UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL AUTOMATICĂ

Mediu de învățare 3D

Lucrare de disertație

Absolvent
Mureșan Cătălin-Gabriel

Iulie, 2014

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN
Prof.Dr.Ing. Liviu MICLEA

DIRECTOR DEPARTAMENT
Prof.Dr.Ing. Rodica POTOLEA

Mediu de învățare 3D

Lucrare de disertație

1. Absolvent: Mureșan Cătălin-Gabriel
2. Coordonator științific (1): Conf.dr.ing. Mihai DAMIAN
3. Coordonator științific (2): Prof.Dr.Ing. Liviu MICLEA
4. Conținutul lucrării: Pagina de prezentare, Capitolul I - Introducere, Capitolul2 - Obiectivele cercetării , Capitolul 3 - Stadiul actual în domeniu, Capitolul 4 - Planul aplicației, Capitolul 5 - Designul aplicației server, Capitolul 6 - Designul aplicației client, Capitolul 7 - Instrucțiuni, Capitolul 8 - Concluzii, bibliografie, anexe, CD - codul sursă al aplicației.
5. Locul documentării: UTCN, Cluj-Napoca
6. Consultanți: .....
7. Data emiterii temei: .....
8. Data predării: .....

Semnătură coordonator
Conf.dr.ing. Mihai DAMIAN
Prof.Dr.Ing. Liviu MICLEA

Semnătură absolvent
Mureșan Cătălin-Gabriel

Iulie, 2014

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Declarție pe proprie răspundere privind autenticitatea lucrării de disertație

Subsemnatul *Mureșan Cătălin-Gabriel*, legitimat cu *CI* seria *KT* numărul *771623*, CNP *1820513013911*, autorul lucrării *Mediu de învățare 3D* elaborată în vederea susținerii examenului de finalizare a studiilor de masterat la Facultatea de Automatică și Calculatoare, Departamentul Automatica, Specializarea *Informatică Aplicată* din cadrul Universității Tehnice din Cluj-Napoca, sesiunea *Iulie* a anului univeristar *2013/2014*, declar pe proprie răspundere, că această lucrare este rezultatul propriei mele activități intelectuale, pe baza cercetărilor mele și pe baza informatiilor obținute din surse care au fost citate în textul lucrării și în bibliografie.

Declar că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență sau disertație.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de disertație*.

Cluj-Napoca
data

PRENUME NUME
Semnătură absolvent

Rezumat

Progresul rapid în domeniul I.T.& C și scăderea prețurilor componentelor hardware performante a făcut fezabilă aplicarea metodei de învațare in medii virtuale 3D în oricare treaptă a sistemului educațional. Într-un mediu virtual tridimensional, obiectele studiate sunt reprezentate prin coordonate ce descriu forma și pozitionarea lor în spațiu, apropiind astfel modul de reprezentare de cel al obiectelor din lumea reală. Utilizatorii se pot poziționa în orice punct al spațiului virtual, fapt care le va permite să studieze obiectele din orice unghi, lucru dificil de realizat cu programele informatice cu redare 2D. De asemenea o parte dintre obiectele virtuale pot fi programate să răspundă la acțiunea utilizatorului, conducând la sporirea gradului de implicare a utilizatorului, cu rezultate benefice în procesul de învățare.

Foarte importantă este siguranța și costul redus. Utilizatorii au șansa de a efectua activități care ar fi altfel foarte costisitore pentru instituțiile implicate în procesul de educare, sau în medii cu grad mare de risc. În mediile de învățare 3D se pot repeta în siguranță proceduri care nu sunt tolerante la erori, precum operațiile chirurgicale sau controlul proceselor într-o centrala nucleară.

În această lucrare sunt sunt identificate elementele necesare pentru implementarea unui program informatic reprezentând un asemenea mediu de învațare 3D, care să ruleze în sistemul de operare Linux, și se va face descriere a arhitecturii aplicației. Această lucrare va conține și codul sursă cu comentarii.

Cuprins

Listă de tabele **iv**

Listă de figuri **v**

Capitolul 1 Introducere **1**
1.1 Introducere 1
1.2 Problema adresată 1
1.3 Motivație 1
1.4 Descrierea studiului pe capitole 2
1.5 Învațarea și mediile 3D. Concepte relevante. 2

Capitolul 2 Obiectivele cercetării **4**
2.1 Introducere 4
2.2 Obiective 4
2.3 Argumente 5
2.4 Mini studiu empiric privind raportul aplicațiilor web - aplicații clasice cu destinație educativă 6

Capitolul 3 Stadiul actual în domeniu **8**
3.1 Introducere 8
3.2 Abordări similare 8
3.3 Second Life 8
3.4 Uther Academy 9
3.5 iSocial 10
3.6 Tehnici/Tehnologii folosite 11

Capitolul 4 Planul aplicației **12**
4.1 Introducere 12
4.2 Funcțiile sistemului 12
4.2.1 Minimul de funcționalitate de cercetat și implementat 12
4.2.2 Funcționalitate extinsă 13
4.3 Decizie privind design-ul sistemului 13

4.4 Test 14
4.5 Argumentarea deciziei pe baza rezultatelor testării 15

Capitolul 5 Design-ul aplicației server **16**
5.1 Introducere 16
5.2 Funcțiile aplicației server 16
5.2.1 Funcții de bază ale serverului 16
5.2.2 Aspecte tehnice și principii generale de design 17
5.3 Mecanism pentru obținera scalabilității aplicației server 18
5.3.1 Interfața PLUG-IN-ului 18
5.3.2 Tipuri de plug-in-uri dezvoltate pentru serverul mediului de învățare 3D 19
5.3.3 Descrierea mecanismului de încărcare a plug-in-urilor 20
5.4 Modul de reprezentare a datelor 22
5.4.1 Scurtă introducere în XML 22
5.4.2 Structura XML a datelor transmise dinspre server spre aplicația client 23
5.5 Schema serverului. Diagrama claselor 24
5.5.1 Modelarea comenzilor primite de server 24
5.5.2 Modulul de interpretare a mesajelor XML 25
5.5.3 Server TCP/IP multi-client 25
5.6 Testarea mecanismului de încărcare a plug-in-urilor 26

Capitolul 6 Design-ul aplicației client **27**
6.1 Introducere 27
6.2 Funcțiile aplicației client. 28
6.3 Conceptul aplicației și tehnologiile utilizate. 28
6.3.1 Modul de livrare a conținutului educativ în mediul 3D 29
6.3.2 Tehnologii utilizate pentru redarea 3D a mediului de învățare. . . . 30
6.3.3 Tehnologii utilizate pentru relizarea interfației grafice (GUI) 35
6.3.4 Tehnologia utilizată pentru comunicarea cu serverul 36
6.4 Decizie privind design-ul aplicației client. 38
6.5 Comnezi către server 43
6.6 Diagrama claselor 45
6.7 Test 45

Capitolul 7 Instrucțiuni **48**
7.1 Instalarea bibliotecilor software 48
7.2 Compilarea aplicației 49
7.3 Setarea unei surse locale ca depozit pentru plug-in-urile aplicației client . . 50

Capitolul 8 Concluzii **51**
8.1 Instalarea bibliotecilor software 51
8.2 Compilarea aplicației 51
8.2.1 Dimensiune 51

Anexa A Testarea fiabilității modelului CLIENT-SERVER **52**
A.1 Script 52
A.2 Server de test 52
A.2.1 Rezultate 55

Anexa B Încărcarea dinamică a claselor în JAVA **56**

Anexa C Iconuri grafice utilizate pentru nodurile VRML în lucrarea "The Inventor Mentor" **61**

Bibliografie **62**

Listă de tabele

2.1 Rezultate 6

3.1 Tehnologii utilizabile petru dezvoltarea de medii 3D sub Linux 11

Listă de figuri

2.1 Căutare - tehnologii web 6
2.2 Căutare - tehnologii 3D și c++ 6

3.1 Eveniment în Second Life 9
3.2 U.A. - aulă virtuală 10
3.3 iSocial - panou informativ 10

4.1 Modelul P2P 14
4.2 Modelul Client-Server 14

5.1 Arhitectura Plug-in 18
5.2 Schema Plug-in Hibrid 19
5.3 Structură arborescentă de date descrisă cu XML 23
5.4 UML - COMENZI 24
5.5 UML - PARSER GENERIC XML 25
5.6 Principiul serverului multiclient 25
5.7 UML - Clasele principale ale serverului multi-client 26

6.1 Dispunerea elementelor în interfața grafică 28
6.2 Mediul 3D și materialul educativ interactiv 29
6.3 Graf VRML (exemplu) 32
6.4 Axe 33
6.5 Rotație 33
6.6 ANIMAȚIE VRML 34
6.7 Mecanismul SIGNAL-SLOT 36
6.8 Mecanismul I/O asincron 37
6.9 Bucla schimbului de informații între client și server 44
6.10 Diagrama claselor aplicației client 45
6.11 Test - avatare - față în față 46
6.12 Test - avatare aliniate la intrarea într-o fortificație romană (3D) 47

B.1 Server - diagrama UML completă 60

C.1 Noduri VRML 61

Capitolul 1

Introducere

1.1 Introducere

Acest capitol definește în secțiunea 1.2 problema adresată, fiind subliniați principalii piloni ai problemei: costul serviciilor educaționale și în special complexitatea și costul ridicat al soluțiilor existente ce implică tehnologia mediilor de învățare 3D. Secțiunea 1.3 este un enunț al soluției ce se dorește a fi oferită în această lucrare. Secțiunea 1.4 subliniază etaplele studiului iar secțiunea 1.5 tratează în linii mari teorii referitoare la mediile de învățare 3D și la învățare în general, teorii ce vor avea influență asupra produsului final (applicația informatică).

1.2 Problema adresată

Asigurarea unui învățământ de calitate poate implica costuri semnificative pentru diversele instituții sau organizații care oferă asemenea servicii, în special când mediul optim de învățare presupune colaborarea, întreprinderea de experimente multiple sau lucrul în medii cu risc ridicat. De asemenea, în unele cazuri, instruirea poate avea loc în medii simulate al căror cost de construire și folosire ar fi mult prea mare.

1.3 Motivație

Ca răspuns la complexitatea și costul soluțiilor existente, se dorește realizarea unei aplicații reprezentând o un mediu virtual 3D; o aplicație de dimensiuni reduse dar care poate fi cu ușurință extinsă și completată cu noi elemente reprezentând: experimente de laborator virtual, diverse forme de reprezentare în forma grafica 3D a cunoștințelor, spații virtuale pentru desfașurarea de activități educative (muzee, săli de conferință virtuală) etc. O asemenea aplicație va avea se va dezvolta în două module: client și server, ambele dinamice, ușor de extins și care să se constituie o mini platformă pe care alți programatori

să poată construi cu minim efort lumi virtuale orientate spre livrarea conținutului educativ intr-o formă cât mai atragatoare pentru utilizatori.

Aplicația informatică va rula sub GNU/Linux, pentru inplementarea acesteia se vor folosi doar unelte și bibilioteci software dezvoltate de comunitățile free software / open source. Codul sursa va fi eliberat licență GNU.

1.4 Descrierea studiului pe capitole

În ***Capitolul 1*** (capitolul curent) se pune accentul pe descrierea nevoii instituțiilor furnizoare de servicii educațional de unelte moderne pentru îndeplinirea obiectivelor lor specifice și pe lipsa de alternative care să indeplinească **simultan** următoarele aspecte: libertatea codului (open source / free software), dimensiune redusă și modularitate ridicată, costuri minime. Se afirmă obiectivul creării unui program liber (free/open source) ușor de menținut și extins reprezentând o lume virtuală 3D. Definițiile și conceptele teoriei învățării in medii virtuale încheie capitolul 1.

În ***Capitolul 2*** sunt enumerate obiectivele cercetării.

În ***Capitolul 3*** sunt prezentate specificațiile generale ale lucrării de cercetare, atât obiectivele minimale care vor trebui atinse până la definitivarea studiului cât și obiectivele potențiale de atins în cazul extinderii lucrării.

În ***Capitolul 4*** se analizează sistemul în întregime prin descrierea ansamblului *client-server*. Se decide design-ul sistemului și se argumentează alegerea făcută prin testele anterior stabilite.

În ***Capitolul 5*** se realizează un studiu amănunțit al aplicației *server* din sistem. Se decide design-ul aplicației, se efectuează teste și se decid aspectele tehnice referitoare la implementarea aplicației: limbajul folosit pentru implementare, biblioteci software folosite, algoritmi și metode de implementare.

În ***Capitolul 6*** se realizează un studiu amănunțit al aplicației *client* din sistem. Se decide design-ul aplicației, se efectuează teste și se decid aspectele tehnice referitoare la implementarea aplicației: limbajul folosit pentru implementare, librării software folosite, algoritmi și metode de implementare.

În ***Capitolul 7*** sunt prezentate instrucțiunile necesare pentru compilarea aplicației.

În ***Capitolul 8*** sunt prezentate concluziile.

1.5 Învațarea și mediile 3D. Concepte relevante.

În mare masură, soluția tuturor acestor probleme se găsește în mediile de învățare virtuale cu redare 3D, datorită unor caracterisici care le califică ca și cadru (în unele cazuri ideal) de învățare. Unele dintre cele mai importante caracteristici ale mediilor de învățare cu redare tridimensionala sunt: capacitatea de a simula orice spațiu fizic, de a intermedia interacțiunea dintre diverse persoane aflate în zone geografice diferite și oferta de unelte

de observare și măsurare a performanțelor sau a progresului participanților la procesul de învățare și faptul ca orice resursă virtuală poate fi refolosită fară costuri suplimentare.

O analiză a autorilor Wann și Mon Williams oferă o descriere a mediilor tridimensionale ca medii ce "valorifică aspectele naturale ale percepției umane prin extinderea informațiilor vizuale în trei dimensiuni spațiale și care poate suplimenta aceasta informație cu alți stimuli și modificari temporale"[1] și care "permit interacțiunea utilizatorului cu obiectele redate"[1]. Se pot astfel deduce trei elemente care disting mediile de învățare 3D de alte medii de învățare virtuale. Mai detaliat, mediile virtuale tridimensionale sunt medii grafice ce crează impresia de spațiu 3D, în care utilizatorul controlează caractere generate de computer (avatare), caractere care îi reprezintă în timp ce interacționează cu mediul sau cu alți utilizatori. Acestea pot contribui la sistemul educațional prin facilitarea colaborării, comunicării și experimentării. Prin oferirea unui surogat al realității, mediile de învățare pot crea percepția de existență a utilizatorului în mediul simulat.

