor");
            if(specularC.isEmpty()){
                }
            else{
                splitstringfloat(specularC);
                for(int i=0;i<tempValuesFloat.size();i++){
                    specularColor.add(i, tempValuesFloat.get(i));
                }
            tempValuesFloat.clear();
            cleanTempValues();
            }
        }
}//end if Appearance

if ( asd==shape){
        }

//<<
        String sceneStr="Scene";
        if (asd.compareTo(sceneStr)!=0){
            recursivelysearch(no);
        }
 }


static void loadGeometry(){

for(int i=0; i< coodonates.size();i++){
        coordonateShapeLoad[i] = coodonates.get(i);
        }
```

```
for(int i=0; i< normals.size();i++){
      coordonateShapeLoadNormals[i] = normals.get(i);
      }
for(int i=0; i< indices.size();i++){
      coordonateShapeLoadIndices[i] = indices.get(i);
      }
for(int i=0; i< indices.size();i++){
      coordonateShapeLoadNormalIndices[i] = indices.get(i);
      }
}

static void cleanTempValues(){
      tempValuesFloat.removeAll(tempValuesFloat);
      tempValuesInt.removeAll(tempValuesInt);
}

public static void scan() {

try {

      DocumentBuilderFactory factory =
            DocumentBuilderFactory.newInstance();
  DocumentBuilder loader = factory.newDocumentBuilder();

      Document document = loader.parse(x3dFilePath);

//>>>>>>>
  NodeList shapeSearch =
      document.getElementsByTagName("IndexedFaceSet");

      for (int s = 0; s < shapeSearch.getLength(); s++) {
            Node aaa = shapeSearch.item(s).getParentNode();
            String asd = aaa.getNodeName();
            recursivelysearch(aaa);

      //IndexedFaceSet
      //coordonate index
      String indexFace =
            ((Element)shapeSearch.item(s)).getAttribute("coordIndex");
            splitstringint(indexFace);
            for(int i=0;i<tempValuesInt.size();i++){
                  indices.add(i, tempValuesInt.get(i));
                  }
            cleanTempValues();
      //normal index;
      String normalindex =
            ((Element)shapeSearch.item(s)).getAttribute("normalIndex");
            splitstringint(normalindex);
            for(int i=0;i<tempValuesInt.size();i++){
                  normalindices.add(i, tempValuesInt.get(i));
                  }
```

```
            cleanTempValues();

        //Coordonate
            NodeList shapeSearchCoordinate =
                document.getElementsByTagName("Coordinate");
          String coord =
            ((Element)shapeSearchCoordinate.item(s)).getAttribute("point");
          splitstringfloat(coord);
          for(int i=0;i<tempValuesFloat.size();i++){
            coodonates.add(i, tempValuesFloat.get(i));
           }
          cleanTempValues();

        //Normals
            NodeList shapeSearchNormal =
                document.getElementsByTagName("Normal");
          String normalsStr =
            ((Element)shapeSearchNormal.item(s)).getAttribute("vector");
          splitstringfloat(normalsStr);
          for(int i=0;i<tempValuesFloat.size();i++){
            normals.add(i, tempValuesFloat.get(i));
           }
          cleanTempValues();

        //add geometry;

        loadGeometry();

    int nrV = coordonateShapeLoad.length/3;
    int nrI = coordonateShapeLoadIndices.length;
    geom = new IndexedTriangleArray(nrV , IndexedTriangleArray.COORDINATES
        | IndexedTriangleArray.NORMALS , nrI);
    geom.setCoordinates(0, coordonateShapeLoad);
    geom.setCoordinateIndices(0, coordonateShapeLoadIndices);
    GeometryInfo gi = new GeometryInfo(geom);
    NormalGenerator normalG = new NormalGenerator();
    normalG.generateNormals(gi);
        geom =  (IndexedTriangleArray) gi.getIndexedGeometryArray();

