IMAGINAȚIE, CREATIVITATE, DESIGN, DEZVOLTARE

Lucrările
Sesiunii Naționale de Comunicări
Științifice Studențești ICDD 2010 Sibiu

Sibiu 2010

ISSN 2065-927X

IMAGINAȚIE, CREATIVITATE, DESIGN, DEZVOLTARE

Lucrările
Sesiunii Naționale de Comunicări Ştiințifice Studențeşti

Ediția a II-a
Sibiu, 23 – 24 Aprilie, 2010

UNIVERSITATEA "LUCIAN BLAGA" SIBIU, FACULTATEA DE ŞTIINȚE
CATEDRA DE INFORMATICĂ

Lucrările Sesiunii Naționale de Comunicări Ştiințifice Studențeşti
"Imaginație, Creativitate, Design, Dezvoltare"
Ediția a II-a, Sibiu, România

Editor *Prof. Univ. Dr. Dana Simian*

Colectiv tehnoredactare
Prof. Univ. Dr. *Dana Simian*
Prep. Univ. *Laura Cacovean*

Design copertă
Stud. *Nechifor Vasile Nicuşor*

ISSN 2065 – 927X

Motto:

"There are no limits, only your imagination"

Prefață

Prezentul volum reunește lucrările prezentate la a doua ediție a Sesiunii Naționale de Comunicări Științifice Studențești "Imaginație, Creativitate, Design, Dezvoltare", desfășurată la Universitatea „Lucian Blaga" din Sibiu, în perioada 23-24 Aprilie 2010, organizată de către Catedra de Informatică din cadrul Facultății de Științe.

Scopul sesiunii de comunicări este de a reuni studenți din toate centrele universitare din țară pentru a prezenta și a discuta rezultate originale obținute în toate ariile tematice ale domeniului informatică: informatică teoretică, algoritmică, proiectarea și construirea de software, trimiterea datelor prin rețele, noi abordări în probleme de securitate, etc.

Sesiunea științifică include și o secțiune specială dedicată elevilor de liceu.

Mulțumim tuturor participanților, colectivului de organizare și colectivului științific, pentru contribuția adusă la succesul acestei manifestări științifice și la realizarea prezentului volum.

Prof. Univ. Dr. Dana Simian

CUPRINS

Partea I - Secțiunea dedicată STUDENȚILOR

Partea a II-a - Secțiunea dedicată ELEVILOR

PARTEA 1

Secţiunea dedicată studenţilor

Sesiunea Națională de Comunicări Științifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

SIE-Search Inject Exploit

Alexe Iulian, Morariu Stelian, Doda Gheorghe
Coordonator: Lector univ. Dr. Mircea Iosif Neamţu

Abstract

Even though web programming level is reasonably advanced, even in Romania, there are still a lot of web sites that incorporate security vulnerabilities in the database layer of the web site. A technique that exploits these kinds of vulnerabilities is called SQL Injection and is the subject of this paper.

A successful attack using a SQL Injection could lead to the access and modification of the web site's database, which translates into extraction and modification of : ID's, passwords, e-mail addresses, home addresses, phone numbers-basically everything that is stored and used by the web site.

In this state of mind, we developed the application named SIE (Search Inject Exploit) to provide web programmers with a tool that searches vulnerable links of their web site (i.e. a product page), attempts to perform a SQL Injection and finally try to find the admin login page and *crack* the MD5 hashed password (inappropriately called "crack" because we are actually using Rainbow tables). The application is structured on three tabs, each corresponding to the actions performed by the application and, if you take into consideration that this is a project in development, it could be considered an All-In-One database security testing utility.

1 Introducere

Sql injection este una dintre cele mai comune vulnerabilitaţi ale aplicaţiilor de net şi totodată una din cele mai fatale metode de a „exploata" un site, putând afla informaţii confidenţiale din baza de date prin intermediul interogărilor SQL.

Această vulnerabilitate este fatala mai ales azi când informaţia pe web a evoluat atât de mult şi majoritatea informaţiilor despre persoane, conturi bancare, cumpărături online, etc. sunt ţinute în baze de date pe servere,de aceea securitatea trebuie să fie maximă.

Atacurile SQL reprezintă o ameninţare serioasă asupra oricarui site care are implementă o bază de date. În ciuda acestor riscuri un număr foarte mare de site-uri sunt susceptibile unei astfel de forme de atac.

2 Descriere Program

2.1 Metode de gasire a Vulnerabilitaţii SQL Injection

Pentru un link: *http://www.numesite.ro/fisier.php?id=1* linia de cod caracteristică php-ului poate fi*: $sql = "SELECT * FROM users WHERE id=".$_GET['id'];*

Variabila id este preluată cu GET din URL şi afişată.

Pentru a vedea dacă link-ul este vulnerabil se adaugă un apostrof dupa secventa "id=1", adică : *http://www.numesite.ro/fisier.php?id=1'*

Apare o eroare de forma:

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line

Aici intervine programul nostru în descoperirea acesui tip de vulnerabilitate şi verificare securitaţii site-ului.

Metoda de testare a posibilităţii de extragere a datelor din baza de date este prin interogarea bazei de date "*information_schema"* care se creează automat atunci când se creează un user nou cu acces la baza de date. Această baza de date conţine informaţii cu privire la toate celelalte baze de date ale unui user şi date generale despre utilizator. Aplicaţia foloseşte funcţii în care sunt declarate interogării SQL speciale pentru testarea accesului la date.

2.2 Metode de evitare a vulnerabilităţi SQL Injection

- setarea obţiunii *magic_quates* în Php pe enable;
- folosirea funcţiei php *mysql_real_escape_string( )*;
- folosirea funcţiilor php *is_numeric( )* si *ctype_digit( )*;
- folosirea fisierelor apache *.htaccess*.

3 Codul sursă a programului

Numele funcţiilor principale folosite şi descrierea lor:

```
private void ListDB()  // listază bazele de date pe care le conţine site-ul vulnerabil;

private void ListTables()  // listează tabelele din bazele de date;

ListColumns()  //listează coloanele din tablele existente;

private void GetData()  // preia datele din coloanele site-ului;

private string ConvertToHex(string asciiString)  // converteşte URL-ul în Hex;

public void DBTreeNode(string adddb)  // organizează structura bazei de date;
```

private void ShowStatus() // afisează acţiunea curentă.

4 Concluzii

Pe viitor o să dezvoltăm aplicatia să suporte :

- testarea bazelor de date MsSQL, Oracle, MsAccess, Mysql Blind;
- citirea de fişiere de pe server;
- scrierea de fişiere pe server;
- executarea de comenzi(pentru serverele Windows);
- introducerea manuala a sintaxei de injectare.

Bibliografie

[1] http://www.c-sharpcorner.com/.
[2] http://pentestmonkey.net/blog/mysql-sql-injection-cheat-sheet/
[3] http://www.milw0rm.com/

Alexe Iulian
Universitatea Lucian Blaga
Facultatea de Stiinte
Informatica
Str. Dr. Ioan Raiu, nr. 5-7, Sibiu
ROMÂNIA
E-mail: iulian.alexe@ulbsibiu.ro

Morariu Stelian
Universitatea Lucian Blaga
Facultatea de Stiinte
Informatica
Str. Dr. Ioan Raiu, nr. 5-7, Sibiu
ROMÂNIA
E-mail: modev_st@yahoo.com

Doda Gheorghe
Universitatea Lucian Blaga
Facultatea de Stiinte
Informatica
Str. Dr. Ioan Raiu, nr. 5-7, Sibiu
ROMÂNIA
E-mail: godlike_affairs@yahoo.com

Sesiunea Naţională de Comunicări Ştiinţifice a Studenţilor
"Imaginaţie, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Active Shape and Appearance Models

Adriana Ban, Diana Cristea
lect. univ. drd. Ralf Fabian

Rezumat

Statistical models of shape and appearance are powerful tools for interpreting medical images. We assume a training set of images in which corresponding 'landmark' points have been marked on every image. From this data we can compute a a statistical model of the shape variation, a model of the texture variation and a model of the correlations between shape and texture. With enough training examples such models should be able to synthesize any image of normal anatomy. By finding the parameters which optimize the match between a synthesized model image and a target image we can locate all the structures represented by the model. Two approaches to the matching will be described. The Active Shape Model essentially matches a model to boundaries in an image. The Active Appearance Model finds model parameters which synthesize a complete image which is as similar as possible to the target image. By using a 'difference decomposition' approach the current difference between target image and synthesized model image can be used to update the model parameters, leading to rapid matching of complex models. We will demonstrate the application of such models to a variety of different problems.
Keywords: Shape Models, Appearance Models, Model Matching

1 Introducere

Multe probleme în interpretarea imaginilor medicale implică necesitatea unui sistem automatizat care să înţeleagă imaginile ce îi sunt prezentate, aceasta presupunând ca sistemul respectiv să fie capabil să descopere structura imaginii şi să o înţeleagă. Acest lucru implică în mod necesar utilizarea unor modele care descriu şi etichetează anumite structuri. Aplicaţiile reale sunt caracterizate prin nevoia de a face faţă interpretării structurilor complexe şi imaginilor incomplete. De asemenea, este adesea imposibil să interpretezi o imagine fără a avea în prealabil cunoştiinţe de anatomie.

Metodele bazate pe modele oferă potenţiale soluţii la toate aceste probleme. Cunoştiinţe anterioare legate de acestea pot fi folosite, în principiu, la rezolvarea unor potenţiale confuzii cauzate de complexitatea structurală, de lipsa unor date, şi să ofere un mijloc de reîntregire a structurilor. Se doreşte aplicarea cunoştiinţelor referitoare la formele structurilor, relaţiile lor spaţiale pentru a preveni interpretările greşite ale sistemului. De interes special, sunt modelele de tip generator, care sunt suficient de complete pentru a fi în masură să genereze imagini realiste ale obiectelor ţintă. Un exemplu ar fi un model "faţă" capabil să genereze imagini convingătoare ale oricărei persoane, schimbându-le expresia, ş.a.m.d. Folosind un astfel de model, interpretarea imaginii poate fi formulată ca o problemă de potrivire: fiind dată o imagine pentru a fi interpretată, structurile pot fi localizate şi etichetate prin ajustarea parametrilor modelului într-un mod în care se generează o imagine "imaginată" care este cât mai apropiată de realitate.
Deoarece aplicaţiile reale implică adesea lucrul cu clase de obiecte care nu sunt identice, trebuie să ne confruntăm cu variabilitatea. Acest lucru ne duce în mod natural la ideea de modele deformabile - modele care menţin caracteristicile esenţiale ale clasei de obiecte pe care le reprezintă, dar care se pot

deforma pentru a se potrivi unei serii de exemple. Există două caracteristici importante pe care am dori ca aceste modele să le posede. În primul rând, ar trebui să fie generale, adică să fie capabile să genereze orice exemplu plauzibil din clasa pe care o reprezintă. În al doilea rând, şi cel mai important, ar trebui să fie specifice, adică ar trebui să fie capabile să genereze doar modele plauzibile, deoarece scopul este obţinerea unei interpretări plauzibile. Pentru a obţine anumite modele de obiecte diferite trebuie să ştim cum variază acestea.
O abordare interesantă este de a instrui sistemul folosind un set de imagini corespunzătoare. Vom descrie mai jos cum modele statistice pot fi construite pentru a reprezenta atât forma cât şi "textura" (modelul intensităţi pixelilor) exemplelor de structuri de interes. Aceste modele pot generaliza folosindu-se de setul de imagini de instruire şi pot fi utilizate pentru a se potrivi unor noi imagini, localizând structura în imagini. Două abordări vor fi prezentate. Primul model, modelul Active Shape (ASM), se concentrează pe potrivirea unui model pe o imagine, de obicei, potrivirea unui contur pe marginile structurii ţintă. Al doilea model, modelul Active Appearance (AAM), încearcă să sintetizeze apariţia completă a imaginii ţintă, alegând parametri care să reducă diferenţa dintre imaginea ţintă şi imaginea generată de model. Ambii algoritmi s-au dovedit a fi rapizi, exacţi şi fiabili.

2 Cadru general

Variabilitatea inter- şi intra-personală a structurilor biologice face ca interpretarea imaginilor medicale să fie o sarcină destul de dificilă. În ultimii ani s-a dezvoltat un interes considerabil pentru metodele care folosesc modele deformabile pentru a interpreta imagini. Una dintre motivaţii este aceea de a obţine performanţe prin utilizarea modelului pentru a constrânge soluţiile până la a fi exemple valide ale structurilor modelate. De o importanţă fundamentală este faptul că, odată ce s-a găsit un model care să corespundă unei imagini a unui pacient, etichete anatomice şi valori de intensitate pot fi transferate direct. Aceasta constituie o bază pentru interpretarea anatomică automată şi pentru fuziunea datelor în imagini diferite alee aceleiaşi persoane sau pe imagini similare ale unor persoane diferite. Pentru o analiză corespunzătoare a actrivităţii în acest domeniu există studii recente asupra modelelor deformabile în analiza imaginilor medicale.

3 Modele Active Shape (ASM)

Fiind dată o aproximare nu foarte bună a unui model, acesta din urmă se poate potrivi pe o imagine. Alegând un set de parametri de contur, $\mathbf{b}$, definim pentru un model conturul acestuia într-un cadru de coordonate centrat în obiect. Putem crea o instanţă $\mathbf{X}$ a modelului prin definirea poziţiei, orientării şi scalei. O abordare iterativă pentru o potrivire cât mai bună este după cum urmează:

1. Se examinează o regiune a imaginii în jurul fiecărui punct X_i pentru a găsi cea mai bună potrivire pentru punctul X_i'.
2. Se updatează parametrii (t, b) pentru a se potrivi mai bine noilor puncte X găsite.
3. Se repetă până când se ajunge la o potrivire cât mai exactă.

În practică se doreşte crearea unor modele cu un grad cât mai ridicat de folosinţă. Dacă, spre exemplu, ne gândim ca respectivul contur al modelului să corespundă, putem pur şi simplu să localizăm adevărata margine a structurii (incluzând orientarea, dacă aceasta se cunoaşte).

Cu toate acestea, punctele modelului nu se află întotdeauna în apropierea marginii structurii - acestea pot reprezenta o margine secundară sau o altă structură diferită. Cea mai bună abordare este ca sistemul creat să ştie ce să caute în imaginea respectivă. Aceasta se face creând mostre în limitele profilului normal din setul de date de instruire şi construind un model statistic.