În psihologie și educație, învățarea este definită ca fiind procesul care aduce împreună experiența cognitivă, emoțională și influența de mediu, pentru acumularea, îmbunatațirea sau schimbarea cunoștințelor, abilităților sau a concepției despre lume a unui individ.[2]

Clasificarea metodelor de învățare, stabilită de Frederic Vester[3] : învățarea auditivă, învățarea vizuală, învățarea tactilă, învățarea cognitivă (prin intelect). Prin această metodă de clasificare F. Vester, neagă efortul intelectual pentru primele trei tipuri, acest efort fiind atribuit învățării cognitive. Deși nu poate fi în totalitate adevărat, fiecare dintre noi am putut experimenta reducerea 'consumului' intelectual atunci când am învățat folosindu-ne de materiale didactice cu vizuale sau auditive. Personele de toate vârstele învață cel mai bine atunci cînd sunt implicate în experiențe semnificative. Învățarea are loc atunci când mintea este capabilă să pună la un loc informațiile primite de la toate simțurile și să le coreleze cu experințele trecute. Prin folosirea mai multor simțuri pentru a învăța se poate da mai mult sens procesului de acumulare. Copii în mod natural învață folosindu-se de toate simțurile în cel mai eficient mod posibil.

Valoarea unui mediu 3D bine construit constă în faptul ca poate antrena simțul vizual al utilizatorului, cu îmbunătățirea rezultatelor.

Capitolul 2

Obiectivele cercetării

2.1 Introducere

În secțiunea 2.2 se definește obiectivul lucrării și se enumeră etapele necesare îndeplinirii obiectivului. Motivația alegerii acestui obiectiv este prezentată, în secțiunea 2.3, cu argumente de ordin estetic referitoare la calitatea prezentării materialelor educative în redare tridimensională cât și cu argumente practice referitoare la utilitatea aplicației software rezultate. Dorința autorului de a dezvolta o aplicație din categoria free software / open source este evidentă și este în sine un argument.

2.2 Obiective

Pornind de la observația empirică privind existența unui număr mult mai mare de aplicțiilor educative bazate pe tehnologii web comparativ cu numărul aplicațiilor native bazate pe tehnologie ce implică grafică 3D (în special aplicații non proprietare pentru Linux), și luând în calcul conceptul înrădacinat în rândul programatorilor privind gradul de dificultate redus pentru realizarea de aplicații web comparativ cu aplicațiile "standalone", **se dorește prin această lucrare a se demonstra faptul că se poate dezvolta o aplicație non-web care să faciliteze publicarea de materiale educative aproape la fel de usor ca în cazul folosirii tehnologiilor web dar de o calitate mai ridicată și într-o formă mult mai atractivă.**

În acest sens se vor urmări:

- Identificarea unei distribuții Linux care să înlesnească instalarea componentelor necesare pentru dezvoltarea aplicației.
- Identificarea librăriilor software necesare pentru comunicarea in rețele web, grafică 3D, interfețe grafice etc.

- Crearea unei platforme software care să preia în mare măsură complexitatea obișnuită în cazul aplicațiilor cu grafică 3D.

- Crearea unei librării software care să permită altor programatori să extindă funcționalitatea aplicației (sistemului client-server).

2.3 Argumente

Într-o mică măsură se tratează problema practică a reducerii decalajului de proliferare dintre mediile de invățare bazate pe tehnologii web prin realizarea unei aplicații software cât mai flexibile care să permită crearea și publicarea de conținut educativ interactiv intr-un mediu 3D adecvat, de catre persoane ce posedă cunostințe minime de programare. Astfel, se poate echilibra proporția acestor sisteme în totalul produselor informatice destinate învățării și se pot valorifica progresele recente din domeniul hardware.

Argumentele generale în favoarea metodei de învățare în medii 3D sunt : modul de reprezentare a obiectelor studiate și apropierea lor de obiectele din lumea reală prin formă; faptul că utilizatorii se pot poziționa în orice punct al spațiului virtual, fapt care îi permite utilizatorului să studieze obiectele din orice unghi, lucru greu de realizat cu materialele didactice tradiționale sau cu programele informatice cu redare 2D; faptul că obiectele virtuale pot fi programate să răspundă la acțiunea utilizatorului, fapt ce poate conduce la sporirea gradului de implicare a utilizatorului, cu rezultate benefice în procesul de învățare; siguranța și costul redus, utilizatorii având șansa de a efectua activități care ar fi altfel foarte costisitore pentru instituțiile implicate în procesul de educare, sau în medii cu grad mare de risc. În mediile de învățare 3D se pot repeta în siguranță proceduri care nu sunt tolerante la erori, precum operațiile chirurgicale sau controlul proceselor într-o centrala nucleară. De asemenea, un argument puternic în favoarea sistemelor informatice de învățare este eliminarea necesității prezenței studentului intr-o clasă sau în o anumita zonă geografică.

Argumentele speciale ale acestui studiu sunt de natură practică. Se în primul rând pune accentul pe flexibilitatea și extensibilitatea aplicației. Se urmărește în al doilea rând crearea unui concept de aplicație software care să permită persoanelor cu minime cunoștințe de programare să participe prin adăugarea de noi funcții și cu materiale educative. *Simplitatea sistemului* este se asemenea un argument în favoarea aplicației informatice practice. Eliminarea gradului sporit de tehnicitate ce 'acompaniază' în general mediile și sistemele de învățare 3D, ar putea capta interesul diverselor persoane implicate sau implicabile în realizarea de software educativ, a persoanelor implicate în sistemul de învățământ și nu în ultimul rând, al utilizatorilor finali.

	tehnologie	*paginireturnate*
web	32000000	0,48
$3D - C++$	4120000	0.44

Tabela 2.1: Rezultate

2.4 Mini studiu empiric privind raportul aplicațiilor web - aplicații clasice cu destinație educativă

O metodă rapidă pentru cuantificarea interesului public pentru orice domeniu este metoda "motorului de căutare". Astfel, se poate beneficia de efortul uriaș depus de anumite companii pentru colectarea și claificarea datelor. Rezultatele nu sunt la fel de precise ca și studiile direcționate pe fenomenul proliferării tehnologiilor diverse, dar sunt destul de credibile.

Pentru compararea proliferarii metodologiilor web și a celor clasice cu destinație educativă vom considera numărul de pagini returnate precum și viteza de returnare a datelor de către motorul de căutare. Primul indice este concludent. Al doilea indice poate oferi informații suplimentare, considerînd mecanismul de depozitare (cache-ing) folosit pentru stocarea datelor cu număr mai mare de accesări. Pentru cuvinte cheie de cautare:

- web based learning environments 2.1
- 3D learning environment c++ 2.2

rezultatele sunt :

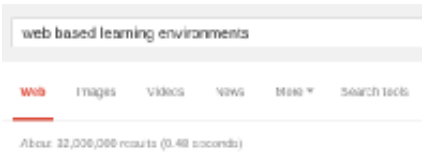

Figura 2.1: Căutare - tehnologii web

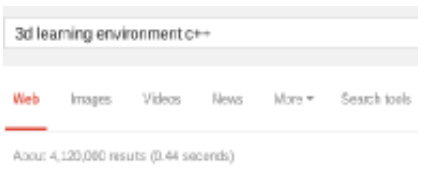

Figura 2.2: Căutare - tehnologii 3D și c++

Datele returnate de către motorul de căutare indică faptul că tehnologiile web sunt cu mult mai apreciate decât tehnologiile clasice cu grafică 3D, numărul de rezultate returnate fiind de 8 ori mai mare în favoarea tehnologiei web pentru timp de răspuns comparabil.

Capitolul 3

Stadiul actual în domeniu

3.1 Introducere

Dezvoltarea tehnologică accelerata din ultimele decenii a afectat și domeniul educației asistate de calculator. Câteva exemple de medii virtuale 3D accesibile publicului sunt: Second Life, Uther Academy și iSocial (http://isocial.missouri.edu/i Fiecare dintre aceste implementări ale aceleiași probleme abordează diferit modul livrare a materialelor educative. Deși codul sursă pentru majoritatea mediilor de învățare 3D este cod proprietar, se va încerca descrierea lor din perspectiva utilizatorului în următoarele subcapitole. Secon Life este open source și poate fi studiat. Secțiunea 3.2 se bazează pe similitudini în încercarea de a determina majoritatea opțiunilor pe care un utilizator le așteaptă de la mediile de învățare 3D.

3.2 Abordări similare

Toate exemplele notabile prezentate ulterior fac uz de avatare pentru a induce utilizatorului sentimentul de participare activă și imersiune în lumea virtuală. Utilizatorii sunt încurajați astfel să comunice între ei atât verbal cât și prin modificarea posturii și aparenței grafice a avatarului ce îi reprezintă în lumea virtuală.

Aceasta tehnică este luată în considerare în implementarea aplicației descrise în acest studiu.

3.3 Second Life

Percepția generală este că Second Life (SL) ar fi un joc pe Internet. Nu este însă un joc organizat, cu reguli impuse și unde să fie urmărit un anumit scop. Pe situl web oficial, Second Life este descris ca fiind „o lume virtuală imaginată și creată de rezidenții

ei"; într-adevăr, SL este o lume diversificată, în care poți întâlni oameni din toate colțurile lumii reale. Second Life este mai mult decât orice altceva, o rețea de socializare.

Figura 3.1: Eveniment în Second Life

Cele mai importante și interesante activități educative derulate în Second Life:

- Vizite asistate în muzee și teatre virtuale.
- Cursuri în săli de clasa virtuale.
- Jocuri de tip "orientare turistică" cu puncte intermediare și indicii cu subiect educațional.
- Proiecte cu colaborare în echipă.
- Cursuri online la diverse universități
- Panouri informative.
- Grupuri educaționale.

Majoritatea facilitaților oferite de SL ar trebui să se regasească în orice platformă de e-learning cu avatare.

3.4 Uther Academy

UtherAcademy este o aplicație de e-learning care facilitează participarea studenților din toata lumea la cursuri în medii imersive 3D. Liniile educative propuse sunt din categoria dezvoltarii profesionale. La data redactării acestei lucrări UtherAcademy avea deschise trei departamente :

- Academia de afaceri online.
- Cursuri pentru decoratori.
- Body arts.

Figura 3.2: U.A. - aulă virtuală

Modul de prezentare generala nu diferă foarte mult de Secon Life. Studenții participa online la cursuri în clase virtuale. Procesul de învățare este supravegheat de instructori.

3.5 iSocial

iSocial este un mediu de învățare 3D, dezvoltat pe baza toolkit-ului pentru construirea lumilor virtuale OpenWonderland, creat pentru predarea de competențe sociale tinerilor diagnosticați cu autism (ASD). În acest scop, iSocial facilitează intracțiunea socială și oferă suport pentru dezvoltarea de competențe sociale într-un mediu sigur și complet controlat.

Localizarea geografica poate restricționa accesul la tratamentele și exercițiile necesare recuperării sociale a tinerilor afectați de ASD. iSocial este una dintre soluțiile rezolvării pozitive a acestei probleme.

Figura 3.3: iSocial - panou informativ

Denumire	*Bibliotec/Ubuntu*	*Utilizare*
OpenGL	libGLU.so	API pentru grfica 2D și 3D
Zlib	libz.so	Comprimare date
OpenSSL	libssl.so	Protocoale de comunicare în rețea SSL și TLS
OGG	libogg.so	Format media audio-video
PNG	libpng12.so	Imagini PNG
GLib	libdbus-glib-1.so	Sistem de transmitere a mesajelor între procese
GTK	libgtk2.0-dev	GIMP toolkit
OpenAL	libopenal-dev;libalut-dev	OpenAL - Biblitecă pt. redarea sunetului (audio)
Vorbis	libvorbis-dev	Codec audio-video (API)
APACHE	libapr1-dev	Apache portabele runtime
JPEG	libopenjpeg.so;libjpeg.so	Codec JPEG
SDL	libsdl1.2-dev	Media Layer - faciliteaza accesul la periferice
Boost	libboost-dev	Alternativa pentru C++ STL (+ comunicare în rețea)
JsonCpp	libjsoncpp-dev	Interpretarea fișierelor Json (c++)

Tabela 3.1: Tehnologii utilizabile petru dezvoltarea de medii 3D sub Linux

3.6 Tehnici/Tehnologii folosite

Second Life (SL) este cea mai cunoscută și populară implementare a unei lumi virtuale. Deși nu este dezvoltată strict ca aplicație destinată instruirii in medii 3D, o parte însemnată a activităților desfașurate în Second Life sunt activități educative.

Un studiul tehnologiilor folosite pentru dezvoltarea mediului Second Life este suficient pentru identifica majoritatea bibliotecilor software folosibile pentru dezvoltarea sub Linux a unei aplicații similare. S-a alcătuit o listă a tehnologiilor folosite pentru dezvoltarea SL.3.1

Pentru dezvoltarea aplicației client SL se folosesc următoarele tehnologii: OpenGL pentru redarea graficii 3D, GTK pentru interfețele grafice, Boost și APACHE pentru schimbul de date în rețea între aplicația client și serverul lumii virtuale, Formatul de date JSON pentru structurarea datelor interschimbate în rețea între aplicația client și server, OpenAL pentru redarea sunetului și OGG/Vorbis ca și format/codec media și PNG/JPEG pentru redarea stocarea imaginilor. Codul sursă pentru serverul SL nu este open source.

O parte dintre aceste tehnologii sunt folosite pentru realizarea mediului de învățare 3D descris în această lucrare.

Capitolul 4

Planul aplicației

4.1 Introducere

În acest capitol, în secțiunea 4.2, se stabilesc funcțiile sistemului. În secțiunea 4.3 se identifică numitorul comun al funcțiilor sistemului ca fiind facilitarea funcției de comunicare și se enumeră metodele prin care comunicarea la distanță se poate implementa. După analiza punctelor slabe și a punctelor forte ale acestor metode se afirmă alegerea metodei CLIENT-SERVER ca fiind decizia de design de urmat și implementat. În secțiunea 4.4 se proiectează un test ca suport pentru argumentarea deciziei de design (din secțiunea 4.5).

4.2 Funcțiile sistemului

4.2.1 Minimul de funcționalitate de cercetat și implementat

Prin analiza funcțională, bazata pe observație liberă a câtorva dintre mediile 3D existente și accesibile pe internet, am putut determina o parte din funcțiile absolut necesare ale sistemului.