        PolygonAttributes PoligonFace = new PolygonAttributes();
        PoligonFace.setCullFace(PolygonAttributes.CULL_NONE);
        shapeAppearance.setPolygonAttributes(PoligonFace);

        Material mat = new Material( );
        if(diffuseColor.size()>0){
            mat.setDiffuseColor( diffuseColor.get(0),
                diffuseColor.get(1), diffuseColor.get(2) );}
        if(specularColor.size()>0){
            mat.setSpecularColor( specularColor.get(0),
                specularColor.get(1), specularColor.get(2) );}
        if(shininess.size()>0){
```

```
        mat.setShininess( shininess.get(0));}
      shapeAppearance.setMaterial( mat );

      result = new Shape3D(geom, shapeAppearance);
      result.setUserData(shapeName);
      result.setPickable(true);
      result.setCapability(Shape3D.ENABLE_PICK_REPORTING);
      result.setCapability(Shape3D.ALLOW_PICKABLE_READ);

      shapePosition = new Transform3D();

      transformPos = new TransformGroup();

      int ii=0;
      for(int m=0;m<transformPosition.size()/3;m++){
           Transform3D tempPos = new Transform3D();
        float Xv = transformPosition.get(m*3);
        float Yv = transformPosition.get(m*3+1);
        float Zv = transformPosition.get(m*3+2);
        float Xr = rotationPosition.get(m*3+ii);
        float Yr = rotationPosition.get(m*3+1+ii);
        float Zr = rotationPosition.get(m*3+2+ii);
        float angle = rotationPosition.get(m*3+3+ii);
//>>>

      transformPos.getTransform(shapePosition);
      tempPos.setTranslation(new Vector3f(Xv, Yv, Zv));
      shapePosition.mul(tempPos);
           tempPos.setIdentity();
        if(Xr==1.0){
           tempPos.rotX(angle);}
        if(Yr==1.0){
             tempPos.rotY(angle);}
        if(Zr==1.0){
             tempPos.rotZ(angle);}
        shapePosition.mul(tempPos);
          tempPos.setIdentity();
        transformPos.setTransform(shapePosition);
//<<<
   ii++;
  }

   transformPos.setTransform(shapePosition);
   transformPos.addChild(result);
   J3Dinterface.BGaddX3D.addChild(transformPos);
      coodonates.removeAll(coodonates);
      normals.removeAll(normals);
   indices.removeAll(indices);
   transformPosition.removeAll(transformPosition);
   rotationPosition.removeAll(rotationPosition);
   ambientIntensity.removeAll(ambientIntensity);
```

```
   diffuseColor.removeAll(diffuseColor);
   shininess.removeAll(shininess);
   specularColor.removeAll(specularColor);

   for(int aa = 0; aa < max;aa++ ){
      coordonateShapeLoad[aa]=0;
     coordonateShapeLoadNormals[aa]=0;
     coordonateShapeLoadIndices[aa]=0;
   }