3.1 Modelarea locală a structurii

Să presupunem că pentru un punct dat creem k pixeli de ambele părţi ale punctului din model în a i-a imagine din setul de instruire. Avem $2k+1$ mostre care pot fi ordonate într-un vector g_i. Pentru a reduce

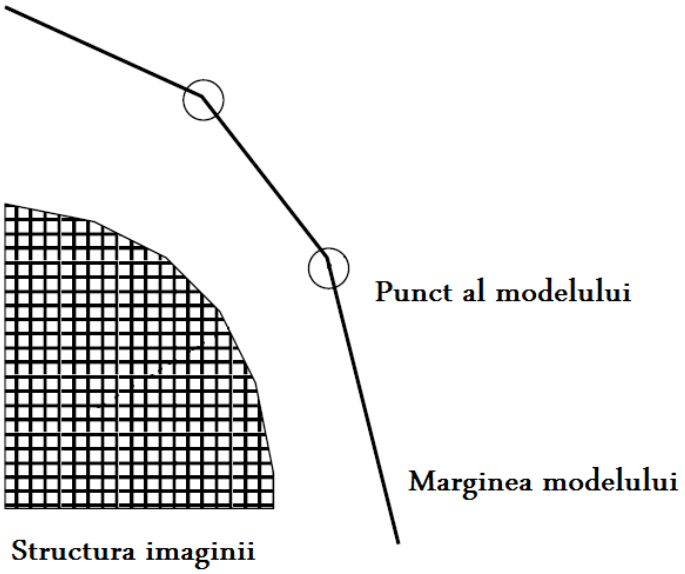


Figura 1: Pentru fiecare punct al modelului creăm mostre ale marginii

efectele schimbării intensităţii globale, creăm mostre ale derivatei de-a lungul profilului. Apoi aducem la normalitate mostra, divizând prin suma valorilor elementelor absolute,

$$g_i \rightarrow \frac{1}{\sum_j \|g_{ij}\|} g_i$$

Repetăm procesul pentru fiecare imagine din setul de instruire pentru a obţine un set de mostre "normale" g_i pentru punctul modelului dat. În timpul căutării, creăm de-a lungul profilului, m pixeli de fiecare parte a punctului respectiv ($m > k$). Testăm apoi calitatea de potrivire cu modelul de nivel gri la fiecare dintre cele $2(m-k)+1$ poziţii posibile de-a lungul mostrei şi alegem acela care se potriveşte cel mai bine.

Aceasta se repetă pentru fiecare punct al modelului, sugerând o nouă poziţie pentru fiecare punct. Apoi intersectăm poziţia şi forma parametrilor care potrivesc cel mai bine modelul la noile puncte, impunând constrângeri asupra poziţiilor punctelor.

3.2 Exemple

Figura 2 demonstrează cum ASM localizează trăsăturile feţei. Modelul este plasat aproape de centrul imaginii şi realizează o căutare exactă. Se realizează paşi mari în primele câteva iteraţii, modelul apropiindu-se destul de mult de forma finală. Convergenţa finală (după un total de 18 iteraţii) oferă o oarecare potrivire cu imaginea ţintă. În acest caz, cel mult 5 iteraţii fiind admise la fiecare rezoluţie, iar algoritmul converge în mai puţin de o secundă (pe un calculator modern).

Figura 3 demonstrează cum ASM poate eşua dacă poziţia iniţială este prea departe de ţintă. Deoarece se caută doar de-a lungul profilelor în jurul poziţiei curente, nu se poate corecta pe distanţe mari faţă de poziţia curentă. Există două posibilităţi:

1. va diverge spre infinit sau

2. va converge la o soluţie incorectă, realizând cea mai bună potrivire pe imaginea locală

(a) Iniţial (b) După 2 iteraţii (c) După 6 iteraţii (d) După 18 iteraţii

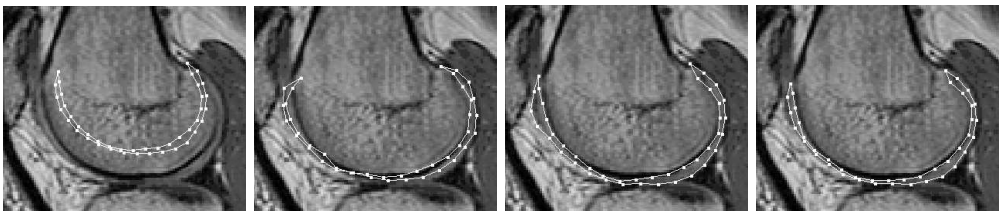
Figura 2: Căutare folosind ASM

(a) Iniţial (b) După 2 iteraţii (c) După 20 de iteraţii

Figura 3: ASM poate eşua în găsirea unui rezultat acceptabil atunci când modelul nu este poziţionat în apropierea structurii

În exemplul arătat, s-a putut localiza doar jumătate din faţă, cealaltă jumătate fiind prea departe.

Figura 4 ne arată utilizarea ASM pe un cartilaj pentru a localiza structura într-o imagine nouă.



(a) Iniţial (b) După o iteraţie (c) După 6 iteraţii (d) După 14 iteraţii

Figura 4: Căutarea unui cartilaj folosind ASM

4 Modele Active Appearance (AAM)

În această secţiune prezentăm algoritmul AAM. AAM este format din 2 componente: un model parametrizat al obiectului de studiat şi o estimare a relaţiei dintre parametrul erorilor şi imaginea reziduală indusă.

Parametrii modelului apărut, notaţi cu c, şi parametrul transformării conturului, notat cu t, definesc poziţia unui punct al modelului în imaginea cadru, X, care ne dă conturul imaginii pe care vrem să o reprezentăm ca model. Pentru o potrivire mai bună a imaginii, eşantionăm pixeli în regiunea imaginii,

g_{im}, şi a proiectului în textura modelului cadru, $g_s = T^{-1}g(_{im})$. Textura modelului curent este dată de $g_m = \overline{g} + P_g Q_g c$.

Diferenţa între model şi imagine este dată de

$$r(p) = g_s - g_m \tag{1}$$

unde p este parametrul modelului, $p^T = (c^T|t^T|u^T)$. Un simplu scalar realizează diferenţa de măsură care este suma pătratelor elementelor r, $E(p) = r^T r$.

O primă ordine pentru extinderea Taylor (1) ne dă:

$$r(p + \partial p) = r(p) + \frac{\partial r}{\partial p}\delta p \tag{2}$$

unde elementul ij al matricii $\dfrac{\partial r}{\partial p}$ este $\dfrac{dr_i}{dr_j}$.

Să presupunem că timpul de potrivire rezidual este r. Trebuie să alegem $\pm p$ astfel încât să minimizăm $|r(p + \delta p)|^2$. Egalând relaţia (2) cu 0 vom obţine o relaţie RSM:

$$\delta p = -Rr(p) \text{ unde } R = (\frac{\partial r}{\partial p}^T \frac{\partial r}{\partial p})^{-1} \frac{\partial r}{\partial p}^T \tag{3}$$

Într-o schemă de optimizare standard ar fi necesar să recalculăm $\frac{\partial r}{\partial p}$ la fiecare pas, o operaţie costisitoare. Oricum, presupunem că, întrucât este calculat într-un cadru de referinţa normalizat, acesta poate fi considerat aproximativ fix. Estimăm $\frac{\partial r}{\partial p}$ de diferenţială numerică, sistematic, deplasând fiecare parametru de la valoarea optimă cunoscută pe imagini tipice şi calculate pe o medie de formare stabilite. Reziduale, la deplasări, de diferite mărimi sunt măsurate (de obicei, pâna la 0.5 deviaţii standard pentru fiecare parametru) şi combinate cu un nucleu Gaussian pentru a le netezi.

Dupa ce recompunem R îl putem utiliza în toate cautârile ulterioare folosind modelul.

Folosind ecuaţia (3) putem sugera o corecţie pentru a face în modelul parametric bazat pe o măsură reziduală r. Acest lucru ne permite să construim un algoritm iterativ pentru a optimiza problema noastră. Fiind dată o estimare curentă a modelului parametric, c, poziţia t, textura transformării u, şi imaginea simplă a estimării, g_{im}, un pas în algoritmul iterativ este următorul:

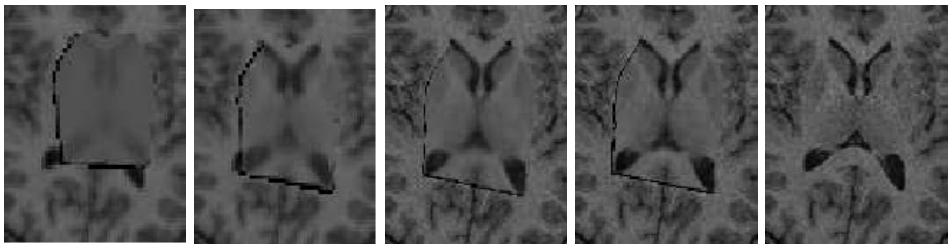
1. Proiectăm textura simplă în textura modelului cadru folosind $g_s = T_u^{-1}(g_{im})$;
2. Evaluăm vectorul de eroare, $r = g_s - g_m$, şi eroarea curentă $E = |r|^2$;
3. Calculăm deplasările $\partial p = -Rr(p)$;
4. Actualizăm modelul parametric $p \rightarrow p + k\partial p$, unde iniţial $k = 1$.
5. Calculăm noile puncte, $X^{'}$ şi textura modelului cadru $g_m^{'}$;
6. Eşantionăm imaginea la noi puncte pentru a obţine $g_{im}^{'}$;
7. Calculăm un nou vector al erorilor $r^{'} = T_{u'}^{-1}(g_{im}^{'}) - g_m^{'}$;
8. Dacă $|r^{'}|^2 < E$ acceptăm noua estimare, altfel încercăm cu $k = 0.5$, $k = 0.25$.

Această procedură se repetă până când nu are loc nicio îmbunătăţire asupra erorii, $|r|^2$, şi convergenţa este găsită. În practică se foloseşte o implementare a multor rezoluţii , în care pornim de la o rezoluţie brută şi iterăm pentru a converge la fiecare nivel înainte de a proiecta soluţia găsită la fiecare nivel al modelului. Acest mod este mai eficient şi putem converge la soluţia corectă.

4.1 Exemple de căutare AAM

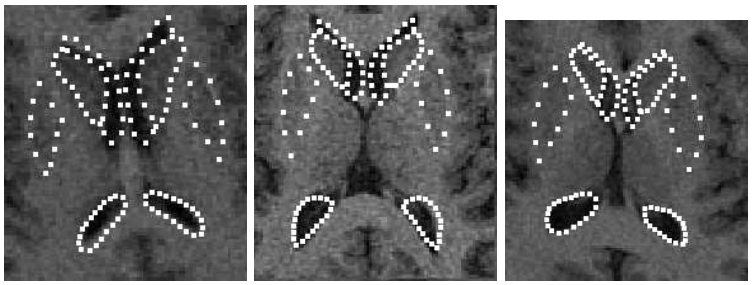
De exemplu, în figura 5, avem un exemplu de AAM al unei structuri centrale al creierului provenită dintr-o poziţie deplasată pe o imagine nevăzută anterior. Modelul poate fi reprezentat pe 10000 pixeli având 30 de c parametri. Această căutare durează o secundă pe un calculator modern. Figura 6 prezintă rezultatele unei căutări, cu punctele modelului găsit suprapuse pe imagini ţintă. Deşi am demonstrat doar în partea centrală a creierului, modelele pot fi construite în întreaga secţiune transversală.

Figura 4.1 arată primele două moduri unui astfel de model.



(a) Iniţial (b) 2 iteraţii (c) 6 iteraţii (d) 16 iteraţii (e) Original

Figura 5: Căutare AAM multi-rezoluţie



(a) (b) (c)

Figura 6: Rezultate ale căutării AAM

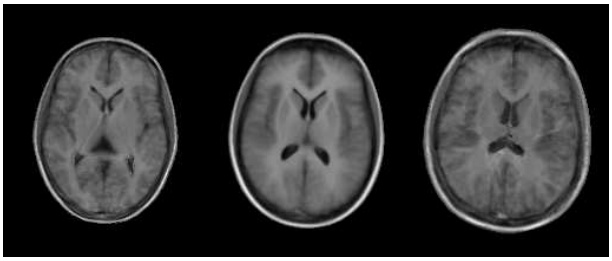
Modelul aspect se bazează pe existenţa de corespondenţei între structuri în imagini diferite, şi, astfel, pe o topologie coerentă în exemple.

Există situaţii în care acest algoritm eşuează după cum vom vedea din următorul exemplu.

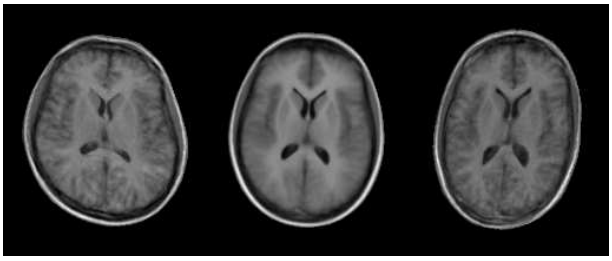
Figura 8 prezintă două exemple în cazul în care AAM nu a reuşit să localizeze corect limitele pe imagini nevăzute. În ambele cazuri, exemplele demonstrează forma extremă variată mai mult de la medie, şi este limita exterioară a faptului ca modelul nu poate localiza. Acest lucru se datorează faptului că probele modelului nu se găsesc în locaţia actuală. Acolo nu sunt întotdeauna suficient de multe informaţii pentru a conduce modelul spre exterior, pâna la frontiera exterioara corectă.

5 Discuţii şi concluzii

Am demonstrat că structurile de imagine pot fi reprezentate folosind modele statistice ASM şi AAM. Atât ASM cât şi AAM pot varia în moduri observate în formare stabilind deformări arbitrare care nu sunt permise. Potrivirile la o nouă imagine pot fi realizate rapid, fie folosind Active Shape Model fie Active Apperance Model cu algoritmi de model.