Aceste funcții sunt parte integrantă a cerințelor sistemului. La acestea se mai adaugă cerințele de ordin general precum: implementarea unei interfețe grafice ușor de folosit, mentenabilitatea și extensivitatea sistemului, etc..

Funcții ale sistemului:

- Utilizatorii sistemului sunt conștienți de prezența altor utilizatori (persoane reale) în lumea virtuală. Sistemul îndeplinește funcția de liant prin facilitarea schimbului de idei între utiliztori.

- Utilizatorii pot interacționa cu o parte dintre obiectele din umea virtuală. În unele situații, rezultatul acestei interacțiuni conduce la modificări permanente în reprezentarea lumii virtuale simulate. Sistemul îndeplinește funcția de stocare a modelului tridimensional al lumii viruale, inclusiv a starii acestui model, stare ce poate fi modificată de către utilizatori.

- Sistemul va stoca informațiile referitoare la identificarea utilizatorilor. Această identificare nu presupune stabilirea identității persoanei reale ci are funcția de a servi ca identificator pentru un catalog al progresului utilizatorului în procesul de învățare.

- Utilizatorii vor putea contribui la extinderea lumii virtuale prin publicarea de materiale educative dezvoltate de către aceștia. Sistemul va permite contribuția utilizatorilor cu noi materiale 3D și va avea funcția de stocare și prezentare a noilor materiale educative.

O parte dintre aceste funcții enunțate se vor implementa în aplicația software demonstrativă atașată acestui proiect.

4.2.2 Funcționalitate extinsă

Următoarele funcții nu vor fi incluse în software-ul demonstrtiv, dar sunt recunoscute ca necesare pentru orice sistem de învățare.

- Funcția de asigurare a accesibilității pentru persoanele cu deficiențe de vedere sau deficiențe motorii.

- Funcția de redare a tuturor formelor de conținut media. Din motive tehnice și pentru complexității aplicației demonstrative, în mare măsură se vor exclude conținuturile media video și audio.

4.3 Decizie privind design-ul sistemului

Toate funcțiile primare ale sistemului fac referire la facilitarea comunicării între utiliztori pentru asigurarea schimbului liber de informații, sau între sistem și utilizatori cu scopul stocării și/sau accesării de date cu pentru diverse scopuri legate de funcționarea sistemului sau menținerea unui catalog al progresului utilizatorilor. Aceste schimburi de date se fac la distanță, pentru a nu condiționa geografic participarea la procesul de învățare.

Pentru implementarea comunicării în rețea se va alege unul dintre cele două modele general utilizate în practică: modelul client-server 4.2 și modelul p2p (point to point) 4.1.

Modelul P2P este un model descentralizat în care fiecare nod este atât client cât și server pentru celelalte noduri din rețea. Modelul este mai dificil de implementat, consistența datelor în rețea este mai greu de menținut și administrarea este destul de dificilă. Deși modelul este mai robust prin faptul că piederea unui nod din rețea nu

conduce la sistarea serviciului deoarece funcția de server este preluată de restul nodurilor; acest model nu este cel mai bun pentru implementarea unei lumi virtuale, unde consistența datelor este foarte importantă. Aceasta consistență a datelor permite tuturor utilizatorilor să perceapă aceeași lume virtuală.

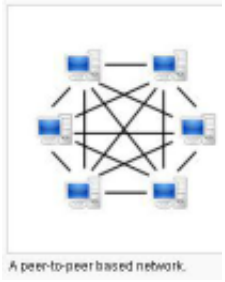

Figura 4.1: Modelul P2P

Modelul Client-Server este un model centralizat în care fiecare nod client comunică și depinde de serviciile nodului server. Acest model este relativ ușor de implementat, ușor de administrat, consistența datelor poate fi verificată și inpusă la nivelul serverului iar stocarea datelor poate fi făcută centralizat, la același nivel. Cel mai mare dezavantaj al acestui model este faptul că pierderea accesului la server prin oprirea acestuia face tot sistemul inoperabil. Acest model este mult mai portivit pentru implementarea unei lumi virtuale și a unui mediu de învățare 3D.

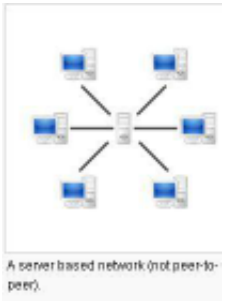

Figura 4.2: Modelul Client-Server

4.4 Test

Testul constă în simularea comunicării dinspre server către client. Un server va primi mesaje de la o aplicație client și va incerca să expedieze mesaje către un număr de 10 alte aplicații client, care salveaza în câte un fișier mesajele primite de la server, cu o etichetă pentru momentul primirii mesajului. Mesajele vor avea o lungime cuprinsă între 1

și 1024 caractere iar timpul din etichetă va fi exprimat în secundele trecute de la 1 ianuarie 1970 (Unix epoch).

Se consideră testul reușit dacă toate mesajele ajung la toate aplicațiile client în aceeași ordine și într-un timp copmarabil, sub 1 secundă distanță.

Pentru **rularea testului** se codifică aplicația server demonstrativă și se utilizează utilitarul **telnet** ca program client. Un "shell script" va fi programat pentru **rularea automată** a testului. **Evaluarea** se va realiza cu ajutorul comenzii **diff**.

Codul sursă al aplicației server, scriptul pentru rularea automată a testului și rezultatele returnte de utilitarul diff sun atașate în Anexa A.

4.5 Argumentarea deciziei pe baza rezultatelor testării

Testul a fost executat cu succes. Datele obținute se află în ANEXA A.

Deoarece comunicarea între aplicațiile client și server sunt livrate consistent și la timp, se concluzionează faptul că se poate folosi schema client-server pentu implementarea aplicației.

Capitolul 5

Design-ul aplicației server

5.1 Introducere

În acest capitol, în secțiunea 5.2, se stabilesc funcțiile serverului și se prezintă tehnologiile folosite la producerea aplicației server. În secțiunea 5.3 se explică mecanismul adoptat pentru atingerea unui scop important, "scalabilitatea aplicației" . În secțiunea 5.4 se stabilește modul și formatul de reprezentare a datelor transmise între aplicația client și aplicația server. Secțiunea 5.5 este dedicată prezentării părții celei mai relevante din diagrama claselor, diagrama completă fiind atașată în Anexa B. În secțiunea 5.6 sunt stabiliți parametrii testului privind scalabilitatea aplicației. Codificarea testului și rezultatele fiind expuse în Anexa B. Testul modului de comunicare și testul eficienței comunicării între client și server s-a efectuat pentru capitolul anterior, rezutatele fiind publicate în Anexa A.

5.2 Funcțiile aplicației server

5.2.1 Funcții de bază ale serverului

- Asigurarea comunicării înspre una sau mai mule aplicații client.
- Asigurarea comunicării între clienți prin intermediul serverului.
- Stocarea datelor referitoare la activitățile întreprinse de către utilizatori într-o bază de date.
- Eliberarea datelor din baza de date, la cererea utilizatorului.
- Stocarea și publicarea informațiilor ce descriu geometria lumii virtuale.
- Stocarea și publicarea acțiunilor utilizatorilor ce au efect asupra limii virtuale.

Serverul are scop demonstrativ. Funcția de securitate a datelor la transfer și la stocare este ignorată. De asemenea, implementarea funcțiilor se realizează în cel mai simplu mod posibil, pentru reducerea complexității și dimensiunii aplicației.

5.2.2 Aspecte tehnice și principii generale de design

Limbaje de programare

Codificarea aplicției server se va executa în limbajele de programare Java și JavaScript.

Java este probabil cel mai potrivit limbaj pentru implementarea aplicațiilor server datorită numărului mare de biblioteci software utilizate pentru realizarea schimbului de date în rețelele de calculatoare. Acestea pun la dispoziția programatorului o multitudine de funcții de nivel înalt pentru comunicarea la distanță folosind diverse protocoale de comunicație de nivel înalt (HTTP/FTP/SSH/etc), cât și funncții de nivel mediu (bazate pe protocoalele TCP/UDP/etc..). Implementarea unei aplicații scalabile prin utilizarea "plug-in"-urlior este relativ usor de realizat. Existența unui interpretor JavaScript pentru platforma Java este încă un argument în favoarea folosirii acestui limbaj de programare.

Sistemul de operare

Aplicațiile java sunt extrem de portabile, acestea rulând pe virtual orice sistem de operare și pe orice platformă hardware. Serverul mediului de învățare 3D va rula în varianta pentru GNU/Linux a "Mașinii Virtuale Java" (JVM). Fiabilitatea kernelului sistemului de operare Linux va afecta pozitiv viteza de rulare a mașinii virtuale java cu efect direct asupra performanței aplicației server dezvoltate pentru mediul virtual 3D.

Principii generale de design

Pentru intermedierea comunicării între utilizatori sau pentru orice fel de notificări trimise utilizatorilor s-a ales un tipar cunoscut în domeniul informatic sub numele "Observer Pattern". Această metodă definește și utilizează o dependență $1 \rightarrow n$ între obiecte astfel încât un obiect își modifică starea, toate obiectele dependente sunt notificate. Aplicat, în cazul serverului (1) , când un set de date este prelucrat, utilizatorii (n) sunt notificați.

Pentru a realiza o aplicație scalabilă se utilizează mecanismul de extindere dinamică a fincționalității cu "plug-in"-uri.

5.3 Mecanism pentru obținera scalabilității aplicației server

Se va urmări o cît mai mare flexibilizare a sistemului, astfel încât dezvoltarea ulterioară să fie cât mai facilă. Pentru realizarea acestui obiectiv se va urmări integrarea limbajului de scriptare JavaScript, în serverul sistemului. Se are în vedere folosirea mecanismului de extindere dinamică a funcționalității sistemului prin ”plug-in”-uri.

Utilizatorul, prin intermediul programului client al mediului virtual 3D, expediază **comenzi** și **informații** către server cu **scopul** de a fi prelucrate de către server și returnate sub forma de comezi, ce urmează a fi executate de către aplicația client, cu **efecte** asupra reprezentării mediului virtual 3D.

DEF: *Un* **plug-in** 5.1 *este o componentă software care poate fi integrată de către aplicația gazdă la momentul execuției sau imediat la începutul procesului de execuție. De cele mai multe ori ”plug-in”-urile sunt salvate în formă binară.*

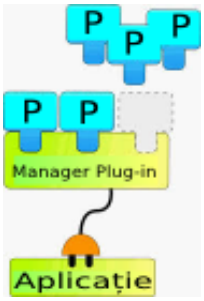

Figura 5.1: Arhitectura Plug-in

5.3.1 Interfața PLUG-IN-ului

”Plug-in”-urile aplicației server implementează următoarea interfață:

```
public interface CommandTypeDependentInterpreter {

      public static final int XML_FORMAT = 1;
      public static final int PALIN_TEXT_FORMAT = 2;
      public void prepareResponse(ServerCommand execCmd, int format);
      String getInterpretedResult();

}
```

Un plug-in extinde funcționalitatea aplicației server prin introducerea unei noi modalități prin care serverul răspunde la o cerere din partea aplicației client. În prima etapă, plug-in-ul preia comanda expediată de aplicația client prin intermediul metodei:

– *prepareResponse(ServerCommand execCmd, int format)* –

În a doua etapă, rezultatul interpretării și procesării datelor de către plug-in se va returna în vederea expedierii către utilizator.

5.3.2 Tipuri de plug-in-uri dezvoltate pentru serverul mediului de învățare 3D

Două **tipuri de plug-in-uri** au fost dezvoltate pentru aplicația server. Primul tip de plug-in este **"clasic"**, implementat folosind 100% limbajul Java. Al doilea tip de plug-in este **"hibrid"**. La inplementarea acestuia sunt utilizate două limbaje de programare: **Java și JavaScript**. Aces tip de plug-in este o noutate.

În principiu, un plug-in hibrid are rolul de a executa la nivelul serverului instrucțiuni codificate în limbaj JavaScript în numele aplicației client. Rezultatele sunt încapsulate în format XML și sunt returnate programului client pentru interpretere. Executarea comenzilor primite de la aplicația client este precedată de următorii doi pași pregătitori: inițializarea unui motor de prelucrare JavaScript (clasa ScriptEngineManager) și concatenera la comanda utilizatorului a unei biblioteci care conține obiecte JavaScript utile.

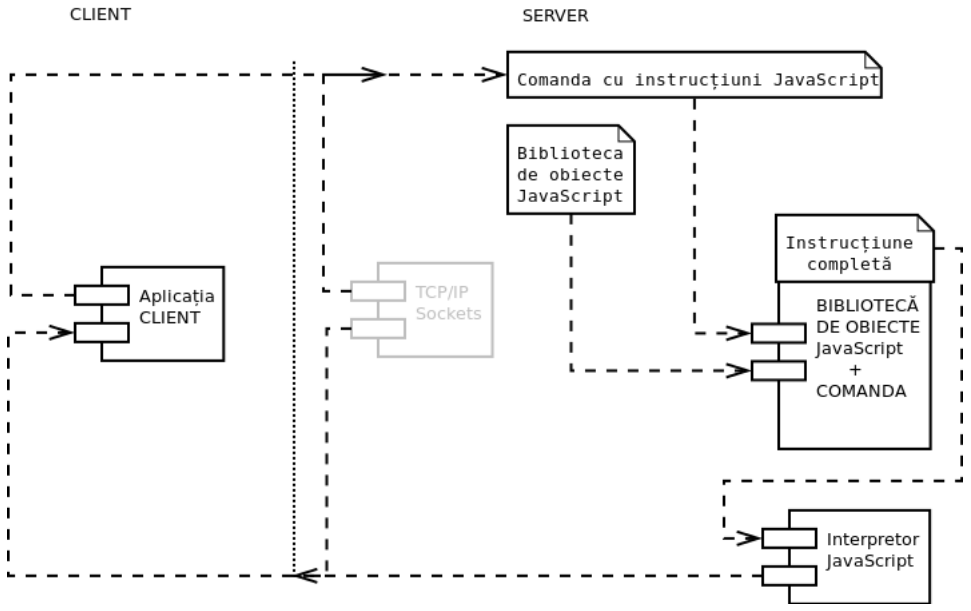

Figura 5.2: Schema Plug-in Hibrid

Avantajele plug-in-urilor hibride:

- Se pot descrie și implementa mult mai ușor și mai rapid.
- Bibloteca de obiecte JavaScript se concatenează la fiecare comandă trimisă de utilizator. Astfel, orice utilizator poate beneficia de obiectele conținute de această bibliotecă de obiecte.
- Codul se execută la nivelul seerverului, cu efect în creșterea vitezei de lucru a programului client.