}

      J3Dinterface.objTransform.addChild(J3Dinterface.BGaddX3D);
}
 catch (Exception ex) {
      ex.printStackTrace();
      }
 }
}
```

ANEXA B

Listă de materiale folosite pentru construirea echipamentului:

- 6 buc. profil aluminiu 20x20x2, 1m;
- 1 buc. profil aluminiu 10x22.5x2, 1m
- 1 buc. profil aluminiu 10x16.5x2, 1m;
- 1 buc. platbanda aluminiu 20x2, 1m;
- 1 buc. platbanda aluminiu 24x2.5, 1m;
- 3 buc. cornier aluminiu 35x20x2, 1m;
- 1 buc. cornier aluminiu 40x15x2, 1m;
- 60 buc. șuruburi M4x10;
- 36 buc. șuruburi M4x20;
- 10 buc. șuruburi M4x40;
- 40 buc. șuruburi M3x15;
- 3 motoare pas cu pas bipolare;
- 3 drive stepper;
- 3 transmisii cu curea sincronă;
- 6 ghidaje liniare;
- 6 microswitch;
- 6 switch-uri pentru limitare de cursă;
- 3 microrelee;

Bibliografie

[1] Advanced Interfaces Group. Maverik. `http://aig.cs.man.ac.uk`.

[2] Allen Bierbaum and Aron Bierbaum. Vr juggler. `http://www.vrjuggler.org`.

[3] Anton, Jezernik and Gorazd, Hren. A solution to integrate computer-aided design (cad) and virtual reality (vr) databases in design and manufacturing processes. *Int J Adv Manuf Technol, London*, 22:768–774, 2003.

[4] Artifice, Inc. Designworkshop lite. `http://www.artifice.com/free/dw_lite.html`.

[5] Atănăsoae, Marcela și Ciocârlie, Mihai. Grafică 3d pentru web. `http://www.cogaionon.home.ro/referat/3d.htm`, 2004.

[6] Basic Micro, Inc. Basicatom pro 28-m. `http://www.basicmicro.com`.

[7] Bing-Yu, Chen and Yutaka, Ono. V3d model deformation along a parametric surface. *Proceedings of IASTED 2002 International Conference on Visualization, Imaging and Image Processing, Malaga, Spain*, pages 282–287, 9-12 September 2002 2002.

[8] Botez, Mihail. Solid edge. Întrebări și răspunsuri. `http://www.cadreport.ro/cadrep97.01/084.htm`.

[9] Boulanger, Pierre. Virtual reality and cad - complementary or competing? *CSA Newsletter, Winter*, XII(3):768–774, 2000.

[10] CADopia Inc. Intellicad. `http://www.cadvance.com`.

[11] Carey, R., Bell, G. and Marrin, C. Iso/iec 14772-1:1997, virtual reality modeling leanguage (vrml97). `http://www.web3d.org/x3d/specifications`.

[12] Cioi, Daniel. Referat nr. 3, realitatea virtuală în proiectarea asistată de calculator. Technical report, UPT, 2005.

[13] Cioi, Daniel. Virtual reality for computer-aided design. *The International Journal Robotica & Management*, 10(2):46–48, December 2005.

[14] Cioi, Daniel. Virtual reality modelling for educational robot systems. *Proceedings of RAAD'05, 14th International Workshop on Robotics in Alpe-Adria-Danube Region, Bucharest*, pages 225–228, May 26-28 2005.

[15] Cioi, Daniel. Collision detection in virtual reality. *Proceedings of ROBOTICA 2006, The 3rd International Conference on Robotics, Iasi*, pages 151–154; September 07-09 2006.

[16] Cioi, Daniel. Comparison of 3d stereoscopic visualization devices. *Proceedings of COMEFIM'8, The 8th International Conference on Mechatronics and Precision Engineering, Cluj-Napoca*, pages 669–674, June 8-10 2006.

[17] Cioi, Daniel. Haptic devices for computer aided design. *Proceedings of COMEFIM'8, The 8th International Conference on Mechatronics and Precision Engineering, Cluj-Napoca*, pages 883–888, June 8-10 2006.

[18] Cioi, Daniel. Interchanging three-dimensional models between cad and vr system. *Proceedings of PRASIC'06, Simpozionul national cu participare internationala Proiectarea Asistata de Calculator - Design de Produs, Brasov*, III:61–64, November 09-10 2006.