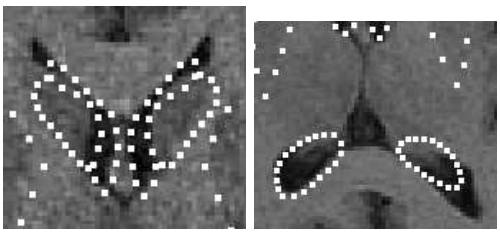


(a) c1 variază cu ± 2 s.d.s



(b) c2 variază cu ± 2 s.d.s

Figura 7: Primele două moduri ale AAM asupra unei secţiuni a creierului



(a) (b)

Figura 8: AAM nu găseşte întotdeauna marginile corecte ale ventriculelor (vezi text)

5.1 Aplicaţii ale AAM şi ASM

ASM a fost folosit pentru a localiza vertebre în imagini ale coloanei vertebrale ,oase şi proteze în radiografii de înlocuitori de şold, structuri în imagini MR a creierului, şi contururile ventriculelor în echocardiograme. Ambele au fost utilizate în interpretarea feţelor. Abordări pot fi extinse la 3D, şi au fost folosite pentru a interpreta volumul unor imagini.

5.2 Comparaţii între ASM şi AAM

ASM caută în jurul locaţiei curente, de-a lungul profilelor, astfel încât tind să aibă o gamă de captare mai mare decât AAM care examinează numai imaginea direct sub zona sa curentă.

ASM utilizează numai datele din jurul punctelor de model, şi nu profită de toate informaţiile la nivel de gri disponibile în cadrul unui obiect aşa cum procedează AAM. Cu toate acestea, punctele de pe model tind să fie locuri de interes ,în cazul în care există cele mai multe informaţii. Dacă de exemplu s-ar pregăti un AAM să caute numai folosind informaţii din zonele din apropierea graniţelor puternic limitate - dar acest lucru ar necesita o imagine având mai puţine probe prelevate în timpul de căutare, astfel un algoritm mai rapid din punct de vedere al potenţialităţii.

Un avantaj al AAM este acela că se poate construi un model convingător cu un număr relativ mic de

repere. ASM are nevoie de puncte, astfel încât să definească direcţiile potrivite pentru căutare. Pentru a obţine imagini de etichetare fiabile este nevoie de cât mai puţine repere necesare pentru a obţine rezultate cât mai bune.

În general, am constatat că ASM este mai rapidă şi mai precisă realizând locaţii punctate mai caracterstice decât AAM. Cu toate acestea, deoarece minimizează în mod explicit erori, textura AAM dă rezultate mai bune la textura imaginii.

5.3 Extinderi ale ASM şi AAM la 3-D

Abordări ale ASM şi AAM au fost demonstrate în 2-D, în 3D ele fiind extensibile. Complicaţiile principale sunt mărimea modelelor şi dificultatea de a obţine rezultate bune. Obţinerea unor corespondenţe în imagini 3D este dificilă dar este obiectul cercetării actuale. Extinderea ASM la 3D este relativ simplă, având în vedere un set adecvat de imagini adnotate. Profile modelate şi eşantion sunt pur şi simplu luate de-a lungul liniilor prin imagini 3D ortogonale locale pe o suprafată.

În teorie extinderea AAM este simplă, dar în practică modelele ar fi extrem de mari. Fiecare mod al aspectului modelului este de mărimea unui complet 3D de imagine, dar vor fi necesare multe moduri pentru a reprezenta imaginea. O abordare mai practică este probabil să fie doar de prelevare de probe în benzi în jurul graniţelor de interes.

Abordări pot fi, de asemenea, extinse în domeniul temporal, pentru a urmări obiecte prin intermediul secvenţelor, de exemplu, graniţa inimii în echocardiograme.

5.4 Concluzii

Am arătat cum modele statistice de aspect pot fi reprezentate în medie în moduri de variaţie de forma şi textura structurilor care apar în imagini. Astfel de modele pot fi adaptate la noi imagini rapid şi fiabil, fie folosind algoritmi ASM sau AAM. Metodele sunt aplicabile pentru o mare varietate de probleme şi oferă un cadru util pentru interpretarea automată a imaginii.

Bibliografie

[1] T. F. Cootes, C. J. Taylor, *Statistical models of appearance for medical image analysis and computer vision.*

[2] T. F. Cootes, C. J. Taylor, D. H. Cooper, J. Graham, *Active Shape Models - Their Training and Application*

[3] T. F. Cootes, C. J. Taylor, *Statistical Models of Appearance for Computer Vision*

Ban Adriana
Universitatea "Lucian Blaga" Sibiu
Facultatea de Ştiinţe
Str. Dr. Ioan Raţiu nr. 5-7
ROMÂNIA
E-mail: *ban.adriana@yahoo.com*

Cristea Diana
Universitatea "Lucian Blaga" Sibiu
Facultatea de Ştiinţe
Str. Dr. Ioan Raţiu nr. 5-7
ROMÂNIA
E-mail: *diana.cristea25@yahoo.com*

Sesiunea Naţională de Comunicări Ştiinţifice a Studenţilor
"Imaginaţie, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Tehnici de planificare practică a drumurilor pentru caractere cu mişcare autonomă din jocuri de strategie

Bota Florentin
Coordonator: Lector univ. drd. Ralf Fabian

Abstract

The purpose of this project is to review and develop different pathfinding algorithms used mainly in computer games but also in some real-world applications. In this study, different programming languages were used, in order to display different examples correctly, but the most important parts were written in Java environment, using the free IDE Eclipse. Once again Java proves itself to be a real asset in the creation of simple examples but fully equiped to solve more complex operations.

The last section contains an implementation of the A* algorithm on a personal RTS game, currently in an alpha stage, written for the better understanding of computer graphics and Artificial Intelligence.

1 Introducere şi descriere

Planificarea drumurilor sau **pathfinding** se referă la găsirea drumului optim dintre punctele A şi B de către o aplicaţie informatică şi constă în metode mai avansate de rezolvare a problemelor de tip labirint.

Din punctul de vedere al dezvoltării de jocuri, pathfinding-ul se ocupa de felul in care o entitate cu mişcare autonoma găseşte o parcurgere, ocolind obstacole. Dupa cum vom observa în urmatoarele secţiuni, nu este necesar ca acel drum să fie cel mai scurt dacă asta ar incetini prea mult sistemul de calcul pe care rulează aplicaţia. Un rezultat apropiat de cel optim este de preferat situaţiei în care un numar mare de unităţi(*sprites*) ar utiliza toate resursele atribuite jocului doar pentru a calcula mişcările pe care urmează sa le facă.

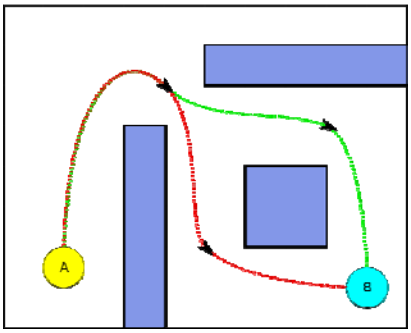


Fig. 1: Rute echivalente in ocolirea unui obstacol

Din acelaşi motiv, cei mai buni algoritmi se bazează pe calcule euristice, obţinând un plus de viteză într-un mediu ce tinde spre un număr de FPS (Frames per Second) cât mai mare.

Acesti algoritmi sunt utilizaţi in mod frecvent în jocuri de strategie în timp real (RTS), în care jucătorul direcţionează caracterele într-o zona virtuală extinsa ce conţine diverse obstacole, însă sunt întalniţi în majoritatea tipurilor de jocuri. Importanţa aplicarii corecte a acestora a crescut proporţional cu complexitatea lumilor virtuale în care se desfaşoară acţiunea.

Pe langă industria jocurilor, algoritmii de pathfinding işi găsesc utilitatea în cele mai diverse domenii, de la transmiterea informaţiilor prin Internet, la sistemele GPS si chiar la programarea noilor generaţii de roboţi.

Ideea de bază a planificarii drumurilor în jocuri este folosirea reprezentării terenului sub formă de noduri, ce pot fi aşa numitele ***tiles (***dale de teren***)*** din care e formată de obicei lumea într-un RTS, sau ***waypoints(***puncte de directie***)*** ce sunt adăugate manual şi folosite in principal in shootere FPS (First Person Shooter) dar şi în alte jocuri 3D.

Într-un joc modern, indiferent de tipul său, simpla găsire a celui mai rapid drum nu este suficientă, intrucât de obicei se pune problema mai multor entitaţi ce se mişcă în acelaşi timp pe hartă, aşadar e necesară implementarea unor algoritmi specializaţi pentru evitarea conglomerării pe puncte de interes major(*fig. 2.1*) sau chiar pe coliziunea dintre unităţi.

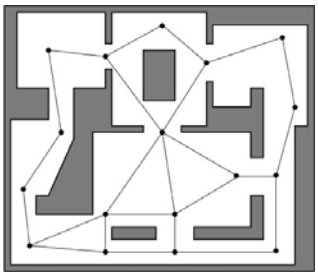


Fig. 2.1: Exemplu de waypoints pe o hartă poligonală
Sursa: Mat Buckland, Programming Game AI by Example

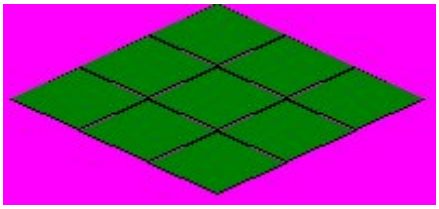


Fig. 2.2: Exemplu de hartă bazată pe celule
Sursa: DelphyX Game Tutorial One, Michael Dundee - http://edn.embarcadero.com/article/22928

2 Interpretare

2.1 Tipuri de creare a grafului in funcţie de tipul hărţii

Deoarece lumea virtuală în care se află agenţii autonomi poate fi de mai multe tipuri, şi metodele de gândire şi reprezentare a grafului de drumuri pot fi dintre cele mai diverse. În funcţie de dimensiunea hărţii, tipul de grafică folosită (2D/3D) dar şi de tipul jocului, conexiunea dintre nodurile grafului şi reprezentarea vizuală variază de la o aplicaţie la alta.

2.1.1 Lumea bazată pe *celule* (tiles)

Jocurile bazate pe celule(tiles) aşa cum sunt RTS-urile şi seriile de jocuri-război au de obicei medii virtuale imense, complexe, bazate pe pătrate sau hexagoane. Aşadar, pare normal construirea grafului navigaţional în jurul acestora. Fiecare nod reprezintă centrul celulei, iar marginile acesteia vor conţine nodurile de legatură dintre celulele alăturate. Uneori, un cost poate fi atribuit in funcţie de tipul celulei, astfel mişcarea printr-o mlastină fiind mai greoaie decât pe un drum, spre exemplu, sau ocolirea unui deal poate fi mai rapidă decât urcarea lui etc.

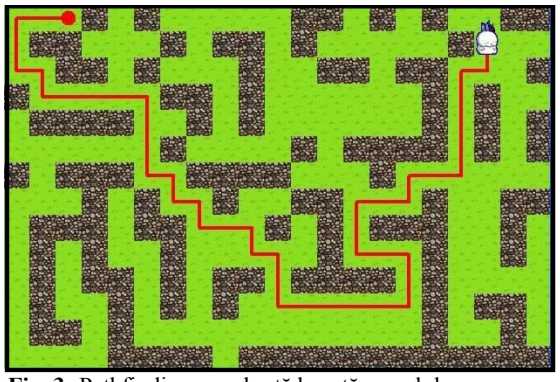


Fig. 3: Pathfinding pe o hartă bazată pe celule
Sursa: Cognition for Machines- http://www.cognaxon.com/index.php?page=educational

Dezavantajul acestei metode este insă numarul mare de noduri, deoarece chiar si o hartă de 100x100 celule are nevoie de 10000 de noduri si ~78000 de muchii. Adăugând la acestea si un numar mare de unităţi, găsirea drumului optim devine o problemă mai complexă.

2.1.2 Puncte de vizibilitate

Un joc bazat pe puncte de vizibilitate(POV-fig. 2) este creat, de obicei, prin adaugarea manuală a unor puncte de interes astfel incât fiecare nod al grafului are viziune directă cel putin asupra altui nod. Poziţionate cu grijă, nodurile grafului pot conecta toate zonele importante ale geometriei hărţii. Avantajul acestei metode este că geometria grafului poate fi uşor modificată în funcţie de diferitele cerinţe ale evenimentelor ce vor avea loc în locuri prestabilite (ambuscadă, pândă, atac etc), iar daca lumea este construită din poligoane, acestea pot fi folosite pentru a crea automat punctele de vizibilitate (fig. 4).

Dezavantajul major al acestei metode este faptul că în cazul hărţilor de dimensiuni mari programatorul pierde foarte mult timp plasând manual puncte de interes, iar in cazul unor medii virtuale complexe este posibilă trecerea cu vederea a unor spaţii necartografiate, ce vor deveni invizibile pentru orice entitate din joc(asa cum se obervă in figura alăturată).

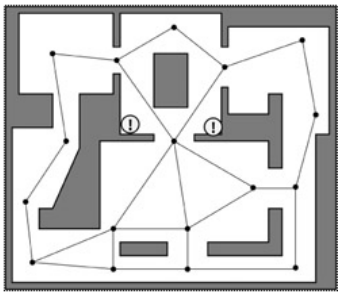


Fig. 4: Spaţii ce nu pot fi vizitate folosind punctele de navigare prestabilite
Sursa: Mat Buckland, Programming Game AI by Example

2.1.3 Reţea de navigare (NavMesh)

O metodă ce se bucura de o popularitate crescută in rândul dezvoltatorilor de jocuri constă in folosirea unei reţele de poligoane convexe pentru a descrie zonele ce pot fi parcurse pe hartă. Un asemenea poligon convex are proprietatea de a permite mişcarea neobstrucţionată între oricare două puncte ale sale. Astfel nodurile grafului pot conţine un spaţiu convex si nu un punct.

In figurile alăturate se poate observa diferenţa clară dintre mişcarea prin *waypoints* şi *navmesh*.