Dezavantajele plug-in-urilor hibride:

- Comenzile JavaScript invalide pot destabiliza aplicația server.
- Codul se execută la nivelul serverului, cu efect în descreșterea vitezei de lucru a acestuia.
- Folosirea limbajului JavaScript introduce diverse riscuri de securitate a datelor.

5.3.3 Descrierea mecanismului de încărcare a plug-in-urilor

Serverul va ține evidența plugin-urilor disponibile într-un fișier de configurare, organizat în format XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- numele comenzii trimise de client -->
<!-- tipul comenzii (CLASIFICARE) -->
<!-- text ajutător (documentatie) -->
<!-- numărul paramerilor .. si expersia (reg-exp) pt diverse prelucrari -->
<!-- plug-in client delegat sa execute acțiuni în aplicația client -->
<!-- sursa de unde poate fi descărcat ălug-in-ul client -->
<!-- executorul .. este plug-in-ul in aplicația server delegat pentru procesarea datel
<!-- .. de la utilizator (CLIENT) -->
<commands>
.....
<command>
<name>@add</name>
<type>math</type>
<help>addition of 1..n numbers</help>
<param_count>-1</param_count>
<param_regexp>n</param_regexp>
<client_plugin>
   libL3dConsole.so
</client_plugin>
```

```
<client_plugin_source>
   http://aSite.com/plugins/libL3dConsole.so.1.0
</client_plugin_source>
<executor>
   ro.utcluj.learning3d.server.TEST_COMMAND_Math_Adition
</executor>
</command>
....
</commands>
```

Încărcarea dinamică a claselor este un mecanism foarte important pus la dispoziție programatorilor Java. Modul în care clasele sunt efectiv încărcate dinamic este farte puțin cunoscut deoarece procesul este ascuns de către dezvoltatorii limbajului de programare. Mașina virtuală Java (JVM) folosește mai multe rutine de încărcare dinamică a claselor, rutine organizate intr-o ierarhie complicată.

În principiu, pentru a încărca dinamic o clasă Java, programatorul trebuie să obțină o referință către un obiect **ClassLoader**. Un obiect "ClassLoader" poate încărca dinamic o clasă Java prin intermediul funcției **loadClass(..)** ce primește ca parmetru numele clasei de încărcat și returnează o instanță a clasei încarcate în caz de success sau NULL în caz contrar.

Porțiunea de cod folosită pentru a încărca dinamic comenzi pentru serverul mediului de învățare 3D este un exemplu mai potrivit pentru explicarea mecanismului de încărcare a plug-in-urilor în Java.

```
...

try {

  ClassLoader myClassLoader = L3DServerManager.class.getClassLoader();
  String classNameToBeLoaded = execCmd.executor;
  Class<?> myClass = myClassLoader.loadClass(classNameToBeLoaded);
  Object executor = myClass.newInstance();
 ((CommandTypeDependentInterpreter)executor).prepareResponse
  (execCmd, CommandTypeDependentInterpreter.XML_FORMAT);

 return ((CommandTypeDependentInterpreter)executor).getInterpretedResult();

 } catch(..)

 ..
```

5.4 Modul de reprezentare a datelor

În prezent, pentru transmiterea datelor între server și aplicații client se pot folosi mai multe scheme de structurare a informațiilor: informații nestructurate (compuse din șiruri de simboluri și caractere fără o structură impusă) sau informații structurate folosind una dintre schemele cele mai des utilizate precum XML (extended markup language) sau JSON (JavaScript Object Notation).

Informațiile transmise între aplicația server și aplicația client a meduilui de învățare 3D sunt structurate și transmise în format XML.

5.4.1 Scurtă introducere în XML

XML : Limbaj Extensibil de Marcare

- XML este un limbaj de marcare deoarece reprezintă un set de adnotări folosit pentru descrierea unor structuri de date de tip text.
- XML este extensibil deoarece adnotările pot fi definite de către utilizator.
- Nu depinde de nici un limbaj de programare.
- Conține blocuri asemănatoare cu elementele HTML.
- Elementele XML se auto documentează prin utilizarea numelor sugestive pentru transmiterea datelor.

Exemplu:

```
<books>

<book category="computer">
  <author> D Knuth </author>
  <title> The art of computer programming. VOL1 </title>
  <edition>1</edition>
</book>

<book category='Computer Programming'>
  <author> D Knuth </author>
  <title> The art of computer programming. VOL2 </title>
  <edition>1<edition/>
</book>

</books>
```

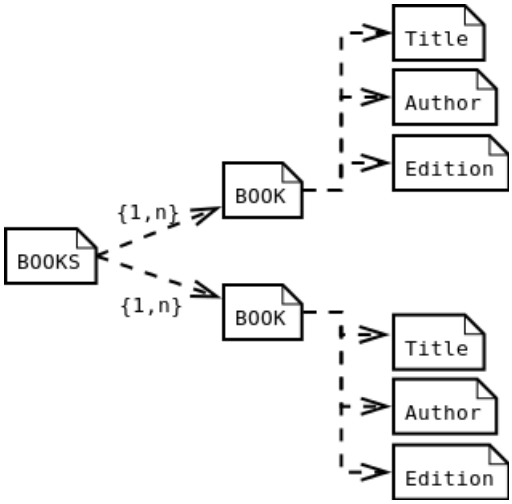

Figura 5.3: Structură arborescentă de date descrisă cu XML

5.4.2 Structura XML a datelor transmise dinspre server spre aplicația client

Atât aplicația server cât și aplicația client au cea mai mare parte din funcționalitate implementată in extensii (sau plug-in-uri). Din acest motiv, doar o porțiune a mesajelor transmise între client și server este standardizată. Restul mesajului trebuie să fie structurat conform nevoilor fiecărei extensii de funcționalitate în parte.

Mesajele transmise către clientul mediului de învățare 3D se clasifică in două categorii:

- Mesaje de eroare (în cazul în care serverul nu reușește să execute cererea clientului).
- Răspunsuri ale serverului la cererea aplicației client.

Structura XML a mesajelor de eroare transmise către aplicația client

```
<?xml version="1.0" encoding="UTF-8"?>"
<response>
  <type></type>
  <exception>
     <message> .... </message>
  </exception>
</response>
```

Structura XML a răspusurilor transmise către aplicația client

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
```

```
  <type> TIPUL COMENZII</type>
  <name> NUMELE COMENZII </name>
  <client_plugin>
     ------  NUMELE PLUG-INULUI CLIENT -----
  </client_plugin>
  <client_plugin_source>
     ------ SURSA ONLINE PT DESCĂRCAREA PLUG-IN-ULUI CLIENT -----
  </client_plugin_source>
  <raw>
     ------  FRAGMENT XML CONFORM CERINȚELOR FIECĂRUI PLUG-IN ------
  </raw>
</response>"
```

5.5 Schema serverului. Diagrama claselor

5.5.1 Modelarea comenzilor primite de server

Clasa **ServerCommand** este imaginea câmpurilor XML ce descriu registurl comenzilor și plug-in-urilor apelabile dinamic de către server. Această structură descrie o comandă.

Rolul clasei **CommandProcessor_step1** este de a despărti textul primit de la aplicația client în subcomponente și de a stoca aceste componente in structura interna a unei instanțe a clasei ServerCommand. Clasa **CommandProcessor** apelează funcțiile celeor două clase amintite anterior pentru a obține textul răspunsului de trimis către aplicația client.

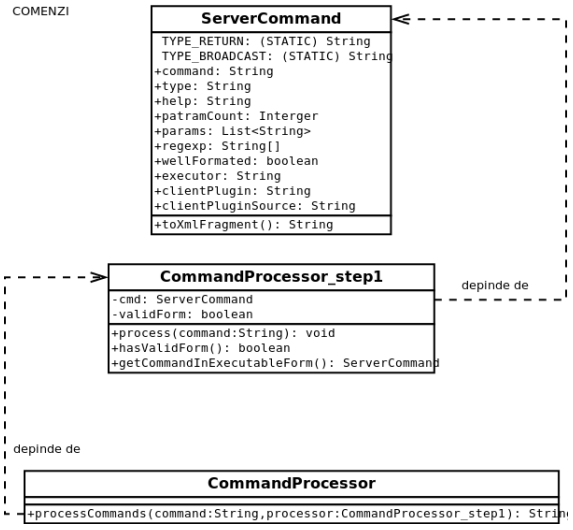

Figura 5.4: UML - COMENZI

5.5.2 Modulul de interpretare a mesajelor XML

Acest modul are la bază o bibliotecă software existentă denumită kxml. Cu ajutorul acestei biblioteci se va construi un interpretor generic pentru datele în format XML (clasele **TagHandler** și **XmlGenericPArser**).

Pentru a interpreta orice text xml este necesară doar o implementare specifică a interfeței "TagHndler" și rularea textului de interpretat cu aceasta prin parserul generic. Clasa **CommandsXmlHandler** este un exemplu de "TagHandler" folosit pentru interpretarea comenzilor primite de la aplicația client.

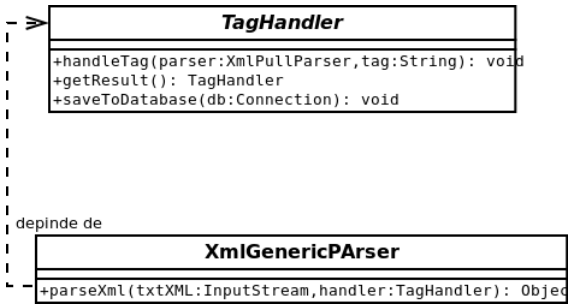

Figura 5.5: UML - PARSER GENERIC XML

5.5.3 Server TCP/IP multi-client

Aplicația server multi-client este un proces cu următoarele elemente de bază:

- Un obiect "java.net.SeverSocket" care facilitează comunicația in rețea. Acesta este setat să aștepte conexiuni pe portul de comunicație 8080.
- O structură repetitivă cu ciclu infinit care crează obiecte "java.net.Socket" pentru fiecere cerere de conexiune validă.
- În ciclul repetitiv infinit se mai crează și câte un obiect special care rulează intr-un fir de execuție separat și care primește ca parametru noul obiect "Socket". Acest obiect special reprezintă coneziunea cu un program client ("ConcreteServerInstance").

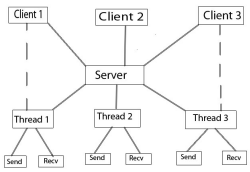

Figura 5.6: Principiul serverului multiclient

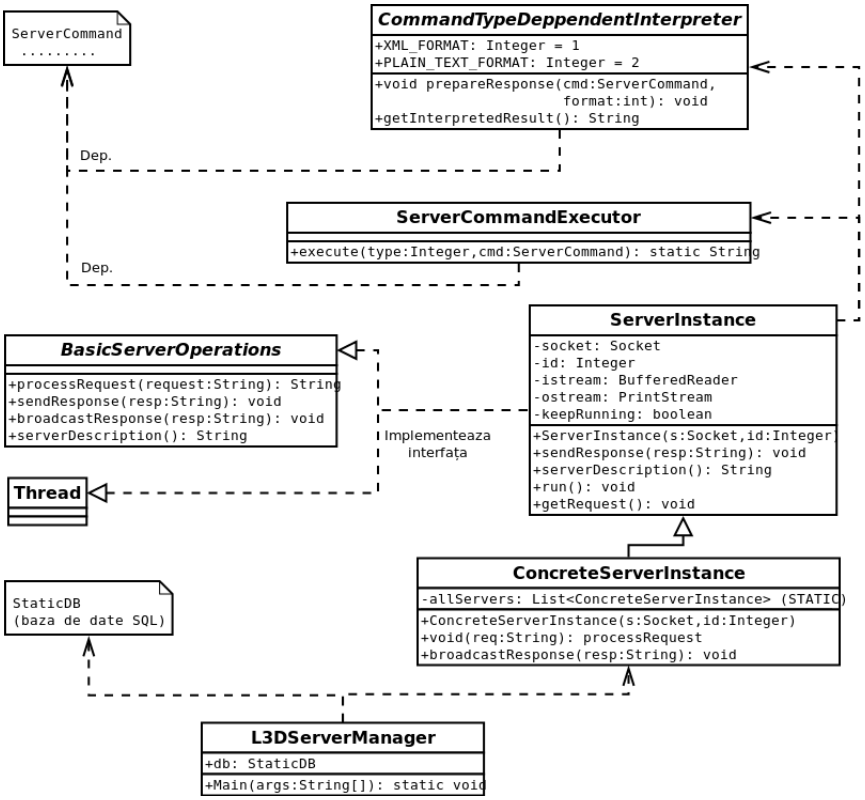

Figura 5.7: UML - Clasele principale ale serverului multi-client

5.6 Testarea mecanismului de încărcare a plug-in-urilor

Testarea modului de funcționare a mecanismului de încăracare a plug-in-urilor se poate efectua și cu o aplicație care să lucreze off-line (fără comunicre în rețea) deoarece testul comunicării on-line a fost efectuat pentru capitolul anterior.

Testul constă în încercerea de a încărca dinamic două clase diferite în funcție de datele de intrare furnizate în linie de comanda, printr-un script. În acest scop se vor concepe două clase care execută două operații aritmetice de bază și implementează următoarea interfață :

```
public interface Operatie {
void evaluate(Integer a, Integer b);
}
```

Testul se rulează de 10 ori. Se consideră **test reușit** dacă pentru toate încercările se incarcă modulul corect. Codul sursă pentru teste și rezultatele sunt listate în anexa B.

Capitolul 6

Design-ul aplicației client

6.1 Introducere

În acest capitol, în secțiunea 6.2, se stabilesc funcțiile aplicației client. În secțiunea 6.3 se prezintă conceptul aplicației software și se enumeră tehnologiile folosite la realizarea acestuia în structura următoare:

- 6.3.1 - Cercetare a modului de livrare a conținutului educativ în mediul 3D.
- 6.3.2 - Tehnologii utilizate pentru redarea 3D a lumii virtuale.
- 6.3.3 - Tehnologii utilizate pentru relizarea interfației grafice (GUI).
- 6.3.4 - Tehnologia utilizată pentru comunicarea cu serverul.