[19] Cioi, Daniel. Robot simulation using java 3d. *Proceedings of OPTIROB 2006, 1st International Scientific Meeting "Optimization of the Robots and Manipulators, Predeal*, pages 57–59, May 26-28 2006.

[20] Cioi, Daniel. A java3d application for visualization and deformation of cad models in virtual reality. *ISI Proceedings of The 18th INTERNATIONAL DAAAM SYMPO-*

SIUM "Intelligent Manufacturing & Automation: Focus on Creativity, Responsibility and Ethics of Engineers", Croatia, Zadar, pages 153–154, October 24-27 2007.

[21] Computer Hope. Virtual reality. `http://www.computer.com/jargon/v/vr.html`.

[22] Corseuil, Eduardo T. L. Environ - visualization of cad models in a virtual reality environment. *Eurographics Symposium on Virtual Environments*, 2004.

[23] Cruz-Neira, C. Virtual reality: The design and implementation of the cave. *Proceedings of SIGGRAPH 93 Computer Graphics Conference*, 1993.

[24] Crystal Space Team. Crystal space. `http://www.crystalspace3d.org`.

[25] Dani TH, Chu CP, Gadh R. Covirds: shape modeling in a virtual reality environment.

[26] Dani TH, Gadh R. A framework for designing component shapes in a virtual reality environment.

[27] David Scherer. The visual module of vpython. `http://www.vpython.org/webdoc/visual/index.html`.

[28] Digital Creations. Digital creations. `http://www.digicool.com`.

[29] DIMITRIU, Adrian. *Bazele sistemelor mecatronice.* Editura Universitatii Transilvania din Brasov, 2006.

[30] Dirk Reiners. Opensg. `http://opensg.vrsource.org`.

[31] Don Burns. Openscenegraph. `http://www.openscenegraph.org/projects/osg`.

[32] Eclipse Entertainment. Genesis 3d engine. `http://www.genesis3d.com`.

[33] EPSICOM. Productie si inginerie tehnica. `http://www.epsicom.com`.

[34] F I T, Inc. Cadvance. `http://www.cadvance.com`.

[35] Fiorentino, M., De Amicis, R., Monno, G., Stork, A. Spacedesign: A mixed reality workspace for aesthetic industrial design. *Proceding of ISMAR 2002 IEEE*, 2002.

[36] F.Kuester, M.A.Duchaineau, B.Hamann. Designersworkbench: Towards real-time immersive modeling. *Proc. of SPIE*, VII, January 27-27 2000.

[37] Free Software Foundation, Inc. The gnu project. `http://www.gnu.org`.

[38] Gary Bishop and H. Fuchs. Research directions in virtual environments. *ACM Computer Graphics*, 3(26):153–177, 1992.

[39] GERIGAN, C. and OGRUTAN, P. *Tehnici de interfațare.* Editura Universității Transilvania din Brasov, ISBN 973-9474-94-2, 2000.

[40] Gigante, M. Distributed, multi-person, physically-based interaction in virtual worlds. *Communicating With Virtual Worlds, Proceedings of Computer Graphics International, Tokyo, JP*, pages 41–49, Jun. 21-25 1993.

[41] Gomes, Antonio and Zachmann, Gabriel. Virtual reality as a tool for verification of assembly and maintenance processes. *Computer & Graphics*, 23:389–403, 1999.

[42] Grigore C. Burdea. Haptic feedback for virtual reality. *special issue on Virtual Prototyping, International Journal of Design and Innovation Research*, 2:17–29, 2000.

[43] Haiduc, Daniel. *Grafică și realitate virtuală.* Timișoara, 2002.

[44] Hannes Kaufmann and Dieter Schmalstieg and et al. Construct3d: A virtual reality application for mathematics and geometry education. *ACM Multimedia Conference*, pages 263– 276, 2000.

[45] Hasso-Plattner-Institute. Virtual rendering system. `http://www.vrs3d.org`.

[46] Holger, Regenbrecht. *A tangible AR desktop envoronment.* Pergamon, 2001.