Fig. 5.1: Navigarea de la punctul A la punctul B folosind o reţea de navigare
Sursa: Paul Tozour - http://www.ai-blog.net/archives/000152.html

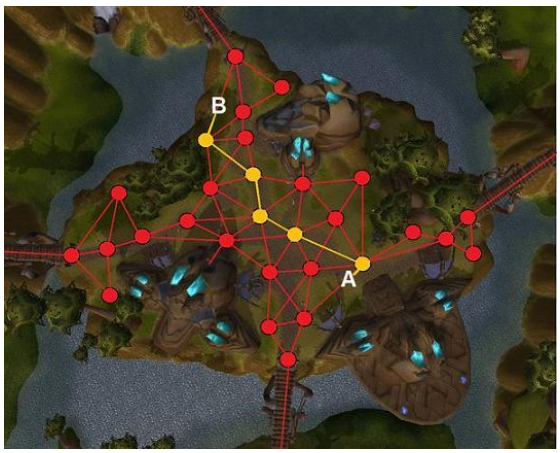


Fig. 5.1: Navigarea de la punctul A la punctul B folosind un sistem de puncte navigaţionale
Sursa: Paul Tozour – Solving Pathfinding

2.2 Căutarea Depth-First

DFS este un algoritm clasic de căutare sau parcurgere a unui arbore sau a unui graf, bazat pe stivă. Se incepe de la rădăcină (sau de la un nod selectat in cazul grafului) şi se explorează cât de departe este posibil pe fiecare ramură înainte de a se intoarce înapoi prin backtracking. Dupa cum se observă, metoda va parcurge exhaustiv graful, executand o multitudine de mişcari inutile.

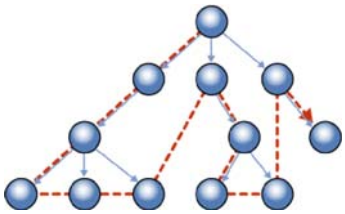


Fig. 6: Parcurgerea unui arbore folosind DFS

2.3 Căutarea Breadth-First

BFS este, de asemenea, o metoda clasică de căutare/parcurgere a unui graf, parcurgând sistematic fiecare soluţie, folosind o coadă, pe principiul FIFO. Nu foloseşte o metodă euristică şi nu ţine cont de destinaţie decât în momentul în care o găseşte. Dacă considerăm ca distanţele dintre noduri au acelaşi cost, distanţa drumului minim este costul inmulţit cu nivelul pe care se află nodul destinaţie.

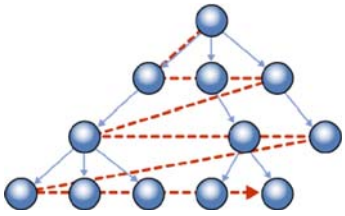


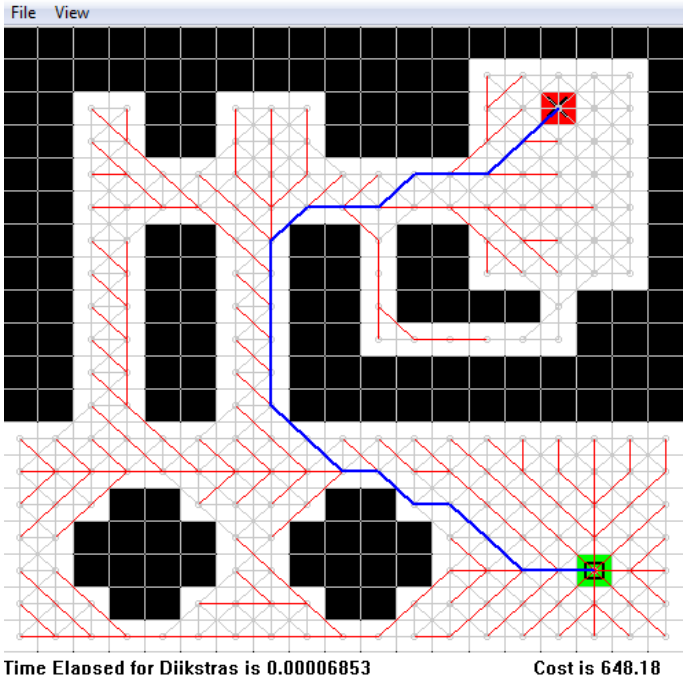
Fig. 7: Parcurgerea unui arbore folosind BFS

2.4 Algoritmul lui Dijkstra

O altă abordare clasică, insa de bază in cautarea drumului minim într-un graf îl reprezintă algoritmul lui Dijkstra, conceput in 1959 de omul de ştiinţă olandez Edsger Dijkstra, ce rezolvă problema drumului minim într-un graf cu muchii pozitive, calculând drumul minim între un nod al grafului şi toate celelalte noduri ale acestuia. Acesta este folosit îndeosebi în problemele de rutare din reţelele informatice.

2.4.1 Paşi:

1. Se creează o listă cu distanţe, o listă cu nodul anterior, o listă cu nodurile vizitate şi un nod curent.



2. Toate valorile din lista cu distanţe sunt iniţializate cu o valoare infinită, cu excepţia nodului de start, care este setat cu 0.
3. Toate valorile din lista cu nodurile vizitate sunt setate cu fals.
4. Toate valorile din lista cu nodurile anterioare sunt iniţializate cu -1.
5. Nodul de start este setat ca nodul curent.
6. Se marchează ca vizitat nodul curent.
7. Se actualizează distanţele, pe baza nodurilor care pot fi vizitate imediat din nodul curent.

Fig. 8: Găsirea drumul de cost minin într-un labirint folosind algoritmul lui Dijkstra
-Nodurile vizitate sunt marcate cu roşu
Sursa: O'Reilly, Killer Game Programming in Java - Pathfinding

8. Se actualizează nodul curent la nodul nevizitat care poate fi vizitat prin calea cea mai scurtă de la nodul de start.
9. Se repetă (de la punctul 6) până când toate nodurile sunt vizitate.

În figura 8 se observă drumul minim găsit de aplicarea algoritmului Dijkstra, iar timpul de execuţie este unul dintre cele mai bune, bazat pe complexitatea $O(|E| + |V| \log |V|)$ în cel mai rău caz posibil (E-nr. de muchii, V-nr. de noduri/vertecşi).

Următorul algoritm, A*, considerat printre cele mai bune in pathfinding, este practic o versiune al algoritmului lui Dijkstra, îmbunatăţit prin adăugarea unei funcţii euristice şi eliminarea operaţilor considerate inutile.

3 A* (A star)

A* este un algoritm de căutare bazat pe cel mai apropiat minim ce gaseşte drumul optim dintr-un nod iniţial şi un nod destinaţie, folosind o funcţie distanţă+cost euristică f(x) pentru a determina ordinea în care sunt vizitate nodurile din graf.

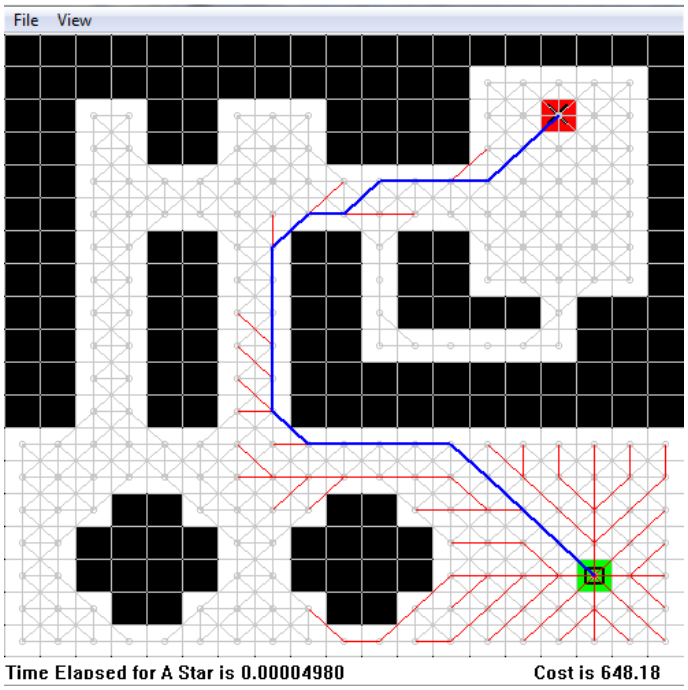
Funcţia este formată din suma altor doua funcţii:

- funcţia de cost notată de obicei g(x)
- şi o estimare euristică admisibilă a distanţei până la destinaţie – h(x)

Estimarea trebuie sa fie admisibilă, însemnând ca nu trebuie sa supraestimeze distanţa până la nodul destinaţie. În unele aplicaţii, această distanţă poate fi dată şi de o simpla linie ce conectează nodurile, deoarece este cea mai mică distanţa posibilă ântre două puncte.

Observăm în graficul de mai jos aceeasi problemă de pathfinding din fig. 8 rezolvată de data aceasta folosind algoritmul A*.

Notabilă este mult mai buna gestionare a memoriei, deoarece s-au facut foarte puţine operaţii inutile în drumul spre nodul destinaţie(liniile cu roşu), lucru dovedit şi de timpul de execuţie cu 30% mai rapid. Chiar dacă valoarea finală este aceeaşi, observăm că drumul este puţin diferit.



În cele ce urmează vom explica pas cu pas cum funcţionează acest algoritm şi ce îi permite să ruleze cu un spor de performanţă aşa mare.

La nivel de practică, diferenţa dintre un procedeu euristic şi unul clasic este asemănătoare cu o persoană care ştie in ce direcţie se află destinaţia şi incearcă să păstreze direcţia ocolind obstacolele şi un robot, încercând toate uşile până când găseşte soluţia optimă, fără să ţină cont de direcţie. Din această perspectivă, un rol foarte important îl are funcţia de estimare.

Fig. 9: Găsirea drumul de cost minin într-un labirint folosind algoritmul A*
-Nodurile vizitate şi direcţia sunt marcate cu roşu
Sursa: O'Reilly, Killer Game Programming in Java

3.1 Paşii algoritmului:

1. Se adaugă nodul de start în listă.
2. Repetă:
    a. Căutăm nodul cu cea mai mică valoare din listă şi îl considerăm nodul curent.
    b. Îl marcăm vizitat
    c. Pentru fiecare din cele 8 celule adiacente...
        - Dacă nu se poate folosi sau e deja vizitată, îl ignorăm. Altfel:
        - Dacă nu e in listă il adaugam si marcăm nodul curent drept părinte al acestuia. Înregistrăm costurile F,G si H ale nodului.
        - Dacă este deja în listă, verificăm dacă această cale este mai bună, iar în caz afirmativ schimbăm părintele nodului cu cel curent şi recalculăm G si F
    d. Ne oprim când:
        - Nodul destinaţie a fost marcat ca şi vizitat.
        - Nu am ajuns la nodul destinaţie şi lista este goală. În acest caz nu există drum.
3. Salvăm drumul, parcurgând invers şirul de părinţi.

3.2 Exemplu:

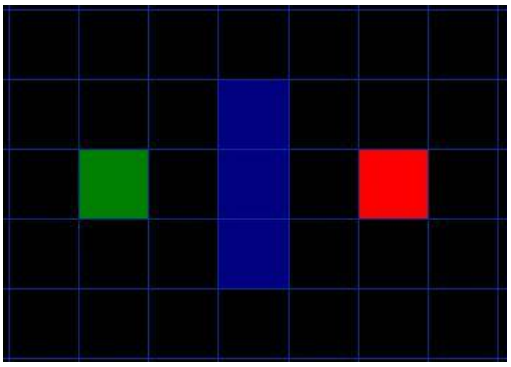


Fig. 10.1: Un exemplu simplu de problemă de tip labirint. Punctul verde este sursa, iar cel roşu destinaţia
Sursa: Patrick Lester, A* Pathfinding for Beginners

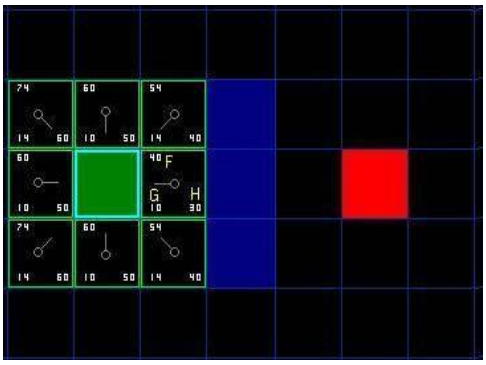


Fig. 10.2: Considerăm punctul iniţial drept punct de plecare şi marcăm apoi toţi vecinii săi, alături de costul de parcurgere aferent şi direcţia faţă de părinte
Sursa: Patrick Lester, A* Pathfinding for Beginners

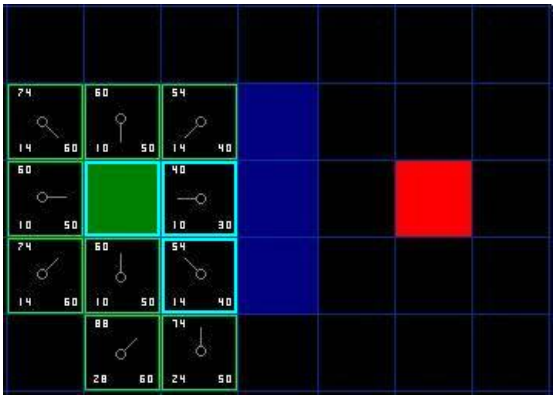


Fig. 10.3: Din nodurile marcate selectăm nodul cu valoare minimă conform funcţiei de cost şi marcăm vecinii acestuia iar apoi paşii se repetă pentru urmatorul nod cu valoarea minimă etc
Sursa: Patrick Lester, A* Pathfinding for Beginners

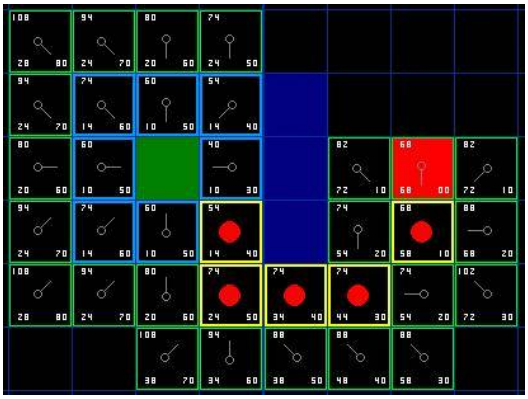


Fig. 10.4: Graficul final ne arată paşii realizaţi dar şi drumul de cost minim găsit parcurgând invers mişcările algoritmului. Se observă că nu au fost parcurse toate nodurile grafului.
Sursa: Patrick Lester, A* Pathfinding for Beginners

4 Mediul şi modul de programare

Aşa cum am putut observa pe parcurs, găsirea drumului optim este o problemă cu o complexitate ridicată, indeosebi datorită puterii totuşi limitate de calcul a PC-urilor dar şi a altor sisteme de calcul. Pe langă găsirea drumului minim, în dezvoltarea unui joc mai intervin factori, cum ar fi:

1. Collision detection
2. Probleme de fizică
3. Congestionarea punctelor de interes sporit
4. Mişcări coordonate etc

Fig. 11: http://www.java.com/en/

Fig. 12: http://www.eclipse.org/

Mediul de programare ales este unul mai puţin obişnuit, însă care a realizat progrese notabile în direcţia graficii 2D şi 3D şi ne pune la dispoziţie uneltele perfecte pentru realizarea sarcinilor propuse.