În secțiunea 6.4 se explică mecanismul adoptat pentru atingerea unui scop important, "scalabilitatea aplicației" . În secțiunea 6.5 se explică mecanismul CERERE-PROCESARE-RASPUNS. În aceeași secțiune se prezintă formatul de reprezentare a datelor transmise între aplicația client și aplicația server. Secțiunea 6.6 este dedicată prezentării diagramei claselor. În secțiunea 6.7 se enunță parametrii testării aplicației client. Rezultatele acestei testări expuse în același capitol.

6.2 Funcțiile aplicației client.

Aplicația client are scop demonstrativ. Implementarea se va executa în cel mai simplu mod posibil, cu interfețe utilizator minimale. Funcțiile de bază ale aplicației client sunt enumerate dar implementarea nu se execută în integralitate.

Funcții de bază:

- Redarea 3D a obiectelor lumii virtuale.
- Schimbul de date între aplicație și server, după modelul CERERE - RĂSPUNS.
- Facilitarea comunicării directe între participanții la procesul de învățare în mediul virtual.
- Reprezentarea utilizatorilor în mediul virtual prin intermediul avatarelor. Fiecare utilizator va fi reprezentat de un avatar identificabil de către ceilați utilizatori.
- Facilitarea cooperării între utilizatori.
- Aplicația, în anumite instanțe, va asista utilizatorul în procesul de învățare.

6.3 Conceptul aplicației și tehnologiile utilizate.

Interfața aplicției cu utilizatorul este esențială pentru asugurarea unei experințe cât mai plăcute în timpul utilizării. Pentru realizarea interfețelor se vor respecta următoarele aspecte: simplitate, etichete și denumiri sugestive.

Designul interfeței grafice 2D este minimal. Se va dezvolta în principal componenta de redarea 3D a obiectelor din mediul virtual.

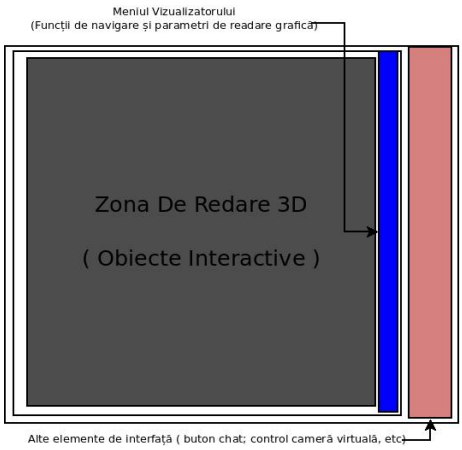

Figura 6.1: Dispunerea elementelor în interfața grafică

6.3.1 Modul de livrare a conținutului educativ în mediul 3D

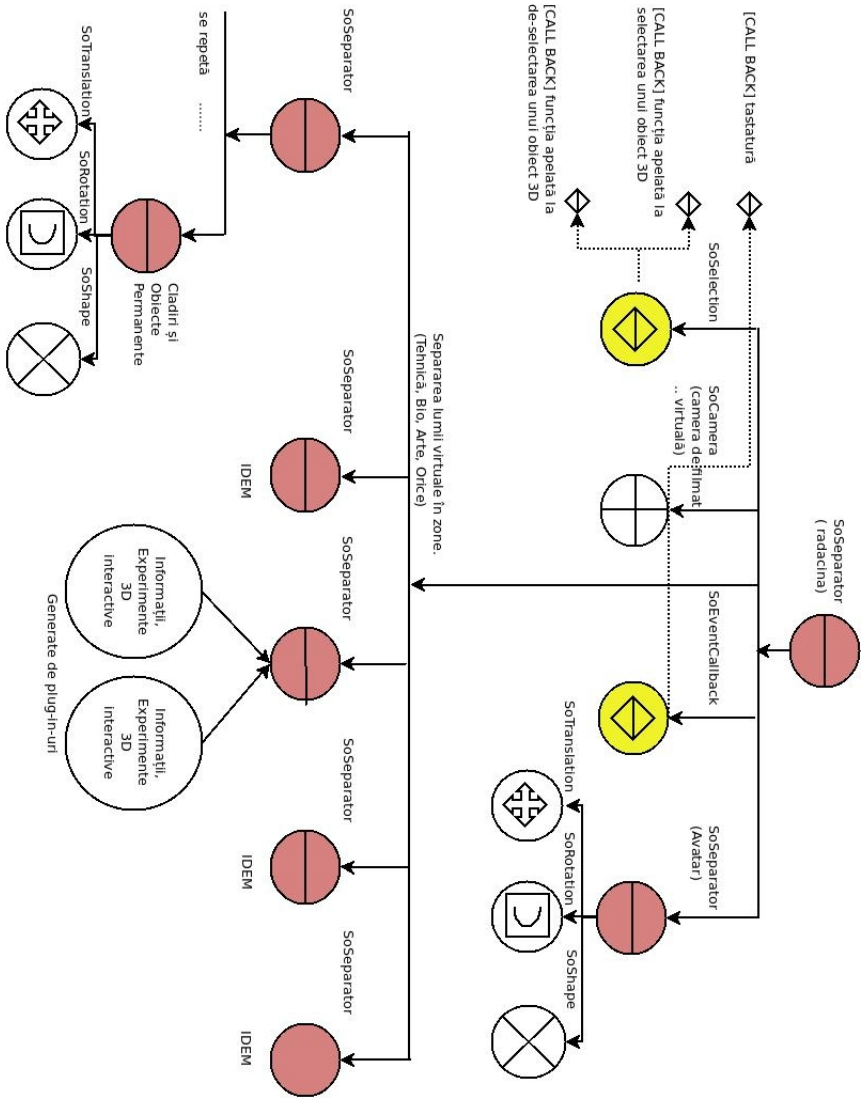

Figura 6.2: Mediul 3D și materialul educativ interactiv

6.3.2 Tehnologii utilizate pentru redarea 3D a mediului de învățare.

Sistemul de operare

Sistemul de operare ales pentru dezvoltarea aplicatiei este GNU/Linux (sitemul de operare GNU + kernelul Linux), distribuția Ubuntu, un sistem de operare modern, foarte asemanător sistemului de operare Unix, dezvoltat de către comunitățile "Free Foftware" și "Open Source".

Motivele alegerii acestui sistem de operare sunt urmatoarele:

- Stabilitatea sistemului de operare.
- Performanța sistemului de operare.
- Simplitatea în instalare și menținere.
- Multitasking și managementul memoriei foarte eficient implementate.
- Compilatoarele, librariile grafice și cele mai importante instrumente pentru dezvoltarea de medii 3D sunt ușor de instalat șconfigurat, sunt gratuite și bine documentate pentru GNU/Linux.

Limbajul de programare

Limbajul de programare ales pentru realizarea aplicației client este C++. Sunt motivat în alegerea mea de urmatoarele caracteristici ale acestui limbaj:

- Aplicațiile programate în C++ ruleaza vizibil mai rapid decat aplicațiile realizate în alte limbaje de programare.
- O multitudine de biblioteci software utile sunt implementate în C++.
- Aplicația finală poate fi compilată pentru mai multe platforme.
- C++ este limbajul de programare preferat de dezvoltatorii aplicatii bazate pe grafica 3D.

Compilatorul ales este g++ din colectia GCC (Gnu Compilers Collection).

Limbje și biblioteci software utilizate pentru reprezentarea mediului virtual 3D

Când se alege o bibliotecă 3D, factorii care trebuiesc luaţi în considerare sunt: performanța, existența şi calitatea documentaţiei, platformele care acceptă biblioteca grafică 3D, existenţa unui format standardizat pentru a descrie obiectele grafice şi interacţiunile obiect, licenţiere etc. Ţinând cont de criteriile menţionate mai sus s-a ales biblioteca Coin3D cu suport VRML (Virtual Reality Modeling Language).

Coin 3D (Open Inventor reimplementat cu suport pentru V.R.M.L.)

Generalități

Coin se bazează pe librăria grafică 3D OpenGL (The Open Graphics Library), și își are rădăcinile în Open Inventor 2.1 API, cu care Coin este încă compatibil. Precursorul bibliotecii Coin3D, Open Inventor, își bazează modul de păstrare și manipularea a obiectelor grafice, pe libraria și structura claselor C++ original proiectată pentru SGI.[4]

Open Inventor a devenit repede după lansare biblioteca grafică standard de facto pentru vizualizare 3D și software de simulare vizuală în comunitatea științifică și inginerească, și mai târziu, bază pentru formatul standard VRML1.

Există multe publicații pe subiectul Open Inventor, cele mai importante fiind: "The Inventor Mentor", "The Inventor Toolmaker", amandouă fiind recomandate pentru cei ce doresc să învețe sa utilizeze Open Inventor. Aceleași materiale de studiu se pot folosi pentru inițierea în utilizarea librariei grafice Coin3D.[4]

Coin3D a fost dezvoltată independent, de la zero înainte ca Open Inventor să devină open source. Coin și-a atins țelul compatibilității cu standardul Open Invetor 2.1 în toamna anului 2000 și de atunci a fost și-a dezvoltat foarte mult numarul de caracteristici, variind de la suportul pentru sunetul 3D (deficitar pentru sistemul de operare GNU/Linux) la suportul GLSL shader, formate de fișiere suplimentare cum ar fi VRML97, și un numar larg de schimbări interne pentru a ține pasul cu noul OpenGL, optimizat cu o varietate mare de tehnici ce nu erau disponibile la început.[4]

Istoric

Coin își are începuturile în anul 1995, fiind conceput ca o librarie de redare grafică pentru standardul VRML 1.0 . Initial a fost dezvoltată plecând de la librăria Qv a SGI care are rolul de a analiza fisierele în format VRML 1.0. După ani de extindere anevoioasă a bibliotecii soft, începând cu noi funcționalități de redare și exportare precum VRML 1 și VRML 2, în anul 1997 s-a ajuns la o reală nevoie de reproiectatare.[4]

La prima vedere API-ul are o asemănare izbitoare cu Open Inventor. Conceptele utilizate de către Open Inventor sunt deseori amintite ca o bună metodologie de proiectare în multe cărti de inginerie software, astfel că o parte dintre programatorii care s-au ocupat de dezvoltarea acestuia și care aveau deja experiența cu această librarie, au luat conceptul Open Inventor ca un exemplu demn de urmat. Concomitent cu identificarea strategiei de rescriere a codului, s-a luat decizia de colaborare cu membrii comunității Free Software care au contribuit cu o serie de idei tehnice interesante. Ca o urmare firească a acestei colaborări, libraria s-a dezvoltat ca o alternativă liberă la varianta Open Inventor, cu suport pentru sistemele de operare GNU/Linux, IRIX, Windows, Cygwin și Mac OS X.[4]

V.R.M.L. (Virtual Reality Modeling Language)

VRML este un limbaj de modelare a lumilor virtuale și un standard internațional pentru descrierea formelor și a scenelor pentru World Wide Web. Organizația responsabilă cu dezvoltarea și publicarea standardului VRML este "Web 3D Consortium" anterior

cunoscută sun numele ”VRML Consortium”. În prezent standardul a ajuns la versiunea a doua.

În principiu, o lume virtuală descrisă cu limbajul VRML se poate reprezenta printr-o structură arborescentă în care „nodurile și frunzele” arborelui sunt noduri VRML. [5]

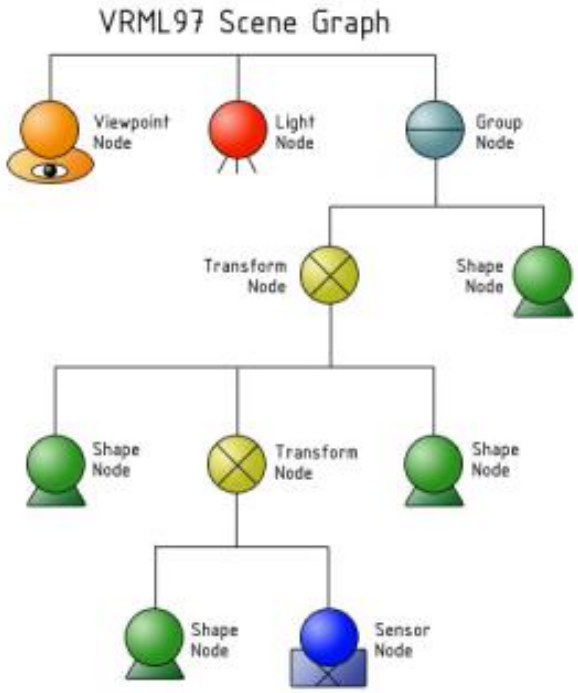

Figura 6.3: Graf VRML (exemplu)

Modul în care lumea virtuală este afișată pe display-ul computerului este influențat de ordinea și amplasarea nodurilor în arbore. Fiecare nod este redat în mod grafic după următoarele reguli:[6] [7]

- Redarea unui grup de noduri se face în ordine, de regulă, de la stânga la dreapta. Aplicând această regulă la ilustrația precedentă 6.3, putem trage concluzia că nodurile de tip ”transformări” (transform) au efect asupra nodului ”Formă geometrică” (Shape Node) doar dacă sunt amplasate înaintea acestuia.

- Fiecare nod execută propria afișare și afectează starea următoarelor noduri prin modificarea proprietăților membre. Spre exemplu, un obiect al carui suprafață este reflexivă sau radiantă din puntul de vedere al luminozității, poate afecta modul în care urmatoarele noduri sunt iluminate.

- Nodurile de tip ”transformare” (Transform Node) se influențează prin combinare (ex: două rotații de 90° se acumulează formând o rotație de 180°).