[47] IMSI/Design, LLC. Turbocad. `http://www.imsidesign.com`.

[48] Ioan, Daniel. Cad pe internet studiu pentru comisia curiculara a upb. `http://www.lmn.pub.ro/~daniel/research/cad.htm`.

[49] Ionescu, Felicia. *Grafica în realitatea virtuală.* București, Editura Tehnică, 2000.

[50] Jack, Breen and Robert, Nottrot. Shading, texturing and anti-aliasing. *Automation in Construction*, 12(6), 2003.

[51] Jargon Dictionary. Virtual reality. `http://dict.die.net/virtual\%20reality`, 1995.

[52] Jayaram, S., Wang, Y., Jayaram, U., Lyons, K., Hart, P. A virtual assembly design environment. *Proceedings of the IEEE VR'99*, 1999.

[53] John Apperson. Cadsdt lite. `http://www.cadstd.com`.

[54] Juan Gabriel Del Cid Portillo. Parallel printer port access through java. `http://www.geocities.com/Juanga69/parport`.

[55] Kameyama, Ken-ichi. Institution Toshiba R&D Cent, Kawasaki, Jpn. Virtual clay modeling system. *Proceedings of the 1997 ACM Symposium on Virtual Reality Software and Technology, VRST. Lausanne*, pages 197–200, 1997.

[56] Kitware, Inc. Visualization toolkit (vtk). `http://public.kitware.com/VTK`.

[57] Liang J, Green M. Jdcad: a highly interactive 3d modeling system.

[58] Liua, X. and Doddsb, G. Virtual designworks - designing 3d cad models via haptic interaction. *Computer-Aided Design*, 36(12):1129–1140, 2004.

[59] Lynxmotion, Inc. Atom bot board. `http://www.lynxmotion.com`.

[60] Manetta, C. and Blade, R. Glossary of virtual reality terminology. *International Journal of Virtual Reality*, Vol. 1, No.2., 1995.

[61] Marin, Vlada. Realitatea virtuală (virtual reality), tehnologie modernă a informaticii aplicate. *Conferința Națională de Învățământ Virtual, Ediția a II-a, Bucuresti*, 29-31 oct. 2004.

[62] Mark, Foskey and Miguel, A. Otaduy and Ming, C. Lin. Artnova: Touch-enable 3d model design. *Proceedings. IEEE Virtual Reality, Orlando, FL, USA*, pages 119–126, 2002.

[63] Midnight Software, Inc. Deltacad. `http://www.deltacad.com`.

[64] Open CASCADE. Open cascade. `http://www.opencascade.com`.

[65] Parallax, Inc. Basic stamp 2e. `http://www.parallax.com`.

[66] Patti Koenig. Fastscript3d. `http://fastscript3d.jpl.nasa.gov`.

[67] Python Software Foundation. Psf. `http://www.python.org/psf`.

[68] R. Neugebauer, D. Weidlich, H. Zickner and T. Polzin. Virtual reality aided design of parts and assemblies. *Journal International Journal on Interactive Design and Manufacturing*, pages 15–20, April 2007.

[69] Regis Le Boite. Minos. `http://www.le-boite.com/minos.htm`.

[70] Sabin, Corneliu, Buraga. *Tehnologii web, vol.1 și 2.* Editura Matrix Rom, București, 2001.

[71] Sachs E, Roberts A, Stoops D. 3-draw: a tool for designing 3d shapes.

[72] Savii, G. George și Luchin, Milenco. *Modelare și simulare.* Editura Eurostampa, Timișoara, 2000.

[73] Savii, G. Geroge. Computer-aided design. *H. Bidgoli (ed) Encyclopedia of Information Systems, Academic Press*, pages 171–186, 2002.

[74] Schmalstieg, D., Fuhrmann, A., Hesina, G., Szalavári, Z. S., Encarnacao, L. M., Gervautz, M., Purgathofer, W.,. The studierstube augmented reality project.

[75] SGI. Opengl. `http://www.opengl.org`.

[76] SGI. Opengl performer. `http://www.sgi.com`.

[77] Shuming Gao, Huagen Wan, Qunsheng Peng. An approach to solid modeling in a semi-immersive virtual environment.

[78] Siemens Product Lifecycle Management Software Inc. Solid edge origin. `http://www.ugs.com`.

[79] SoftCAD International. Softcad 3d lite. `http://www.softcad.com`.