Java este un limbaj de programare orientat-obiect puternic, conceput de Sun Microsystems la începutul anilor 90, caracterizat de motto-ul „Write once, run everywhere", bazat pe o maşina virtuală ce poate rula orice program scris în java pe aproape orice platformă de calcul.

Environment-ul folosit, Eclipse, este open-source şi este susţinut de fundaţia Eclipse.

5 Concluzii şi dezvoltări ulterioare

- Realizarea unu joc complex de strategie în timp real (RTS) scris în Java care să rezolve cu succes diversele probleme legate de pathfinding-ul unităţilor din joc, Goal Behaviour şi alte probleme de AI.
- Portarea acestui joc pe alte platforme de calcul disponibile
- O mai bună înţelegere a modului în care funcţionează sistemele grafice ale sistemelor de calcul
- Aprofundarea cunoştinţelor privind mediul de dezvoltare Java

Bibliografie

[1] Mat Buckland, Programming Game AI by Example, Wordware Publishing, 2005
[2] http://en.wikipedia.org/wiki/A*_search_algorithm
[3] http://en.wikipedia.org/wiki/Dijkstra's_algorithm
[4] O'Reilly, Killer Game Programming in Java, May 2005
[5] DelphyX Game Tutorial One, Michael Dundee , http://edn.embarcadero.com/article/22928,
[6] Paul Tozour , Solving Pathfinding, http://www.ai-blog.net/archives/000152.html
[7] Patrick Lester, A* Pathfinding for Beginners

Bota Florentin
Anul III de studiu
Universitatea „Lucian Blaga" din Sibiu
Facultatea de Ştiinţe , Catedra de Informatică
Sibiu, str dr. Ioan Raţiu, nr 5-7
botaflorentin@yahoo.com

Sesiunea Națională de Comunicări Științifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Impactul legăturilor între elemente din conjuncturi diferite într-un mediu SSD

Francesco Capuzzo
Coordonator: Prof. Univ. Dr. Dana Simian

Abstract

The paper presents a new and original method for re-thinking a Decisions Support System (DSS) from a different perspective. A fact is seen as the result of a situation of equilibrium, where the two scales are a conjuncture and the event generated by it. Furthermore, in this work are described the relations between the elements of one or more conjunctures, named links. As the description of the internal and external links becomes exaustive, the more successful will a DSS be.

1 Introducere

Metoda prezentată vrea să exploreze noi posibilități de a gândi metodologia de abordare a sistemelor suport pentru decizii (SSD) într-un mediu managerial. Un SSD de succes este acela care asistă și nu înlocuiește elementul uman. Plecăm de la următoarea premiză: elaborarea deciziei constă în crearea de cunoștințe noi și este realizată de capacitățile cognitive folosind elementele de cunoaștere deja existente în depozitul de cunoștiințe sau achiziționate în acest scop. Folosirea capacităților cognitive presupune interpretarea realității, adică abstractizarea. Acest lucru este necesar pentru a putea crea metadate care vor fi elaborate de calculator. Față de calculator, omul are, printre celelalte, următoarele capacități deosebit de dezvoltate: conceptualizarea, intuiția si creativitatea. Metoda descrisă în această lucrare încearcă să ajute utilizatorul să folosească aceste capacități cât mai bine pentru a obține un set de metadate care, pe lângă datele de intrare, vor fi introduse în calculator. Acesta va restitui, din acest motiv, informații mult mai utile și previziuni cât mai apropriate de realitate. De aceea este foarte important sa vedem un sistem suport pentru decizii ca un parteneriat între resursele și capacitățile utilizatorului uman și a calculatorului. În acest parteneriat este necesar un nivel înalt de interacțiune între utilizator și calculator, aceasta fiind o facilitate a mediului de lucru SSD, care furnizează oportunități pentru utilizator să ghideze calculatorul în acele arii ale procesului de decizie cum ar fi conceptualizarea și intuiția, unde abilitățile utilizatorului sunt de departe superioare calculatorului. Automatizarea ar trebui restricționată la monitorizarea activităților de rezolvare a problemelor, detectarea conflictelor, executarea evaluărilor și a secvențelor de căutare și planificare.

2 Mediul SSD, conjuncturi, legături

În dezvoltarea unu SSD, vom folosi termenii conjunctură, eveniment, legături. Lucrarea de față se va concentra numai asupră acestor 3 elemente. Când vorbim de o conjunctură, ne gândim la totalitatea elementelor şi a legăturilor care permit funcţionarea armonioasă a unui mediu de afaceri, fiind acesta comercial, industrial, agricol sau de servicii. O funcţionare armonioasă permite atingerea obiectivelor dorite. De asemenea, capacitatea de a reconstitui conjuncturile, ne va permite să avem un instrument puternic pentru a înţelege cauzele unui eveniment şi pentru a putea face previziuni pe viitor. De obicei, SSD-urile focalizează atenţia asupra conjuncturile firmei luate în considerare. Acest lucru reprezintă o mare limitare atunci când vrem să obţinem previziuni de către sistemul nostru. Fiind economia de piaţă un mediu deschis în care competiţia generează situaţii neaşteptate, este evident că succesul oricărei firme depinde considerabil şi de relaţia ei cu exteriorul. Din acest motiv este fundamental sa avem la dispoziţie un instrument pentru recunoaşterea şi abstractizarea legăturilor externe.

Sistemul suport pentru decizii trebuie proiectat ca un set de instrumente şi nu ca un set de soluţii la un set de probleme predeterminat. Natura nedeterminată a problemelor complexe nu ne permite să prevedem cu un anumit grad de certitudine circumstanţele specifice ale problemelor viitoare sau termenii precişi ai soluţiei. În aceste circumstanţe este mai constructiv să furnizăm instrumente care vor extinde capacităţile decidentului într-un mediu foarte interactiv pentru rezolvarea de probleme. O reprezentare înaltă a obiectelor din lumea reală, care definesc relaţiile sistemului de probleme, formează baza interacţiunii dintre utilizatori şi sistem şi de asemenea, gradul de inteligenţă care poate fi încapsulat în componentele sale. Sistemul suport pentru decizii trebuie să fie un sistem bazat pe cunoaştere. În acest context, cunoaşterea poate fi descrisă ca o experienţă derivată din observarea şi interpretarea evenimentelor sau fenomenelor trecute, dar poate deriva şi din studiul acelor simulări care s-au dovedit câştigătoare. Bazele cunoaşterii captează această experienţă în forma regulilor, studiilor de caz, practicilor standard, descrierea tipică a obiectelor şi a obiectelor sistem care pot servi ca prototip. Aplicaţiile de rezolvare a problemelor tipice manipulează aceste prototipuri prin adaptare, rafinare, mutaţie, analogie şi combinare, pe care apoi le aplică soluţiei problemei curente.

3 Concluzii

Proiectul prezentat este în curs de dezvoltare din punctul de vedere a conceptelor, cât şi a implementării. Pentru implementarea voi folosi mediul Visual Web Developer 2008, baze de date SQL Server 2005. Odată implementat, sistemul nu va furniza răspunsuri la întrebări, dar va permite utilizatorului, prin filtrări, recunoaşteri şi comparaţii, să identifice relaţiile de cauză/efect şi să simuleze noi situaţii de echilibru. O firmă de obicei judecă în termeni de bilanţ şi PPM. Metoda prezentată, vrea să pună în discuţie corectitudinea evaluării trecutului pentru a stabili, de exemplu, că o situaţie acceptată ca fiind pozitivă nu a fost atare; în consecinţă vor trebui luate măsuri pentru a îmbunătăţi performanţele viitoare. Acest proces nu este altceva decât stabilirea unui nou target. De fapt necesitatea stabilirii unui target mai greu de atins decât cel precedent, este la fel ca a spune că targetul precedent, care poate a fost atins cu succes, nu a fost stabilit după potenţialul resurselor umane ale firmei. Totuşi, procesul de formare a resurselor umane cere timp şi trebuie efectuat treptat. Această constrângere este o piedică psihologică care limitează imaginaţia angajaţiilor şi managerilor firmei. Metoda prezentată vrea să furnizeze instrumente pentru recunoaşterea potenţialelor probleme şi/sau oportunităţi, şi în acelaşi timp vrea să

sporească imaginaţia şi creativitatea în mediul firmei. Sistemul va permite utilizatorilor sa foloseasca la maxim acele caracteristici umane care deosebesc omul de calculator: conceptualizarea, intuitia si creativitatea. Datele vor fi elaborate din aceasta prizma, abstractizate si furnizate calculatorului sub forma de metadate, spre obtinerea uneia sau mai multor solutii ale problemei de rezolvat

Bibliografie

[1] BÎZOI Mihai, *Sisteme suport pentru decizii. Utilizare. Tehnologii. Construire*, Academia Româna, Sectia Ştiinta si Tehnologia Informatiei, Institutul de Cercetări pentru Inteligenţa Artificială, 2007

[2] COURTNEY James F., *Decision making and knowledge management in inquiring organizations: towards a new decision-making paradigm for DSS,* Elsevier, 2001.S

[3] CHINNECK John W., Feasibility and Infeasibility in Optimization, Springer, 2008.

Surse Internet

1: http://en.wikipedia.org/wiki/TRIZ

2: http://www.ideationtriz.com/new/materials/TbMainpostulates.pdf

3: www.ktree.it/ktimg/file/Un'introduzione%20all'I-TRIZ.pdf

4: http://www.triz-journal.com/archives/2006/12/09.pdf

5: http://www.eclipse.org/birt/phoenix/

6: http://rapid-i.com/

CAPUZZO FRANCESCO
Universitatea Lucian Blaga, Facultatea de Ştiinţe
Informatica
Str. Dr. Ioan Raţiu, nr. 5-7, Sibiu
ROMÂNIA
E-mail: francesco.capuzzo@gmail.com

Sesiunea Națională de Comunicări Științifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Soluție mobilă pentru cumpărături - sHELPy

Luiza Cicone
Coordonator: Asist. univ. Ing. Alexandru Radovici

Abstract

sHELPy este o aplicaţie destinată telefoanelor mobile, dezvoltată pe platforma Google Android, ce îşi propune să faciliteze gestionarea cumpărăturilor. Ea permite crearea, modificarea şi partajarea unei listei de cumpărături şi în plus realizarea de profile particularizate în funcţie de produse sau magazine. Aplicaţia foloseşte diferite module integrate de Google, ca de exemplu localizarea utilizatorului sau trimiterea de alerte.

1 Introducere

Obiectivul companiei Google în domeniul telefoniei mobile este să transforme aceste terminale în mijloace de accesare a Internetului, cu ajutorul motorului de căutare Google şi să folosească, eventual, alte servicii ale companiei. Astfel a fost lansat Google Android care nu este doar un sistem de operare ci şi o platformă mobilă ce cuprinde mai multe funcţionalităţi. Sistem de operare mobil (având la bază sistemul de operare Linux, cu toate avantajele oferite de acesta privind securitatea informaţiilor, fiabilitate şi performanţă), framework de dezvoltare, ce permite reutilizarea resurselor software între aplicaţii, browser web integrat, librărie grafică (2D cât şi 3D), suport SQL, GSM, 3G, EDGE, Bluetooth, WiFi, GPS, camera video/foto şi accelerometru.
Inovaţia pe care o reprezintă Android se referă în special la programe. Google a pus la dispoziţie dezvoltatorilor o platformă mobilă completă, de ultimă generaţie, le-a oferit acestora uneltele necesare dezvoltării de aplicaţii pentru această platformă, a făcut publice specificaţiile şi documentaţia. Dar, pe de altă parte, crearea unei astfel de aplicaţii nu este deloc uşoară datorită restricţiilor precum: ecranul mic, viteză şi memorie reduse, ecranul tactil imprecis, eventuala lipsă a unei tastaturi [1].
În ziua de azi, lumea se confruntă cu problema lipsei de timp. Totul se face în mare viteză, una din consecinţele acestui fapt fiind uitarea diferitelor taskuri importante. Partea bună este că au fost create gadgeturi ce ne simplifică gestionarea activităţilor. În plus, lumea are tendinţa de a nu ezita când îşi achiziţionează un astfel de dispozitiv, lucru bun din punctul de vedere al producătorilor şi al dezvoltatorilor de aplicaţii.
O analiză amănunţită a ofertei de aplicaţii pentru dispozitive mobile a arătat că există numeroase aplicaţii ce se ocupă cu gestionarea diferitelor baze de date, chiar şi cu scopul de a crea o listă de cumpărături. Pe de altă parte recenta introducere de funcţii de localizare pe telefoanele mobile a condus la dezvoltarea unui număr relativ mare de aplicaţii specifice. Aplicaţia propusă, sHELPy, aduce un element de noutate prin combinarea celor două tipuri de aplicaţii anterior descrise.

2 Descrierea aplicaţiei

sHELPy este o aplicaţie utilitară ce propune gestionarea rapidă şi eficientă a cumpărăturilor casnice precum şi o serie de alte facilităţi aferente. Are o interfaţă grafică atractivă şi uşor de folosit.
sHELPy este alcătuită din trei componente principale: aplicaţia propriu-zisă, widgetul şi o bază de date. Widgetul este un mic dispozitiv ce se află pe desktopul telefonului mobil şi păstrează o instanţă din aplicaţie. Fie că utilizatorul vrea informaţii despre lista de cumpărături sau despre magazinele din apropiere, widgetul este în permanentă legătură cu aplicaţia şi cu baza de date astfel încât să afişeze ceea ce i se cere.
Baza de date este partea din „spate" a aplicaţiei, pe care utilizatorul o vede doar sub forma unei liste sau a unui profil. Ea stochează informaţii despre produse, magazine, setări. Astfel toate informaţiile din aplicaţie sunt extrase de aici. Această bază de date este modelată cu ajutorul suportului SQL integrat de Android.
Aplicaţia propriu-zisă poate fi accesată prin widget sau din meniul telefonului. Se poate naviga uşor prin aceasta cu ajutorul butoanelor de control (start, înapoi) sau din meniu. Funcţiile principale sunt lista de cumpărături, alertele, localizarea şi grupul de partajare a informaţiilor.