- Nodurile de tip ”formă geometrică” sunt afișate în modul dictat de starea actuală (ultima culoare setată, ultima luminozitate setată).
- Nodurile de tip SoSeparator (separatoare), spre diferență de SoGroup (grup), salvează starea actuală înainte de a-i parcurge și afișa nodurile membre. Această stare este restaurată după parcurgerea separatorului. Astfel, nodurile din separator nu afectează nodurile următoare, fiind izolate.
- Nodurile sunt ”construite” la punctul de coordonate XYZ = (0,0,0) dacă nu sunt afectate de nici o transformare.
- Sistemul de coordonate VRML este următorul : pentru valori pozitive, axa Y este orientată în sus, axa X este orientată spre dreapta și axa Z este perpendiculară pe axele XY și este orientată spre observator. 6.4
- Pentru efectuarea rotațiilor se aplică regula mâinii drepte. Sensul pozitiv al rotației este sensul invers acelor de ceasornic. Rotația se exprimă în grade radian. 6.5

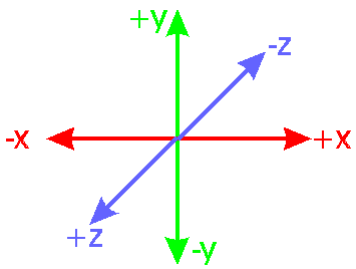

Figura 6.4: Axe

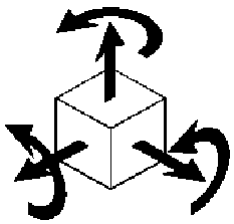

Figura 6.5: Rotație

Animații descrise în VRML

Scenele 3D descrise cu ajutorul limbajului VRML pot fi animate și pot fi programate să răspundă la comenzile utilizatorului. Acestă interacțiune se poate realiza în două moduri: programatic, prin accesarea și modificarea nodurilor grafului scenei VRML prin rutinele unui program dezvoltat de utilizator (în C++/Python/etc); sau prin folosirea

mecanismulor de descriere a interacțiunilor și animațiilor pus la dispoziție de limbajul VRML.

Interactivitatea în VRML se realizează prin transmiterea de valori ale ”proprietăților” între diversele noduri ale grafului VRML. Când un nod trimite informații către un alt nod, se crează un **eveniment (”event”)**, o structură ce transportă două informații date (valori) și informație referitoare le timp (timestamp) pentru exprimarea momentului în care evenimentul s-a produs, cu scopul asigurării integrității logice pentru secvențele de evenimente [CITE COMENT Rohel]. Mecanismul animației presupune crearea unei conexiuni explicite între noduri cu ajutorul construcției sintactice **ROUTE** între ”câmpurile” ce stochează atributele nodurilor implicate în animație. Majoritatea nodurilor sunt construite după structura:

```
nod {
   camp              (field)         ; inaccesibil
   eveniment primit  (eventIn)       ; permite doar scriere
   eveniment emis    (eventOut)      ; permitere doar citire
   câmp expus        (exposedField)  ; permite citire și srcirere
}
```

Evenimentele transmise prin câmpul eventOut al unui nod, vor putea fi recepționate de un al doilea nod prin câmpul eventIn, având ca efect animația nodului receptor. 6.6

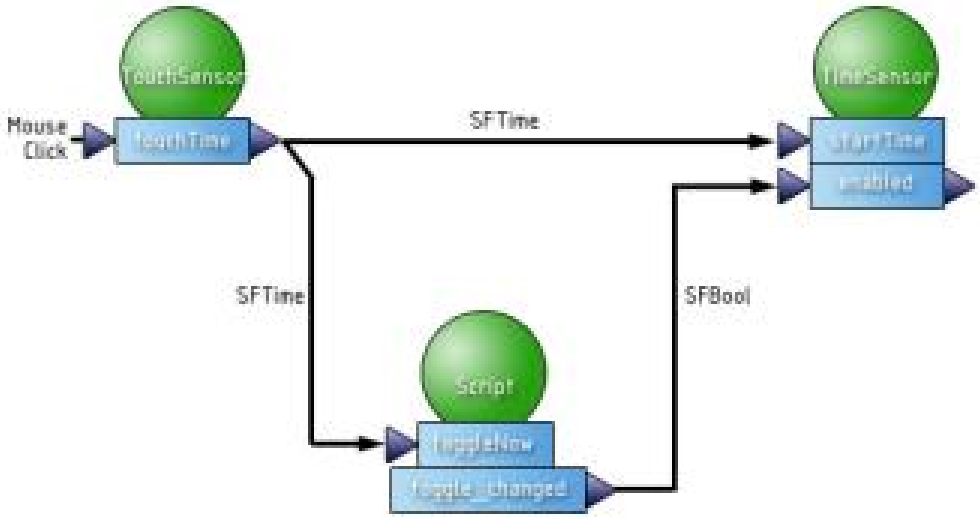

Figura 6.6: ANIMAȚIE VRML

Exemplu: (Orientarea obiectului grafic se modifică în timp, din momentul activării senzorului de atingere)

```
# VRML V2.0 utf8

DEF Touch TouchSensor { }
DEF Timer1 TimeSensor { }
DEF Rot1 OrientationInterpolator { }

DEF Frame1 Transform {
children [
Box { }
]
}

ROUTE Touch.touchTime TO Timer1.set_startTime
ROUTE Timer1.fraction_changed TO Rot1.set_fraction
ROUTE Rot1.value_changed TO Frame1.set_rotation
```

6.3.3 Tehnologii utilizate pentru relizarea interfaței grafice (GUI)

Interfețele grafice se vor implemeta folosind atât bibilotecile software Qt/SoQt cât și tehnologii Web/HTML. Interfețele Web sunt adoptate deoarece implementarea lor necesită mai puțin timp și sunt mult mai dinamice decât interfețele implemetate în c++ utilizând bibliotecile Qt/SoQt.

Qt

Qt este un cadru software pentru programarea aplicațiilor multi-platformă și pentru dezvoltarea de interfețe grafice folosind ca limbaje: C++, QML, CSS și JavaScript. Librariile Qt și uneletele de dezvoltare sunt eliberate prin licențe compatibile cu modelul "open source".

Qt implementează metode inteligente pentru realizarea comunicării între divesrele obiecte ale interfeței grafice. Metoda constă în conectarea evenimentelor prelute de elementele de interfță la rutinele asociate cu aceste evenimente sau conectarea la alte evenimente. Acest lucru se realizează automat, cu ajutorul unui meta-compilator care generează codul necesar conectarii evenimentelor (SIGNAL) la rutine (SLOT). 6.7 [8]

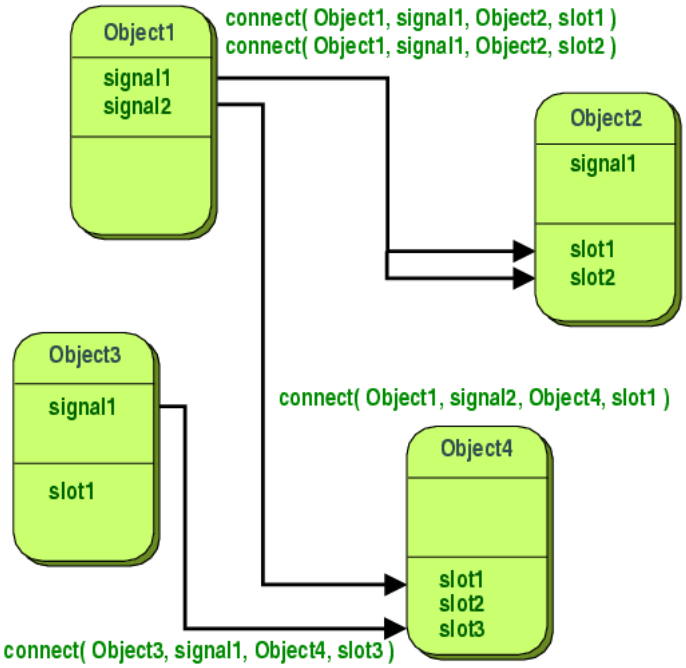

Figura 6.7: Mecanismul SIGNAL-SLOT

SoQt

SoQt furnizează programatorului o interfață de nivel înalt în C++. Librăria, include o ierarhie de clase cu rol de componentă de vizualizare (viewers), cu funcționalități diferite, împreună cu o serie de modalități de control a interacțiunii dintre camerele virtuale de filmat și scena 3D. SoQt este practic liantul dintre librăriile de interfețe Qt și Coin 3D.

Interfețe Web/HTML prin mecanismul "QtWebKit Bridge"

Mecanismul "WebKit bridge" este parte a bibliotecii sotware Qt. Prin acest mecanism se facilitează accesul obiectelor codificate in JavaScript la rutinele codificate în c++ în aplicația realizată cu ajutorul bibliotecii software Qt. În acest fel, o pagină web poate fi programată ca o interfață pentru un program codificat în c++. [8]

6.3.4 Tehnologia utilizată pentru comunicarea cu serverul

Viteza de execuție a aplicațiilor codificate în C++ și existența librariei Coin3D, este motivul alegerii acestui limbaj pentru implementarea aplicației client. Ultimul standard adoptat de ISO include extensii pentru limbaj și adaugarea de module noi la librăria standard. Cea mai binevenită modificare este suportul pentru programarea firelor de execuție în modelul tehnicii OOP.

Pentru programarea rețelelor de calculatoare nu există suport modern, acest aspect fiind în agenda ISO pentru următorul standard. Pentru acest "task" am folosit librăria Boost Asio, o subcomponentă a biblotecii software Boost.

Boost Asio implemetează un model modern bazat pe modelul OOP, pentru programarea asincronă a operațiilor I/O la nivel scăzut și programarea aplicațiilor pentru rețele de calculatoare (cu asincronism în ce privește trimiterea/primirea de date în rețea). 6.8

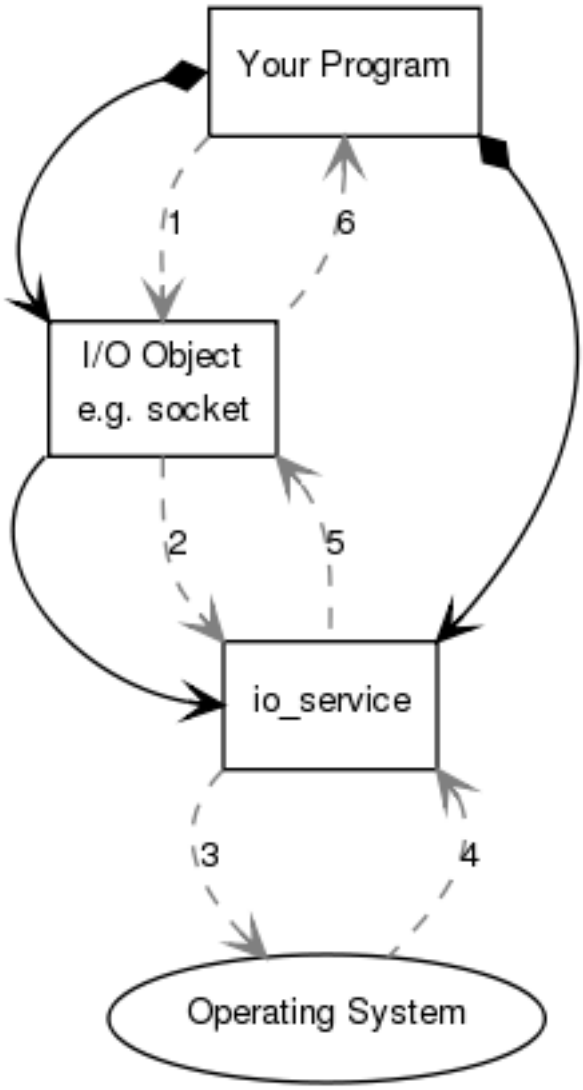

Figura 6.8: Mecanismul I/O asincron

6.4 Decizie privind design-ul aplicației client.

Se va urmări o cît mai mare flexibilizare a sistemului, astfel încât dezvoltarea ulterioară sa fie cât mai facilă. Se are în vedere folosirea mecanismului de extindere dinamică a funcționalității sistemului prin "plug-in"-uri. Plugin-urile vor putea fi descărcate din surse externe în timpul rulării aplicației client, și incluse dinamic în aplicație, fără a se întrerupe rularea programului. Un plug-in va prelua funcția de încărcare de cursuri interactive și activități educative diverse.

Un plug-in respectă următoarea interfață:

```
#ifndef L3DCLIENT_PLUGIN_H
#define L3DCLIENT_PLUGIN_H

#include <Inventor/nodes/SoSeparator.h>
#include <string>

  //interfața plug-in
  extern "C" void get_plug_in( std::string data, SoSeparator* insertionPoint );
  //pointer la o funcție cu aceeiași parametri și tip returnat ca plug-in-ul
  typedef void (*PLUG_IN)( std::string , SoSeparator*);

#endif
```

Plug-inurie sunt în fapt funcții standardizate, compilate în librării dinamice "Shared Object". Descărcarea librăriilor de pe internet se va realiza prin apelarea comenzii Linux wget.

Compilarea bibliotecilor software dinamice se efectuează cu specificarea unor parametri speciali pentru compilator. [**?**]

- **-Wall**, afișează toate mesajele de alarmă
- **-fPIC**, "Position independent code"; pubctul de încărcare a instrucțiunilor variază la bibliotecile dinamice.
- **-c**, doar compilare
- **-Wl,-snoname,<name>**, instrucțiuni către link-editor pentru a genera obiecte dinamice

Următorul script crează biblioteci software dinamice ce pot utiliza clase Qt, Boost și Coin:

```
#!/bin/bash
# ->  mod de foflosire:
# $>  build_lib.sh <sursa.cpp> <denumire fisier obiect> <nume biblioteca>
# ->  DOAR sursei .CPP i se adauga extensia !!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
CPPFLAGS_COIN=‘soqt-config --cppflags‘
CXXFLAGS_COIN=‘soqt-config --cxxflags‘
LDFLAGS_COIN=‘soqt-config --ldflags‘
LIBS_COIN=‘soqt-config --libs‘

CPPFLAGS_QT=‘soqt-config --cppflags‘
CXXFLAGS_QT=‘soqt-config --cxxflags‘
LDFLAGS_QT=‘soqt-config --ldflags‘
LIBS_QT=‘soqt-config --libs‘
INCLUDEDIR_QT=" -I/usr/include/qt4 -I/usr/include/qt4/QtCore \
                -I/usr/include/qt4/QtGui -I/usr/include/qt4/QtOpenGL \
                -I/usr/include/qt4/QtWebKit "

BOOST_CPPFLAGS=" -pthread "
BOOST_FILESYSTEM_LDFLAGS=" -L/usr/local/lib -Wl,-R,/usr/local/lib "
BOOST_FILESYSTEM_LIBS=" -lboost_filesystem "
BOOST_SYSTEM_LDFLAGS=" -L/usr/local/lib -Wl,-R,/usr/local/lib "
BOOST_SYSTEM_LIBS=" -lboost_system "
BOOST_THREAD_LDFLAGS=" -L/usr/local/lib -Wl,-R,/usr/local/lib "
BOOST_THREAD_LIBS=" -lboost_thread -lboost_system -pthread "

#
# [ -Wall ] display all warnings
# [ -fPIC ] generate Position Independent Code
#
#  MODEL g++ -c -Wall -fPIC plugin1.cpp  -o plugin1.o

g++ -std=c++11 -c -Wall -fPIC $INCLUDEDIR_QT $1 -o $2.o

#
# [ -Wl,-snoname,<name> ] instuct the linker the
# linker to generate a Dynamic Shared Objects
# instead of an application
#
g++ -std=c++11 $BOOST_CPPFLAGS $CPPFLAGS_COIN $CXXFLAGS_COIN $CPPFLAGS_QT  \
     $CXXFLAGS_QT -I$INCLUDEDIR_QT -shared -Wl,-soname,$3.so -o $3.so $2.o \
     $BOOST_SYSTEM_LDFLAGS $BOOST_SYSTEM_LIBS $BOOST_THREAD_LIBS  \
     $BOOST_THREAD_LDFLAGS $BOOST_FILESYSTEM_LDFLAGS $BOOST_FILESYSTEM_LIBS \
     $LDFLAGS_COIN $LIBS_COIN  $LDFLAGS_QT $LIBS_QT
```