[80] Stephen F. White. White_dune. `http://vrml.cip.ica.uni-stuttgart.de/dune`.

[81] Steve J. Baker. Plib. `http://plib.sourceforge.net`.

[82] Stork, A.; Rix, J. Arcade. advanced realism cad environment. *Produktdaten-Journal*, 2(2):45–46, 1995.

[83] Sun Microsystems, Inc. Java 3d api tutorial. `http://java.sun.com/developer/onlineTraining/java3d`.

[84] Systronix, Inc. Jstamp. `http://jstamp.systronix.com`.

[85] Talabă, Doru. *Bazele CAD Proiectare Asistată de Calculator.* Editura Universităţii Transilvania, 2000.

[86] The DICT Development Group. `http://www.dict.org`.

[87] Tlinea. Tlinea. `http://tlinea.iespana.es`.

[88] Tungsten Graphics. Mesa 3d graphics library. `http://www.mesa3d.org`.

[89] Varkon. Varkon. `http://tlinea.iespana.es`.

[90] Vince, John. *Realitatea virtuală. Trecut, prezent și viitor.* București, Editura Tehnică, 2000.

[91] Volkov, S. and Vance, J.M. Effectiveness of haptic sensation for the evaluation of virtual prototypes. *Journal of Computing and Information Science in Engineering*, 1(2):123–128, June 2001.

[92] Von Schweber, L. and Von Schweber, E. Virtual reality: Virtually here. *PC Magazine*, pages 168–170, March 14 1995.

[93] Web3D Consortium. Open standards for real-time 3d communication. `http://www.web3d.org`, 2007.

[94] Weidlich, D. Cser, L. Polzin, T. Cristiano, D. Zickner, H. Virtual reality approaches for immersive design. *ANNALS- CIRP*, 56(1):139–142, 2007.

[95] Whyte, J. and Bouchlaghem, N. From cad to virtual reality: modelling approaches, data exchange and interactive 3d building design tools. *Automation in Construction*, 10:43–55, 2000.

[96] Wikipedia, the free encyclopedia. Mediu de dezvoltare. `http://ro.wikipedia.org/wiki/Mediu_de_dezvoltare`.

[97] Wikipedia, the free encyclopedia. Open source. `http://ro.wikipedia.org/wiki/Open-source`.

[98] WordPress. Diverse. `http://diverse.sourceforge.net/diverse`.

[99] Zachmann, Gabriel. Virtual reality in assembly simulation - collision detection, simulation algorithms, and interaction techniques. *Dissertation - Dem Fachbereich Informatik der Technischen Universität Darmstadt eingereichte*, 10. Juli 2000.

Listă de figuri

Lista publicațiilor personale

- Cioi Daniel, George Savii, "*Limbaje de realitate virtuală*", Editura Politehnica, în curs de apariție.

- Daniel Cioi, "A Java3d Application for Visualization and Deformation of CAD Models in Virtual Reality", ISI Proceedings of The 18th INTERNATIONAL DAAAM SYMPOSIUM "Intelligent Manufacturing & Automation: Focus on Creativity, Responsibility and Ethics of Engineers" (Croatia, Zadar, 24-27th October 2007): pp 153-154.

- Steliana Vatau, Daniel Cioi, Radulescu Corneliu, "Mechanical design of a hip joint for an anthropomorphic leg", ISI Proceedings of The 18th INTERNATIONAL DAAAM SYMPOSIUM "Intelligent Manufacturing & Automation: Focus on Creativity, Responsibility and Ethics of Engineers" (Croatia, Zadar, 24-27th October 2007): pp 801-802.

- Daniel Cioi, "Interchanging Three-Dimensional Models Between CAD and VR System", Proceedings of PRASIC'06, Simpozionul national cu participare internationala Proiectarea Asistata de Calculator, Vol. III - Design de Produs (Brasov November 09-10, 2006): pp 61-64.

- Daniel Cioi, "Collision Detection in Virtual Reality", Proceedings of ROBOTICA 2006, The 3rd International Conference on Robotics (Iasi September 07-09, 2006): pp 151-154.