2.1 Lista de cumpărături

Cea mai importantă funcţie a aplicaţiei este crearea listei de cumpărături (Fig. 1). Aici utilizatorul are de ales între adăugarea produselor proprii sau alegerea acestora dintr o listă predefinită. La fiecare produs adăugat pot fi selectate atribute precum prioritatea (cu valori de la 1 la 5), cantitatea, unitatea de măsură, un preţ estimativ şi de asemenea pot fi alese magazinele de unde utilizatorul preferă să-şi achiziţioneze produsele dorite.
Ulterior, lista poate fi modificată cu uşurinţă cu ajutorul butoanelor din meniu: adăugare, respectiv editare şi ştergere. Setările legate de unităţile de măsură, monedă pot fi modificate din meniul principal.

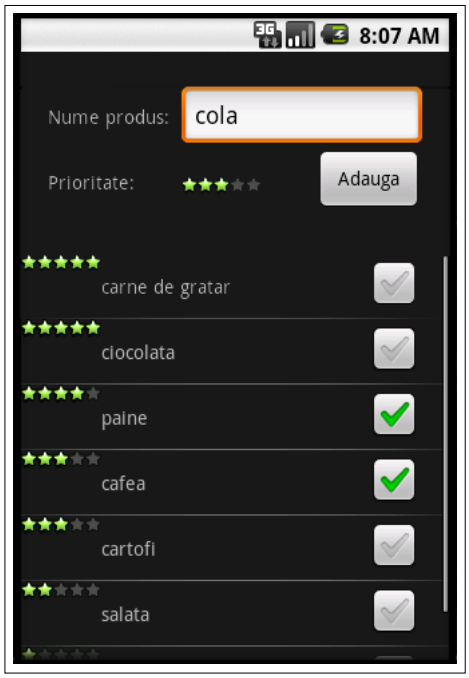


Fig. 1

Fig 2

2.2 Localizarea

sHELPy integrează modulul de localizare Google (Fig 2). Cu ajutorul acestuia, aplicaţia oferă utilizatorului date cu privire la poziţia acestuia, adresa magazinelor, distanţele aferente sau diferite trasee pentru cumpărături. În funcţie de setările făcute, această funcţie poate filtra informaţiile în raport cu alţi parametri. Aglomeraţia la magazine sau pe bulevarde la orele de vârf sau programul magazinelor (zilele/orele în care acestea sunt închise).

2.3 Alertele

Aplicaţia permite realizarea unor alerte în cazuri prestabilite sau alese de utilizator. Aceste alerte pot fi programate periodic sau pentru anumite produse, grade de prioritate sau la apropierea de magazine. Spre exemplu dacă utilizatorul se apropie de un supermarket şi are în lista de cumpărături câteva produse cu prioritate ridicată (5/5, de cumpărat în următoarele 24 de ore), primeşte alertă prin Widget, cu sau fără sunet (Fig. 3).

2.4 Grupul de partajare

O altă caracteristică utilă a aplicaţiei este că informaţia din listă poate fi actualizată de către alte persoane prin reţeaua wireless. Pentru oricare listă utilizatorul poate seta membrii ce pot avea acces la aceasta cu drepturi de editare sau doar de citire (Fig 4). Din punctul în care avem o relaţie de partajare, fiecare dispozitiv se conectează la serverul aplicației și își actualizează informațiile din listă. Această operațiune se face în mod automat, fără a fi nevoie de actualizarea permanentă a utilizatorului.

Fig. 3

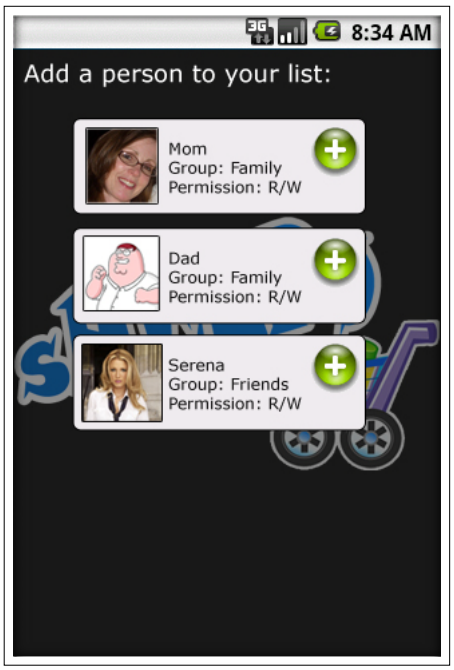


Fig 4

3 Dezvoltarea ulterioară

Prima îmbunătăţire adusă aplicaţiei va fi cea de creare de liste în mod automat, după preferinţele utilizatorului. Acestea se pot realiza la intervale diferite de timp, lunar, săptămânal, cu sau fără alerte pentru produsele casnice uzuale.

De asemenea, aplicaţia este creată pentru o dezvoltare ulterioară folosind un modul SmartPrice. Acesta reprezintă o bază de date cu o varietate de produse şi magazinele distribuitorilor. Conţine preţuri, disponibilitate (in stoc) şi de asemenea review-uri ale cumpărătorilor. Acest modul este foarte util deoarece permite realizarea anumitor economii. În cazul diferenţelor de preţuri dintre magazine, organizează cumpărăturile astfel încât utilizatorul să iasă în avantaj (preţuri, drumuri, cantităţi).
O altă direcţie de dezvoltare este colaborarea cu diferite reţele de socializare cum ar fi Facebook, MySpace, Twitter. Astfel, pentru crearea unei liste nu mai este nevoie ca utilizatorul să adauge fiecare membru în parte. Foarte simplu, acesta se conectează la baza de prieteni şi alege cui să trimită o invitaţie. După ce o persoană acceptă o invitaţie, această poate accesa lista de cumpărături a aplicaţiei, unde poate adăuga, modifica, selecta produse.

4. Concluzii

sHELPy este soluţia potrivită pentru orice persoană dornică de o îmbunătăţire a modului de viaţă. Este o aplicaţie rapidă, eficientă, ce se adresează orcărei persoane şi are un foarte mare potenţial de dezvoltare. Se mulează perfect pe tendinţele actuale de a păstra legăturile cu prietenii, a accesa în permanenţă internet-ul şi de a realiza o organizare potrivită a timpului. Elementul de originalitate îl constituie faptul că îmbină o bază de date cu Google Maps şi cu funcţiile de partajare.

Bibliografie

[1] Mark L. Murphy, Beginning Android, Apress, 2009
[2] http://developer.android.com.

CICONE Luiza
Universitatea Politehnică Bucureşti
Calulatoare şi tehnologia informaţiei
Splaiul Independenţei nr.313, Bucureşti
ROMÂNIA
E-mail: luiza.cicone@rdslink.ro

Sesiunea Națională de Comunicări Ştiințifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Portarea bibliotecii OpenYMSG pe platforma Android

Domnina BURCĂ-FLOREA, Marius CONSTANTINESCU
Coordonator: Asist. univ. Ing. Alexandru RADOVICI

Abstract

Mobile devices have become more and more common these days. Recent hardware and software development have transformed these devices into real computers that fit into ones pocket. One of these serious developments is the Google's Android platform, an open source sistem for mobile devices. This paper presents the porting of the *openymsg* library to Android and the implmentation of a YMSG client. The final goal for this research is the implementation of a multiprotocol messenger client, as such a piece of software is curently missing for these types of mobile devices.

1 Introducere

Industria dispozitivelor mobile a evoluat foarte mult pe parcursul ultimilor ani. Datorită evoluției hardware și software, aceste telefoane mobile au devenit veritabile calculatoare ce încap în buzunar. Trebuie avut totuși în vedere faptul că puterea de procesare a acestor dispozitive este foarte limitată. Deși procesoarele sunt destul de rapide, unele ajungând chiar la viteze de peste 1 GHz, energia consumată este extrem de limitată deoarece bateriile folosite nu au evoluat în același ritm.

Dacă până acum programele ce rulau pe dispozitive mobile erau destul de simple, odată cu creșterea puterii hardware au apărut și sisteme de operare mobile foarte avansate. Considerăm un sistem de operare mobil avansat un sistem ce replică funcționalitatea unui calculator personal. Un astfel de sistem este cel dezvoltat de către Google, și anume Android. Acesta reprezintă de fapt adaptarea sistemului *Linux* pe o platformă mobilă și adăugarea de software specific. Deoarece abordarea programării pentru Android este puțin diferită, vom discuta detaliat acest subiect în următorul capitol.

Deoarece Android este un sistem relativ nou apărut pe piață, numărul programelor disponibile este realtiv limitat. Scopul nostru final este realizarea unui program de tip multi-messenger, program ce va facilita și integra comunicarea cu prietenii indiferent de tipul protocolului de mesagerie folosit. Primul pas spre implmentarea acestui software a fost portarea bibliotecii *openymsg* pe sistemul Android. Acesta este scrisă în Java, însă într-un *dialect* incompatibil cu Android. Vom detalia acest subiect pe parcursul lucrării.

Al doilea pas în realizarea dezideratului propus a fost dezvoltarea unei interfețe grafice specifice Android pentru testarea portării bibliotecii. Cu alte cuvinte, lucrarea prezintă un client pentru protocolul YMSG pe platforma Android.

2 Google Android

Deoarece modul de abordare al programării pentru Android este puțin diferit, considerăm că este ncecesară o scurtă prezentare a acesteia. Android reprezintă de fapt o platformă pentru dipozitive mobile, nu un sistem propriu-zis. Sistemul de operare este versiunea 2.6.27 a kernel-ului de Linux și o adaptare a bibliotecilor *user-space*, numită *bionic*. În alte cuvinte, avem un sistem Linux ce rulează pe telefoane mobile. Adevărata putere a platformei Android o reprezintă mașina virtuală *Dalvik*, biblioteciile pentru acesta și modul de abordare al programării aplicațiilor.

Limbajul de programare folosit pentru Android este Java. Mașina virtuală pe care vor rula aceste programe nu este cea standard (realizată de către Sun Microsystems) ci se numește *Dalvik*. Acesta reprezintă o implmentare specializată pentru telefoane mobile, implmentare care urmărește să minimezeze spațiul de memorie ocupat și să optimizeze modul de colectare al gunoaielor. Din acest motiv, mașina folosește alt cod mașină, diferit de cel standard java. Astfel clasele java compilate (.class) trebuie în prealabil traduse în cod *Dalvik* (.dex). Aici însă intervin niște probleme de compatibilitate. Dalvik recunoaște cod java până la versiunea 1.5 inclusiv, cu câteva limitări: au fost excluse bibliotecile pentru interfețe grafice, bibliotecile pentru imprimare și alte câteva biblioteci specializate. Acesta este și motivul pentru care *openymsg* trebuie modificată pentru a funcționa sub Android.

Un alt lucru inovativ introdus de către Android este modul de abordare al programării aplicațiilor. Modelul clasic de programare prevede pentru fiecare program un singur punct de intrare, în general reprezentat de către funcția *main*(). În viziunea Android, fiecare program este compus din componente diferite, fiecare dintre acestea putând fi definit ca punct de intrare. În acest fel, aceeași aplicație poate avea diverse puncte de pornire (intrare), în funcție de modul și locul în care trebuie executată. Tipurile de componente Android posibile sunt:

- **Activități** – reprezintă o fereastră afișată pe toată suprafața ecranului. Acest tip de componentă este una din cele două vizibile de către utilizator. Există mereu o singură astfel de componentă activă. În cazul în care fereastra nu este vizibilă și sistemul necesită mai multă memorie, există riscul ca acesta să fie ștearsă.
- **Widget** – este o componentă ce este afișată pe ecranul principal al telefonului. În general ocupă o suprafață mică și este folosită pentru notificări, urmând ca în urma unui click sau apăsării unui buton să deschidă o activitate.
- **Servicii** – reprezintă o componentă ce rulează în fundal si nu interacționează direct cu utilizatorul. Acesta este în general folosită pentru procesări de date și interacționează cu o activitate sau un widget.
- **Intenții** – reprezintă un eveniment de sistem. Orice programator își poate defini anumite evenimente. Acest tip de componentă nu este importantă pentru lucrarea de față.
- **Receptori de evenimente** – acesta reprezintă de fapt un observator pentru evenimentele publice ale sistemului. Este compus dintr-o singurp funcție ce este apelată la apariția unui anumite eveniment.

Pentru lucrarea de față vom folosi toate tipurile de componente, mai puțin Intențiile.

3 Portarea bibliotecii OpenYMSG

OpenYMSG este o bibliotecă scrisă în Java ce implmentează un subset al protocolul YMSG (Yahoo Messenger). Deoarece acest protocol nu este public și este propietar, bibioteca implmentează protocolul *descoperit* prin *reverse engineering*. Din păcate acesta înseamnă ca

anumite funcţii ale protocolului nu sunt disponibile. Deşi este scrisă în Java, acesata nu respectă restricţiile impuse limbajului de către *Dalvik*, astfel neputând fi folosită direct.

Pentru folosirea bibliotecii sub Android, au fost necesare diverse modificări ale acesteia. Principala problemă întâlnită a fost incompatibilitatea anumitor clase din bibliotecă cu *Dalvik*. La rulare se obţinea mereu *VerificationError*. Problema provine din folosirea anumitor strcuturi de Colecţii ce nu sunt compatibile cu versiunea Java 1.5. A fost necesară încolurirea unora sau dezactivarea completă a acestora. Datorită documentaţiei foarte slabe a bibliotecii, unele funcţii ale acesteia nu ne sunt cunoscute, aşa că am preferat să dezactivăm clasa şi să testăm dacă biblioteca este în continuare funcţională.

La momentul scrierii acestei lucrări, biblioteca a fost aproape complet portată, mai existând încă probleme de compatibilitate ce sperăm să fie rezolvate cât mai curând.

4 Implmentarea clientului pentru YMSG

Pentru implmentarea clientului se va folosi biblioteca OpenYMSG am folosit patru din cele cinci componente Android posibile.