Exemplu de implementare a unui plug-in

Exemplul listat este plug-in-ul care are rolul de a crea si anima avatarele celorlalți utilizatori. Acestea își schimbă orientarea și poziția în spațiu la fiecare 0.5 secunde. Parametrii furnizați de aplicație plug-in-ului reprzintă:

- Un număr variabil de informații și parametri primiți de la server, în format XML, reprezentînd numele acțiunii ed executat la nivelul aplicației client și datele necesare pentru implemetarea acestei acțiuni.
- Un pointer spre un "separator" al lumii virtuale, reprezentând zona în care programatorul plug-in-ului poate insera grafica 3D dorită.

```
#include "../src/plugin.h"
#include "../src/vrml_utils.h"
#include <Inventor/nodes/SoSeparator.h>
#include <Inventor/nodes/SoTranslation.h>
#include <Inventor/nodes/SoRotation.h>
#include <Inventor/nodes/SoCone.h>
#include <string>
#include <iostream>
#include <cstdlib>
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/xml_parser.hpp>
#include <sstream>

#include "../src/vrml_utils.cpp"

extern "C" void get_plug_in( std::string data, SoSeparator* insertionPoint )
{

  using boost::property_tree::ptree;
  ptree pt;

  std::stringstream streammed_msg;
  streammed_msg<<data;

  read_xml(streammed_msg, pt);

  std::string action = pt.get<std::string>("response.action");
  std::string gender = pt.get<std::string>("response.avatar_type_id");
  std::string name = pt.get<std::string>("response.other_user");

  std::string rotN = "MOC_ROTATE_"+name;
```

```
std::string movN = "MOC_MOVE_"+name;

//create grapichs first time only
SoSeparator* _3d_obj = (SoSeparator*)findNode(name.c_str(),insertionPoint);

std::cout<<std::endl<<data;
if( _3d_obj == NULL )
  {
    std::cout<<std::endl<<"CREATE A MOC AVATAR";

    SoTranslation *translation = new SoTranslation;
    translation->translation.setValue(0.1,0,0);
    translation->setName( movN.c_str() );
    SoRotation *rotation = new SoRotation;
    rotation->rotation.setValue(SbVec3f(0, 1, 0), 0.1);
    rotation->setName( rotN.c_str() );

    _3d_obj = new SoSeparator;
    _3d_obj->setName(name.c_str());
    _3d_obj->addChild( translation );
    _3d_obj->addChild( rotation );

    if ( gender == "0" )
{
_3d_obj->addChild(
    get_scene_graph_from_file("/usr/local/share/l3dclient/avatar_male.wrl")
    );
}
    else
{
_3d_obj->addChild(
    get_scene_graph_from_file("/usr/local/share/l3dclient/avatar_female.wrl")
    );
}

    insertionPoint->addChild( _3d_obj );
  }
//end create graphics

//execute code for the current action
if ( action == "rotate" )
  {
```

```
      std::string s_angle = pt.get<std::string>("response.degrees");
      float angle = std::stof( s_angle.c_str() );
      SoRotation* rotation = (SoRotation*) findNode( rotN.c_str(), insertionPoint );
      if ( rotation == NULL )
{
  rotation = new SoRotation;
  rotation->setName( rotN.c_str() );
  _3d_obj->addChild( rotation );
}

      rotation->rotation.setValue(SbVec3f(0, 1, 0), angle);
    }
  else if ( action == "move" )
    {
      float x,z;

      std::string s_X = pt.get<std::string>("response.x");
      x = std::stof( s_X.c_str() );
      std::string s_Z = pt.get<std::string>("response.z");
      z = std::stof( s_Z.c_str() );

      //SbVec3f pos(x,y,z);
      SoTranslation* translation = (SoTranslation*) findNode( movN.c_str(), insertionl
      if ( translation == NULL )
{

  translation = new SoTranslation;
  translation->setName( movN.c_str() );
  _3d_obj->addChild( translation );

}
      translation->translation.setValue(x,0.0,z);

    }
  else if ( action == "talk" )
    {
// TO DO ... chat action
    }


}
```

6.5 Comnezi către server

Instrucțiunile trimise către server reprezintă șiruri de caractere ce se încadrează în următorul tipar:

```
@<COMANDA> <parametrul 1> <parametrul 2> ... <parametrul 2>
```

Răspunsul primit de aplicația client este în format XML în una din următoarele forme :

```
Excepții / Mesaje de eroare
<?xml version="1.0" encoding="UTF-8"?>"
<response>
   <type></type>
   <exception>
     <message> .... </message>
  </exception>
</response>

Mesaje Valide
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <type></type>
  <name></name>
  <client_plugin></client_plugin>
  <client_plugin_source></client_plugin_source>
  <raw> </raw>
</response>
```

Schimbul de date între client și server formează o buclă completă (CLIENT)-(SERVER)-(CLIENT/CLIENȚI). Primul pas este reprezentat de expedierea unei comenzi dinspre client către server, sub forma unui șir de caractere. Primul caracter este "@" urmat fără spațiu de un nume unic ce are rol de "selector" de operație la nivelul serverului. Selectorul poate fi urmat de unul sau mai mulți parametri separați prin spații. Comanda expediată, la nivelul serverului este despărțită în elementele constituente: selector de comandă și lista de parametri. Selectorul de comandă este folosit pentru a obține din registrul de comenzi a serverului următoarele informații utile:

- Numele plug-in-ului ce va prelucra răspunsul serverului la nivelul aplicției client.
- Adresa web de la care aplicația client poate încărca plug-inul client.
- Numele plug-in-ului care va procesa la nivelul serverului comanda primită de la aplicația client.

Pe baza informațiilor de mai sus, serverul își incarcă propriul plug-in de prelucrare a comenzii de la aplicația client (cu parametrii necesari), execută prelucrarea datelor primite de la client și returnează o comanda în format XML cu următoarele informații:

- tipul comenzii în blocul ”type”
- numele comenzii în blocul ”name”
- numele plug-in-ului ce se va executa la nivelul aplicației client, în blocul XML ”client_plugin”
- adresa web de la care acest plug-in se poate descărca, în blocul XML ”client_plugin_source”
- parametrii de furnizat plug-in-ului client, în blocul XML ”raw”

Această comandă este transmisă aplicației client sau mai multor clienți, în funcție de tipul comanzii. Aplicația client preia mesajul XML și execută în ordine următoarele operații:

- interpretează comanda XML pentru a extrage numele plug-in-ului asociat acestei comenzi
- extrage pararmetrii de furnizat plug-in-ului
- verifică dacă plug-in-ul este deja incărcat in memorie
- dacă plug-in-ul nu este încărcat, se execută încărcarea dinamică a acestuia de la adresa extrasă din comanda XML
- plug-in-ul se execută cu lista de parametri extrasă din comandă

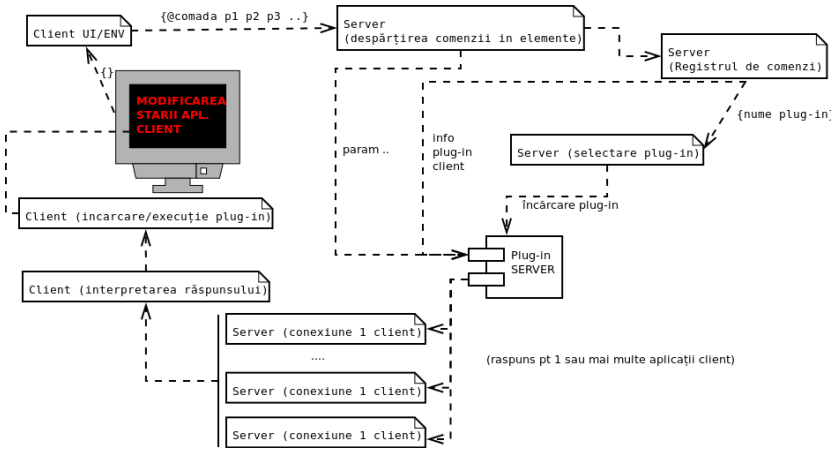

Figura 6.9: Bucla schimbului de informații între client și server

6.6 Diagrama claselor

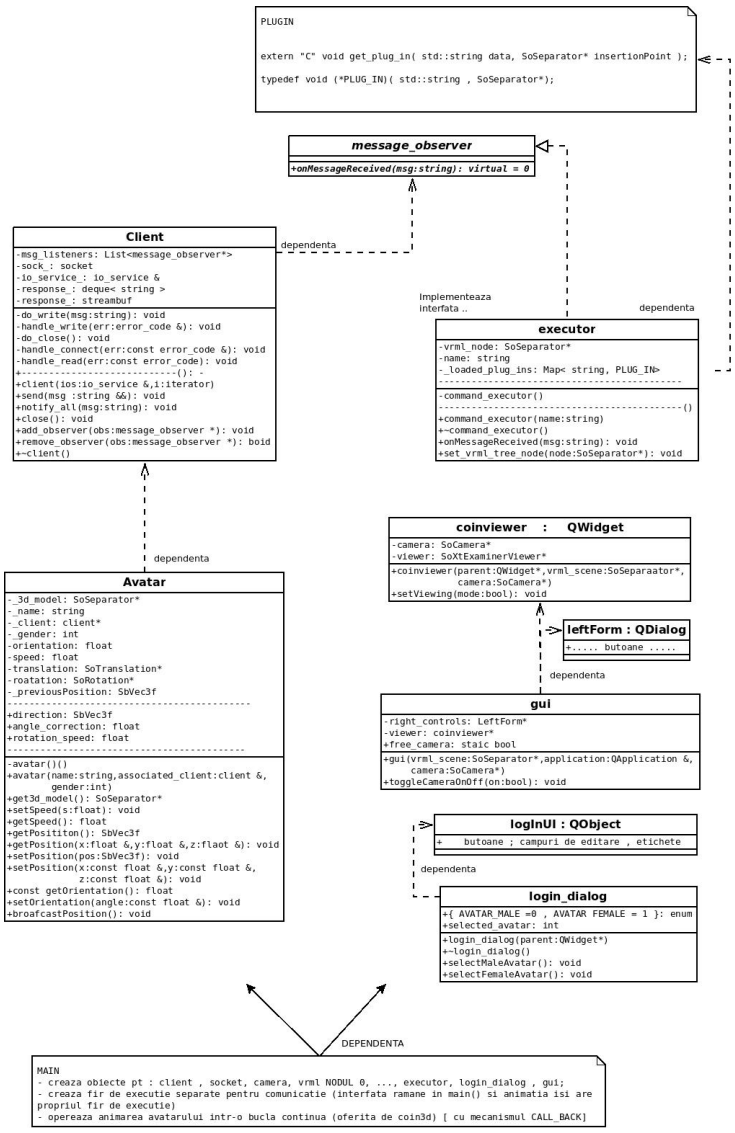

Figura 6.10: Diagrama claselor aplicației client

6.7 Test

Deoarece o metodă automată pentru testarea întregii aplicații este foarte complexă, se recurge la testarea vizuală, constând în verificarea modului în care perspectiva vizuală dintre două avatare este corectă, și în verificarea modului în care mediul virtual este redat din punct de vedere estetic.

Figura 6.11: Test - avatare - față în față

Figura 6.12: Test - avatare aliniate la intrarea într-o fortificație romană (3D)

Capitolul 7

Instrucțiuni

7.1 Instalarea bibliotecilor software

Sub Linux, comenzile de instalare a programelor pot să difere. Se or expune etapele de urmat pentru instalarea bibliotecilor software în Ubuntu Linux. Această distribuție beneficiează de un depozit de aplicații și biblioteci software. Acestea se pot instala atât cu ajutorul unor programe speciale ce dispun de o interfață grafică, cât și cu ajutorul comenzii **apt-get**.

```
$> sudo apt-get install build-essential libpng++-dev libtool libboost-all-dev
               autoconf autotools-dev libqt4-core libqt4-gui libqt4-dev
               mercurial git gitk libcoin80 libcoin80-doc libcoin80-dev
               libcoin80-runtime
```

Această comandă va instala următoarele:

- Compilatoarele, link-editoarele și bibliotecile necesare pentru programarea în C++.
- Codec pentru formatul PNG.
- Aplicațiile de generare a scripturilor Makefile.
- Bibliotecile BOOST.
- Bibliotecile Qt v4.
- Software pentru controlul versiunii (Mercurial și Git).
- Bibliotecile software Coin3D

Nu există în depozitul Ubuntu Linux - bibliotecile software **SoQt** și **SpiderMonkey JavaSctipt**, conținând liantul dintre Qt4 și Coin3D, respectiv interpretorul JavaScript utilizat de platforma Coin3D. Acestea se vor instala manual.

```
#   pt. SoQt

#  descărcarea codului sursă
$> hg clone http://www.bitbucket.org/Coin3D/SoQt
#  compilarea codului sursă și instalarea
$> cd soqt
$> ./configure
$> make
$> sudo make install


#  codul sursă pt SpiderMonkey  este disponibil pe CD-ul anexat acestei lucrări
$> tar -xvf js-1.60.tar.gz
$> cd js-1.60
#  compilarea
$> make -f Makefile.ref
#  instalarea
$> cp *.h /usr/include/js
$> cp Linux_All_OPT.OPJ/libjs.so /usr/lib
exit
```

7.2 Compilarea aplicației

Aplicația client este disponibilă pe CD-ul anexat la acestă lucrare sau pe contul git : **https://github.com/gabrielchmod777/virtualWorldVRML-CLIENT.git**.

```
#  Pregătiri anterioare

$> export LDLIBRARY_PATH=/usr/local/lib:LD_LIBRARY_PATH
$> export COIN_ALLOW_SPIDERMONKEY=1

#  descărcarea codului sursă
$> git clone https://github.com/gabrielchmod777/virtualWorldVRML-CLIENT.git
$> cd cd virtualWorldVRML-CLIENT
$> autoreconf -i
$> ./configure
$> make
$> sudo make install
```

Aplicația server este disponibilă pe CD-ul anexat la acestă lucrare sau pe contul git : **https://github.com/gabrielchmod777/virtualWorldVRML-serverJAVA.git**. Acesta se rulează din Eclipse.