- Daniel Cioi, "Haptic Devices for Computer Aided Design", Proceedings of COMEFIM'8, The 8th International Conference on Mechatronics and Precision Engineering (Cluj-Napoca June 8-10, 2006): pp 883-888.

- Daniel Cioi, "Comparison of 3D Stereoscopic Visualization Devices", Proceedings of COMEFIM'8, The 8th International Conference on Mechatronics and Precision Engineering (Cluj-Napoca June 8-10, 2006): pp 669-674.

- Daniel Cioi, "Robot Simulation using Java 3D", Proceedings of OPTIROB 2006, 1st International Scientific Meeting "Optimization of the Robots and Manipulators" (Predeal May 26-28, 2006): pp 57-59.

- Daniel Cioi, Steliana Vatau, Inocentiu Maniu, "Virtual Reality Laboratory for Robot Systems", Proceedings of SACI 2006, 3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence (Timisoara May 26-26, 2006): pp 634-641.

- Daniel Cioi, "Virtual Reality for Computer-Aided Design", The International Journal Robotica & Management, Vol.10, No.2 (December 2005): pp 46-48.

- Daniel Cioi, "Virtual Reality Modelling for Educational Robot Systems", Proceedings of RAAD'05, 14th International Workshop on Robotics in Alpe-Adria-Danube Region (Bucharest May 26-28, 2005): pp 225-228.

Listă proiecte de cercetare

Participare în 11 granturi (10 granturi ca membru și 1 grant ca director).

- Grant 694 CEEX, Mediu colaborativ de realitate virtuală pentru planificarea pre-operatorie în ortopedie (MERVI). Membru al echipei 2006-2008.

- Grant 1612 CEEX nr.88, Dezvoltarea și implementarea unor sisteme performante de investigare și recuperare a deformațiilor de coloană vertebrală la populația de vârstă școlară și categorii profesionale cu activități sedentare. Membru al echipei 2006-2008.

- GRANT 2799 CEEX, Optimizarea valorificării potențialului energetic al deșeurilor pentru obținerea de energie curată în instalații industriale românești - OVAPED. Membru al echipei 2006-2008.

- Grant 4293 CEEX, Sisteme pneumatice avansate de acționare precisă în robotică și în alte aplicații industriale bazate pe dezvoltarea de noi tipuri de servodistribuitoare proporționale în concepție mecatronică (SPASERVODIST). Membru al echipei 2006-2008.

- Grant 112 CEEX II 03, Platformă de simulare, control și testare cu aplicații în mecatronică - CONMEC. Membru al echipei, 2006-2008.

- Grant CNCSIS 83, Contribuții la utilizarea realității virtuale în proiectarea asistată de calculator. Director 2006-2007.

- Grant CNCSIS 347, Metode de proiectare asistată - intelligent cad - folosite la personalizarea dispozitivelor de corectare a deficiențelor de schelet. Membru al echipei 2006-2007.

- Grant 320 CEEX, Nanomateriale cu porozitate controlată și proprietăți magnetice și optice dirijate, obținute prin metoda sol-gel și sono-sinteză, cu potențial aplicativ în protecția mediului, biologie și medicină. Membru al echipei 2005-2007.

- Grant 21 CEEX I 03 / 07.10.2005. Cercetări privind posibilități de utilizare ale sistemelor robotice în scopul creșterii competitivității tehnico-economice a industriei românești. Membru al echipei, 2005-2007.

- The 6th FP of the European Commission - Skill-based Inspection and Assembly for Reconfigurable Automation Systems, (SIARAS) STREP NMP2-CT-2005-017146 (Control și asamblare bazate pe capabilități pentru sisteme automate reconfigurabile). Membru al echipei, 2005-2008.

- Grant CNCSIS 2005 TEMA 10 COD 162, Cercetări privind dezvoltarea unui laborator de mecatronică deschis și la distanță. Membru al echipei, 2004-2005.