4.1 Conexiunea cu serverul Yahoo

Conexiunea cu serverul Yahoo, implicit interacţiunea cu biblioteca *openymsg*, este implementată folosind un serviciu Android. Acest tip de componentă este specializată pentru rularea în fundal. Astfel are un timp de viaţă îndelungat, serviciile fiind ultimele procese oprite în cazul lipsei de memorie.

Android presupune existenţa unei conexiuni permanente la reţeaua Internet. Din acest motiv, orice utilizator are posibilitatea să fie mereu conectat la serverul de Yahoo. Trebuie totuşi amintit că, spre deosebire de calculator, telefonul va fi ţinut în buzuar mare parte din timp. Astfel utilizatorul nu priveşte mereu ecranul. Lista de prieteni şi conversaţiile nu trebuiesc afişate mereu. Se economisesc resurse importante prin plasarea conexiunii la server într-un serviciu.

Deoarece conexiunea la Yahoo este în fundal, utilizatorul trebuie să fie anunţat la apariţia unui eveniment nou. Acest lucru se realizează prin trimiterea unui mesaj către sistemul de notificare Android. Acesta va alerta utilizatorul care la rândul său are posibilitatea să deschidă o activitate în care să vadă mesajul. Sistemul este identic cu primirea unui SMS.

4.2 Interfaţa grafică

Interacţiunea cu utilizatorul este implmentată prin componente de tip activitate. Acestea sunt ferestre ce ocupă tot ecranul dispozitivului. Prin intermediul acestora utilizatorul are posibilitatea să vizualizeze lista de priteni, să editeze lista de prieteni, să răspundă la mesajele primite sau să iniţieze conversaţii noi. Programul mai dispune şi de o fereastră specială de configurare în care utilizatorul are posibilitatea să seteze diverşi parametrii ai programului.

O altă componentă a interfeţei grafice este un widget. Pentru a avea acces rapid la lista de prieteni, am realizat un widget pentru ecranul de pornire al telefonului. Acesta comunică cu serviciul din findal şi afişează olista de prieteni ai utilizatorului.

5 Dezvoltări ulterioare

Lucrarea de faţă prezintă portarea bibliotecii OpenYMSG pentru Android şi un exemplu de utilizare. Scopul final este însă realizarea unui program de tip multi-messenger pentru Android. Acesta presupune integrarea sub aceeaşi interfaţă grafică a mai multor protocoale pentru mesagerie. De asemenea, ne propunem realizarea unei punţi între protocoale, asta prespunând utilizarea protocoalelor ca mediu de transport. Spre exemplu, daca în prezent Alice vrea să vorbească cu Bob, dar aceştia au protocoale diferite, atunci unul din ei va trebui să îşi creeze un

alt cont pentru protocolul celuilalt. Dacă însă Alice şi Bob au o prietenă comună, Tracy, care are conturi pe abele protocoale, aceştia ar putea comunica folosind conturile lui Tracy pentru transport. Evident, Tracy trebuie să fie logată în acel moment. Aici însă intervin nişte probleme de securitate.

6 Concluzii

Deoarece sistemele mobile devin din ce în ce mai puternice, le putem deja asemui cu calculatoare ce pot fi ţinute în buzunar. Ceea ce însă defierenţiază dispozitivele mobile de calculatoarele clasice este puterea de calcul. Chiar dacă procesoarele acestora sunt destul de puternice, energia disponibilă este foarte limitată (de către baterii). Pentru a putea realiza sarcini similare cu calculatoarele, programele pentru dispozitivele mobile trebuie gândite diferit.

Pe parcursul acestei lucrări am încercat să vedem în ce măsură este realizabilă portarea bibliotecii *openysmg* pentru Android şi care sunt *costurile* construirii unui client pentru acestea.

Cercetările noastre au confirmat posibilitatea portării parţiale ale bibliotecii şi posibilitatea implmentării unui client eficient. Consumul de energie al bateriei nu a crescut simţitor la folosirea programului.

Bibliografie

[1] MURPHY, Mark; *Beginning Android*, Apress, 2009

[2] BURNETTE, Ed; *Hello Android: Introducing Google's Mobile Development Platform*, Pragmatic Bookshelf, 2009

[3] GRAMLICH, Nicolas; *andBook* – release .002

[4] ***, *Java Yahoo Messenger Library – OpenYMSG*, http://sourceforge.net/apps/trac/openymsg/

[5] ***, *Android Developer*, http://developer.android.com

BURCĂ-FLOREA Domnina
Universitatea „Politehnica" din Bucureşti
Calculatoare
Spl. Independenţei Nr. 313, Bucureşti
ROMANIA
E-mail: domnina.burca@gmail.com

CONSTANTINESCU Marius
Universitatea „Politehnica" din Bucureşti
Calculatoare
Spl. Independenţei Nr. 313, Bucureşti
ROMANIA
E-mail: constantinescu.marius@gmail.com

Sesiunea Națională de Comunicări Ştiințifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Multi-axys encoder using quadrature signals

Tudor Consantinescu
Coordonator: Asist. univ. Ing. Alexandru Radovici

Abstract

Lucrarea de fata prezinta realizarea unui sistem de encoding folosind semnale de cuadratura si o placa FPGA.

1 Problematica

Pentru realizarea unui robot sau a unui sistem automat ce are de parcurs un anumit drum descris de o functie calculata prin diverse metode numerice, este necesar un sistem de encoding care in unele cazuri poate scuti costurile ridicate ale senzorilor. Asadar, in acest scop am incercat sa gandim un sistem de encoding bazat pe 4 semnale de cuadratura si un FPGA cu care sa analizam datele.

2 Semnale de cuadratura

2.1 Definitie

Semnalele de cuadratura sunt semnale dreptunghiulare cu o diferenta de faza de 90 de grade.Acestea sunt folosite pentru diverse procesari si implementari ce au loc in sistemele moderne de comunicatii digitale.

2.2. Aplicatii

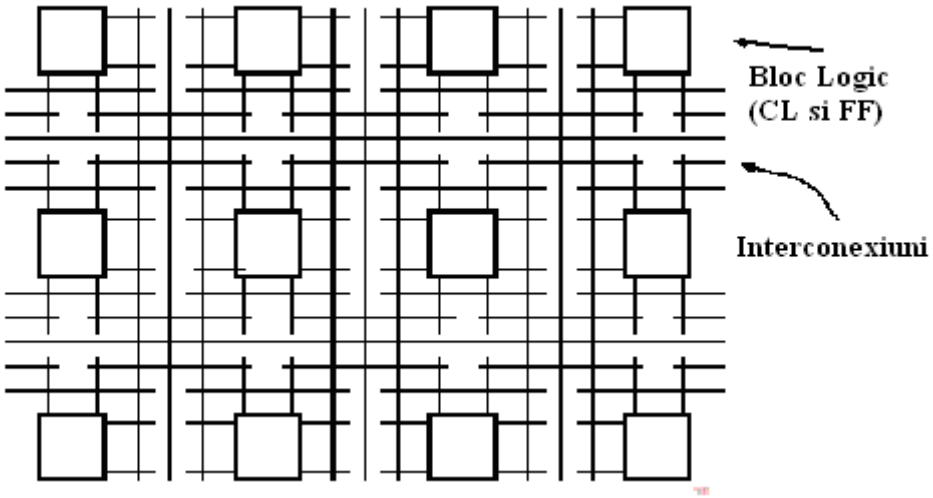
Semnalele de cuadratura sunt folosite in diverse aplicatii din domeniul procesarii de semnal digital cum ar fi:

- sisteme radar
- diverse modulatoare
- diferente de timp in scheme de detectie radio

3. FPGA

3.1 Definitie

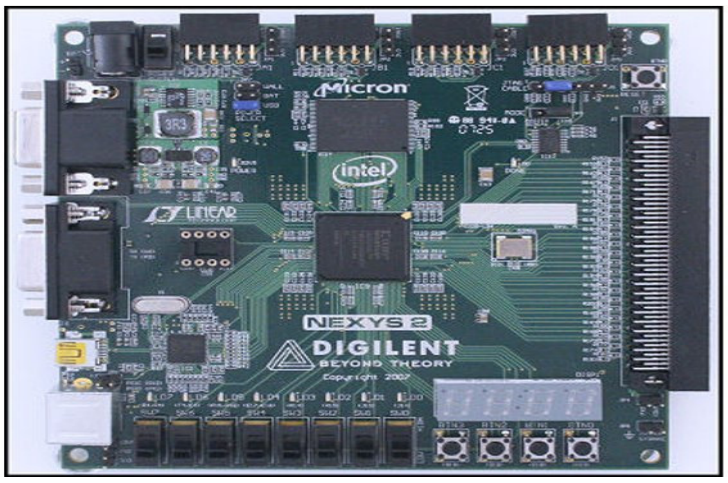
FPGA (Field Programmable Gate Arrays) sunt dispozitive logice programabile, ce contin blocuri logice si bistabile, prevazute cu facilitatile necesare configurarii de catre utilizator atat atat a interconexiunilor dintre blocurile logice cat si a functiei fiecarui bloc.Iata o imagine de asamblu simplificata a unui FPGA:

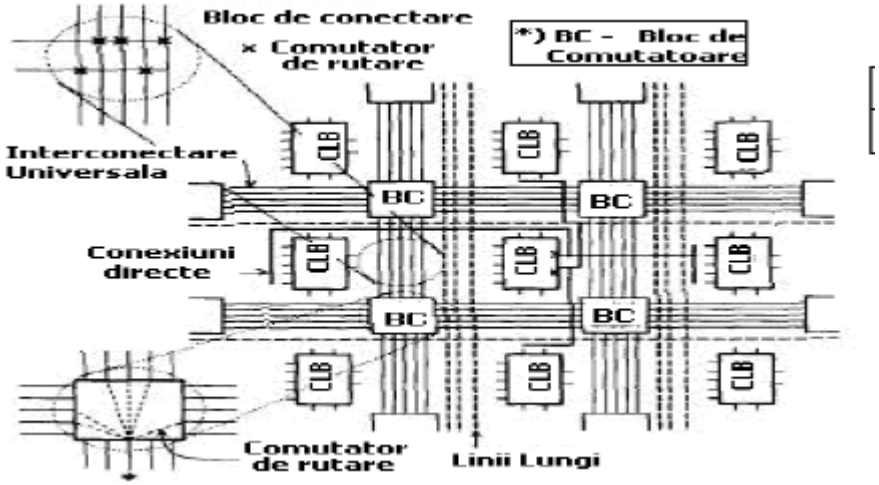


Apare deci problema diferentei intre CPLD si FPGA. Diferenta consta in gradul de integrare a blocurilor logice; in timp ce un CPLD contine in jur de 100 asemenea blocuri, un FPGA poate incorpora pana la 100000. O alta diferenta consta in faptul ca placilr FPGA sunt bazate pe memorii de tip RAM ce trebuiesc reprogramate dupa o intrerupere e tensiunii, contrar dispozitivelor EEPROM si CPLD.

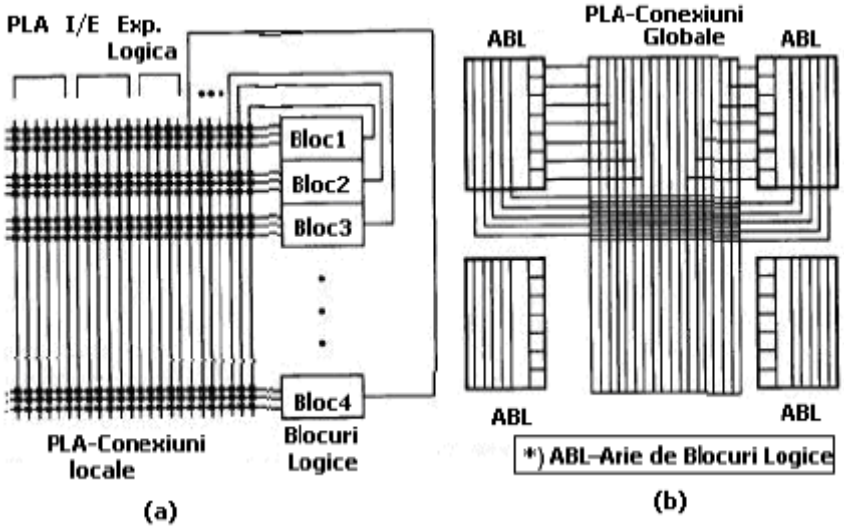
3.2 Producatori

Exista in jur de 5 companii producatoare de dispozitive FPGA insa primele doua(Xilinx si Altera) detin monopolul. Xilinx a inventat aceste dispozitive si este cel mai important nume de pe piata FPGA.Acestia utilizeaza trasee de interconectare care inconjoara fiecare bloc logic combinational(CLB), la intersectiile traseelor fiind plasate blocuri de comutare(BC), organizate sub forma de matrici, formate din tranzistoare de trecere, controlate pe grile cu ajutorul unor celule de memorie SRAM.





Pe de alta parte, ALTERA propune o structura bazata pe doua niveluri ierarhice:local a) si global b)



Dispozitivul utilizat de noi face parte din gama de produse a Xilinx si poarta nume de Nexys2.

Dupa cum se poate observa in figura, acest model contine multiple dispozitive de i/o cum ar fi:

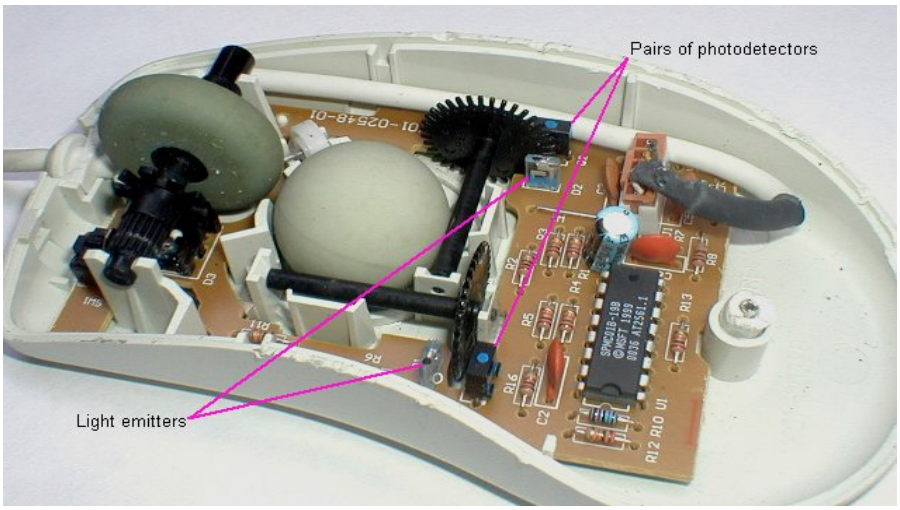
- port VGA pe 8 biti
- port RS232
- conector cu 6 pini ce poate fi interfatat PS/2
- 8 switchuri si 4 butoate ca dispozitive de input
- un dispozitiv de afisare pe 7 segmente

Deasemenea FPGA-ul contine un ceas ce functioneaza la o frecventa de 50 MHz, o memorie Flash de 16MB produsa de Intel si un SDRAM de 16MB produs de Micron.