7.3 Setarea unei surse locale ca depozit pentru plug-in-urile aplicației client

Sistemul demonstrativ are nevoie de un pachet LAMPP cu un server APACHE configurat, pentru a folosi adresa **http://localhost/plugins/** ca depozit pentru plug-in-urile aplicației client. Aplicația demonstrativă are nevoie de acces la această adresă pentru a funcționa.

Capitolul 8

Concluzii

8.1 Instalarea bibliotecilor software

- un rezumat al contribuțiilor aduse
- a analiză critică a rezultatelor obținute: avantaje, dezavantaje, limitări
- o descriere a posibilelor dezvoltări și îmbunătățiri ulterioare

8.2 Compilarea aplicației

8.2.1 Dimensiune

Cca 3–5% din total.

Anexa A

Testarea fiabilității modelului CLIENT-SERVER

A.1 Script

```
#!/bin/bash

# seconds_EPOCH.sh

while read line

do
    echo $(date +%s) ":" $line;
done
```

```
#!/bin/bash

# RUN_TESTS.sh

#!/bin/bash

telnetClients=10
rm ./*.out

echo "INCHIDEM SERVERUL ..."
kill -9 ‘pidof java‘
echo "START SERVER .... in 3 secunde "
java server &
sleep 3s
echo "OK"

for ((1; i<=$telnetClients; ++i )) ;
do
    fileName="./file_$i.out"
    telnet localhost 8080 | ./seconds_EPOCH.sh | tee -a $fileName &
done
```

A.2 Server de test

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.util.ArrayList;

class SClient extends Thread {

    private Socket              socket;
    private Integer             serverId;
    private BufferedReader      istream;
    private PrintStream         ostream;
    private boolean             keepRunning;
    private static ArrayList<SClient> allServers = new ArrayList<SClient>();

    @SuppressWarnings("unused")
        private SClient() {
        // blocat
    }


    public SClient(Socket socket, Integer id) throws IOException {

        this.socket = socket;
        this.serverId = id;
        this.istream = new BufferedReader(new InputStreamReader(this.socket.getInputStream()));
        this.ostream = new PrintStream(socket.getOutputStream());
        this.keepRunning = true;

        allServers.add( this );
    }

    public void sendResponse(String response) {
        this.ostream.println(response);
    }

    private String getRequest() throws IOException {
        String line = "";

        try{
            int c = 0;
            while(((char)c)!='\n'){
                c= this.istream.read();
                line = line+(char)c;
                if(c==-1){
                    this.keepRunning = false;
                    break;
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            this.keepRunning = false;
        }

        return line;
    }

    @Override
        public void run() {
        super.run();
```

```
        while(keepRunning) {

            String text = "";
            try {
                text = this.getRequest();
            } catch (IOException e) {
                e.printStackTrace();
                text = "EROARE";
            }

            if ( text.contains("quit") || text.contains("exit") )
                {
                    System.exit(0);
                }

            try{

                if ( allServers.size() >= 1 ) {

                    for(SClient user : allServers) {
                        if(user!=null) {
                            user.sendResponse(text);
                        }
                    }
                }

            } catch (Exception e){
                e.printStackTrace();
                this.sendResponse("\nEROARE");
            }
        }

        try {
            this.socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}

/*

              serverul de test

*/

public class server {

    public static void main(String[] args){

        /*
         *  in asteptarea clientilor
         */
        int serverId=1;
        ServerSocket serverSocket;
        try {
            serverSocket = new ServerSocket(8080);
            for(;;){
                Socket s = serverSocket.accept();
                System.out.print("\nNew␣server␣[id="+serverId+"]");
                /*
                 *  client nou
```

```
                 */
                Thread t = new SClient(s, serverId);
                serverId++;
                t.start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

A.2.1 Rezultate

La verificarea diferențelor de timp la primirea datelor de către programele client cu comanda **diff <fisier1> <fisier2>** . Testul este disponibil pentru verificare directorul "Teste/ApendixA" în arhiva GIT a lucrării. Descărcarea se poate face cu comanda **git clone https://github.com/gabrielchmod777/virtualWorldVRML-CLIENT.git**.

Testul este efectuat cu succes.

Anexa B

Îcărcarea dinamică a claselor în JAVA

```
#!/bin/bash

#
# check.sh -> assert în SHELL
#
typeset -i tests_run=0
function try { this="$1"; }
trap 'printf "$0: exit code $? on line $LINENO\nFAIL: $this\n"; exit 1' ERR
function assert {
        let tests_run+=1
        [ "$1" = "$2" ] && { echo -n "."; return; }
        printf "\nFAIL: $this\n'$1' != '$2'\n"; exit 1
}

try "test diff"
# ...
assert $1 $2

echo; echo "PASS: $tests_run tests run"
```

```
#!/bin/bash

#
# runTest.sh
#

echo 'Start test\n'
echo '-------------------------'
echo 'Testeaza -> Add 1 1'
v1=`java ApendixB Add 1 1`
./check.sh 2 $v1
echo '-------------------------'
echo 'Testeaza -> Add 5 1'
v1=`java ApendixB Add 5 1`
./check.sh 6 $v1
echo '-------------------------'
echo 'Testeaza -> Sub 1 1'
v1=`java ApendixB Sub 1 1`
./check.sh 0 $v1
```

```
echo '-------------------------'
echo 'Testeaza -> Sub 5 1'
v1=`java ApendixB Sub 5 1`
./check.sh 4 $v1
echo '-------------------------'
echo 'Testeaza -> Add 10 10'
v1=`java ApendixB Add 10 10`
./check.sh 20 $v1
echo '-------------------------'
echo 'Testeaza -> Sub 10 11'
v1=`java ApendixB Sub 10 11`
./check.sh -1 $v1
echo '-------------------------'
echo 'Testeaza -> Sub 19 3'
v1=`java ApendixB Sub 19 3`
./check.sh 16 $v1
echo '-------------------------'
echo 'Testeaza -> Add 101 10'
v1=`java ApendixB Add 101 10`
./check.sh 111 $v1
echo '-------------------------'
echo 'Testeaza -> Sub 0 3'
v1=`java ApendixB Sub 0 3`
./check.sh -3 $v1
echo '-------------------------'
echo 'Testeaza -> Sub 1 1'
v1=`java ApendixB Sub 1 1`
./check.sh 0 $v1
echo '-------------------------'
echo 'Stop test\n'
```

```
public interface Operatie {
void evaluate(Integer a, Integer b);
}
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

public class Add implements Operatie {
@Override
public void evaluate(Integer a, Integer b) {
Integer x = a + b;
System.out.print(Integer.toString(x));
}
}

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

public class Sub implements Operatie {
@Override
public void evaluate(Integer a, Integer b) {
Integer x = a - b;
System.out.print(Integer.toString(x));
}
}

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

public class ApendixB {

public static void main(String[] args) {

if ( args.length < 3 )
{
System.out.println("Eroare");
}

String classNameToBeLoaded = args[0];
Integer a = Integer.parseInt(args[1]);
Integer b = Integer.parseInt(args[2]);

ClassLoader myClassLoader = ApendixB.class.getClassLoader();
try {
Class<?> myClass = myClassLoader.loadClass(classNameToBeLoaded);
Object instance = myClass.newInstance();

if("Add".equals(classNameToBeLoaded)) {
((Add)instance).evaluate(a, b);
} else if("Sub".equals(classNameToBeLoaded)) {
((Sub)instance).evaluate(a, b);
} else  {
System.out.println("Eroare");
}


} catch (ClassNotFoundException | InstantiationException | IllegalAccessException e) {
System.out.println("Eroare");
}

}

}
```

```
Start test\n
-------------------------
Testeaza -> Add 1 1
.
PASS: 1 tests run
-------------------------
Testeaza -> Add 5 1
.
PASS: 1 tests run
-------------------------
Testeaza -> Sub 1 1
.
PASS: 1 tests run
-------------------------
Testeaza -> Sub 5 1
.
PASS: 1 tests run
-------------------------
Testeaza -> Add 10 10
.
PASS: 1 tests run
-------------------------
Testeaza -> Sub 10 11
.
PASS: 1 tests run
-------------------------
Testeaza -> Sub 19 3
.
PASS: 1 tests run
-------------------------
Testeaza -> Add 101 10
.
PASS: 1 tests run
-------------------------
Testeaza -> Sub 0 3
.
PASS: 1 tests run
-------------------------
Testeaza -> Sub 1 1
.
PASS: 1 tests run
-------------------------
Stop test\n
```

SERVER - Diagrama UML completă

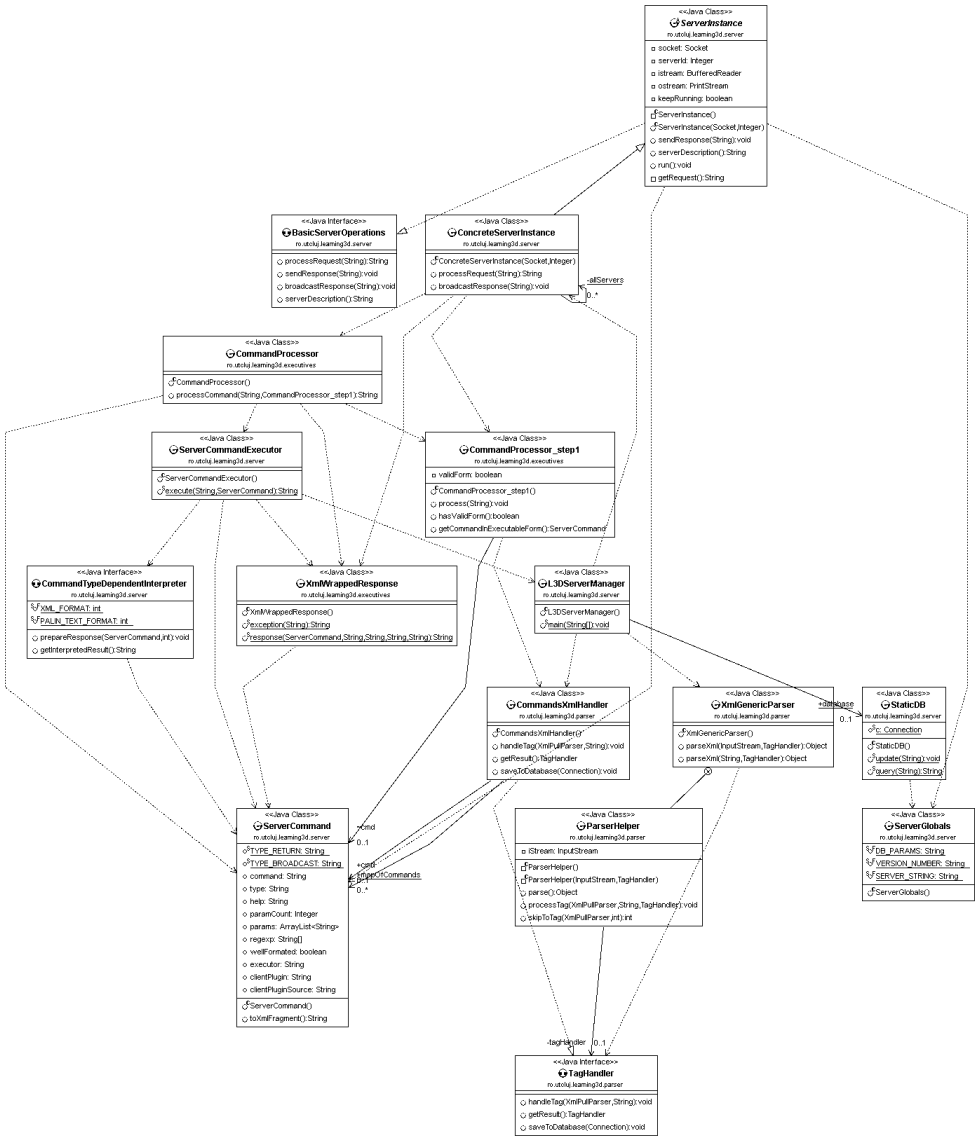

Figura B.1: Server - diagrama UML completă

Anexa C

Iconuri grafice utilizate pentru nodurile VRML în lucrarea ”The Inventor Mentor”

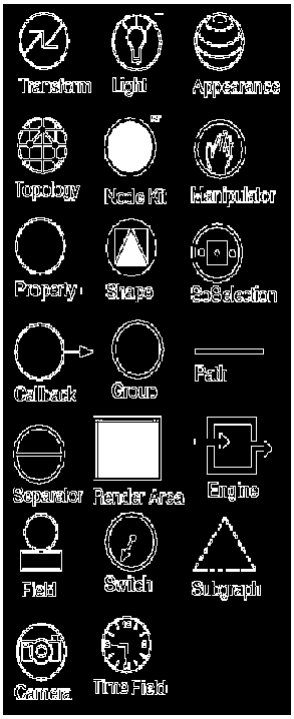

Figura C.1: Noduri VRML

Bibliografie

[1] *What does virtual reality NEED? Human factors issues in the design of three-dimensional computer environments.* International Journal of Human-Computer Studies, 1996.

[2] *Adult Education and Adult Learning.* Krieger Publishing Company, 2004.

[3] *Invatare, Intelegere, Uitare.* DTV, 1998.

[4] C. programatorilor Coin3D. (2013) Introducere. [Online]. Available: https://bitbucket.org/Coin3D/coin/wiki/IntroductionToCoin3D

[5] T. V. C. Incorporated, *The Virtual Reality Modeling Language*, 1997.

[6] "The inventor mentor:programming objectoriented 3d graphics with open inventor, release 2."

[7] "Introduction to vrml 97."

[8] *C++ GUI Programming with Qt 4.* Prentice Hall, 2008.