Placa mai contine inca 4 conectori Pmod scopul lor initial fiind acela de a conecta diverse sisteme de achizitii de date produse de aceasi inteprindere.

4.Emitatorul de semnal de cuadratura

Unul dintre cele mai utilizate dispozitive de acest gen este mouse-ul. La deschiderea acestuia se poate observa urmatoarea figura:



Mouse-ul contine doua rotite dintate(una pentru axa X, cealalta pentru axa Y) ce au de o parte si de cealalta un LED infrarosu ce emite pe o frecventa de 38 de Mhz respectiv un receptor infrarosu. Receptorul de IR este constituit din doua fototranzistoare ce au colectorii legati impreuna iar bazele in aer.Astfel, daca notam cu A si B cei doi emitori ai receptorului IR vom avea urmatoarea succesiune de pasi: presupunem ca la momentul t=0 emitorul A se afla in dreptul unui „gol” in timp ce emitorul B se afla in dreptul unui”dinte” al rotii. Dupa o secventa de miscare ambii emitori se vor afla in dreptul gaurii.Dupa inca o miscare in aceasi directie cel de-al doilea emitor va fi blocat.Asadar, in cod binar am putea avea urmatoarea inlantuire de evenimente:

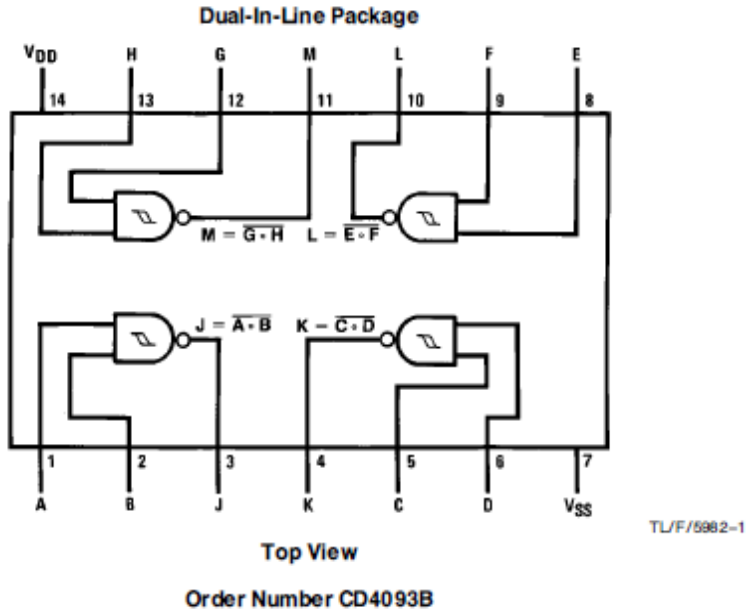
(A, B):..., (1,0), (1,1), (0,1), (0, 0), etc.

Semnalele emise sunt trimiste unui microcontroller (SPCP05A/04B in cazul de fata), produs de SunPlus, un circuit integrat ce are rolul de prelucra informatiile si a le pregati pentru a putea fi trimise prin intermediul protocolului PS2. Un lucru bizar a fost sa descoperim ca, catozii celor 3 LED-uri IR nu erau conectati la masa ci, la unul din pinii integratului (PB16). In urma cercetarilor ce au urmat, s-a descoperit LED-urile erau conectate direct la integrat pentru a modula frecventa acestora la 38MHz.

Asadar, pentru a trimite datele unui sistem care sa le prelucreze(in cazul de fata FPGA-ul) am avut de ales intre doua soutii:

- crearea unei interfete PS2 din punct de vedere logic pe FPGA
- inlocuirea microcontrollerului cu un alt circuit integrat

Avand in vedere parametrii de timing destul de complicati ce trebuiau respectati pentru a crea o interfata PS2, am decis ca solutia potrivita ar fi sa inlocuim integratul existent cu un altul, si anume



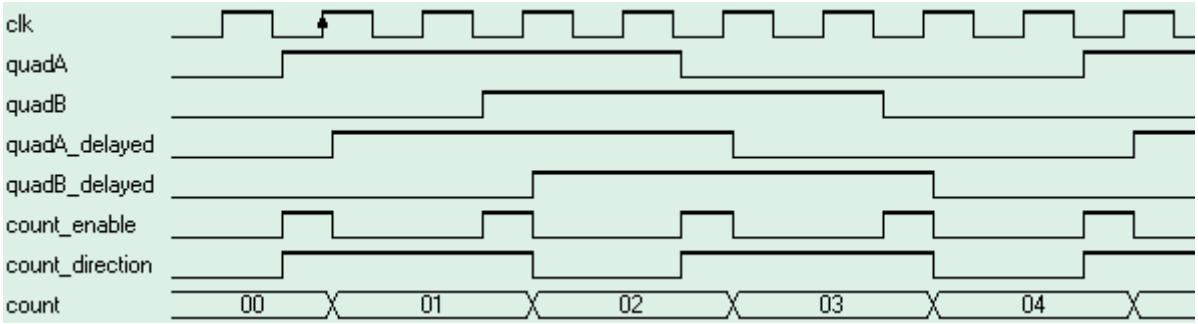
CD4093.

Dupa cum se poate observa si in schema, acest integrat contine 4 porti NAND cu triggere Schmitt.Avand 4 semnale emise de la cei 2 receptori IR am legat intre ei pinii de intrare ai portilor NAND obtinand astfel 4 invertoare.

De asemenea am considerat necesara conectarea unei rezistente de 100 de Ohmi pentru a micsora curentul pe cele trei LED-uri infrarosii.

5.Decalarea semnalului de cuadratura

Avand semnalele emise de catre circuitul integrat, am conectat doua dintre iesiri la pinii DATA ai mufei PS2, iar celelalte 2 iesiri au fost conectate la doi pini DATA ai mufei Pmod(mufa aflata in dotarea FPGA-ului ce ofera posibilitatea conectarii unor extensii).



In majoritatea cazurilor semnalele nu vor fi sincronizate cu semnalul de tact al ceasului asa ca este necesara introducerea a doua variabile care sa decaleze semnalul de cuadratura si sa il sincronizeze cu ceasul.

Figura de mai sus reprezinta punctul de plecare in procesul de proiectare comportamentale, figura in care sunt descrise in mod grafic toate variabilele respectiv functiile ce vor fi utilizate ulterior in codul sursa.

Analizand numarul de secvente ce au loc intr-un ciclu de trecere de la din momentul in care cei doi emitori ai fototranzistoarelor pentru o anumita axa se afla in dreptul unui gol ,pana in momentul in care ajung in dreptul unui „dinte". Masurand raza rotitei si calculand lungimea sa putem deduce ce distanta reprezinta fiecare ciclcu prezentat anterior. De asemenea utilizand un divizor de frecventa putem calcula timpul necesar unui ciclu, si cunoscand lungimea acestuia putem calcula chiar si viteza de deplasare.

6 Codul sursa al procesorului de semnal

In realizarea aplicatiei s-a folosit softul Xilinx numit ISE Web Pack.S-a preferat utilizarea unui limbaj descriptiv(in cazul de fata Verilog) in locul proiectarii cu circuite basaculante bistabile.Iata codul sursa:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Tudor Constantinescu
//
// Create Date:    19:38:38 04/05/2010
// Design Name:
// Module Name:    quad_decoder
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module quad_decoder(clk,quad_Ax, quad_Bx,quad_Ay, quad_By, countx, county);
input clk,quad_Ax,quad_Bx,quad_Ay,quad_By;
output countx;
output county;
reg [2:0] quadAx_delay,quadBx_delay, quadAy_delay, quadBy_delay;
always @(posedge clk) quadAx_delay <= {quadAx_delay[1:0], quad_Ax};
always @(posedge clk) quadBx_delay <= {quadBx_delay[1:0], quad_Bx};
always @(posedge clk) quadAy_delay <= {quadAy_delay[1:0], quad_Ay};
always @(posedge clk) quadBy_delay <= {quadBy_delay[1:0], quad_By};

wire count_enablex=quadAx_delay[1]^quadAx_delay[2]^quadBx_delay[1]^quadBx_delay[2];
wire count_enabley=quadAy_delay[1]^quadAy_delay[2]^quadBy_delay[1]^quadBy_delay[2];
wire count_directionx=quadAx_delay[1]^quadBx_delay[2];
wire count_directiony=quadAy_delay[1]^quadBy_delay[2];
```

```
reg [7:0] countx;
reg [7:0] county;
always @(posedge clk)
begin
  if(count_enablex)
  begin
    if(count_directionx) countx<=countx+1;
   else
   countx<=countx-1;
  end
  else if(count_enabley)
  begin
  if(count_directiony) county<=county+1;
  else
  county<=county-1;
  end
 end
endmodule
```

Bibliografie

[1] situl producatorului, http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,789&Prod=NEXYS2 .
[2] www.colinhafey.com
[3] www.national.com
[4] Richard Lyons, Quadrature signals:complex but not complicated.
[5] www.dpsguru.com
[6] Simona Halunga Fratu, Transmisii analogice si digitale
[7] www.leobodnar.com

TUDOR CONSTANTINESCU
Universitatea Politehnica Bucuresti,
Facultatea de Inginerie in Limbi Straine
Splaiul Independentei 313,
Bucuresti
ROMANIA
cttudor_kobe@yahoo.com

Sesiunea Națională de Comunicări Științifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

FPGA song composer

Alexandra Elena DRAGOMIR
Coordonator: Asist. Univ. Ing. Alexandru RADOVICI

Abstract

Do you want it to sound like Beethoven symphonies or simply play the notes, the major scale? An FPGA can do both. And in as simply way too: it uses a simple square wave with frequency derived from its own system clock.

1 Introducere

Programarea structurilor hardware reprogramabile de tip FPGA poate fi, de cele mai multe ori, statica. Prin lucrarea de fata am vrut sa ofer acea pare „fun", jucandu-ma astfel la nivel de programare harware simuland sirene de amubulanta si politie pana la redarea unei melodii pornind de la o tabulatura. Jongland printre sintaxe de cod sursa HDL Verilog si VHDL, mi-„am invatat" circuitul integrat digital configurabil, FPGA-ul, sa-si imparta frecventa memorand frecventele notelor si astfel sa-„mi cante" cum imi place mie.

2 FPGA

2.1 Ce este un FPGA

Un FPGA (Field Programmable Gate Array) este un circuit integrat digital configurabil alcatuit dintr-o matrice de CLB-uri (Configurable Logic Blocks) si bistabile(FF - flip flop).

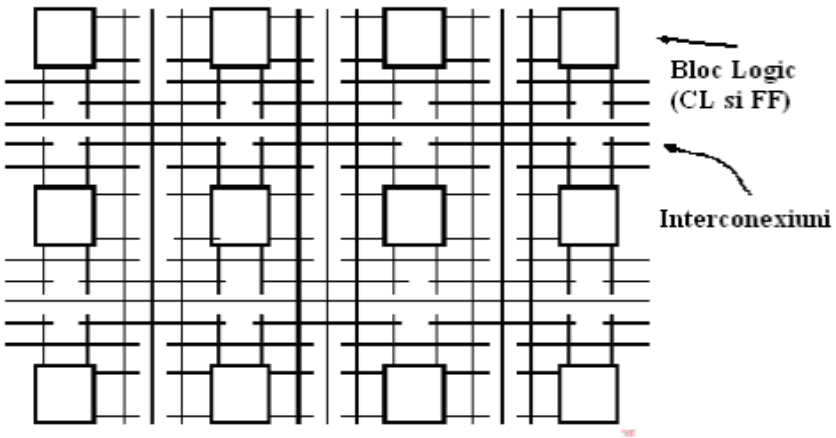


Fig. 1. Schema simplificata a arhitecturii unui FPGA

Un bloc logic FPGA clasic este alcătuit dintr-un tabel de căutare cu 4 intrări, un flip-flop şi un multiplexor are selectează fie ieşirea tabelului de căutare, fie ieşirea sincronizată a acestuia (trecută prin flip-flop).

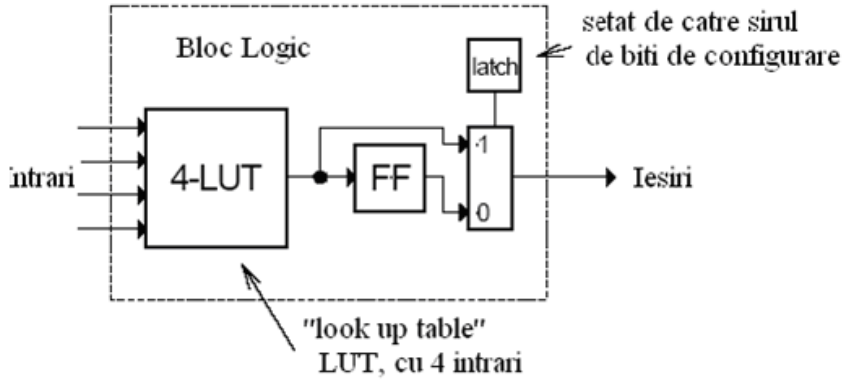


Fig.2. Bloc logic simplificat

Fiind o retea de porti logice programabile, putem programa FPGA-ul pentru aproape orice functie digitala. Iata pasii urmariti cand se lucreaza cu un FPGA:

- se creaza functia logica dorita intr-un soft oferit o data cu placuta (pentru FPGA-ul meu, Spartan3E-500 FG320, softul utilizat a fost Xilinx ISE) plecand de la scheme ce genereaza automat codul sau scriind direct cod;
- se conecteaza un cablu de la FPGA la calculator incarcand fisierul binar si se asteapta ca FPGA-ul sa afiseze rezultatele functiei create.

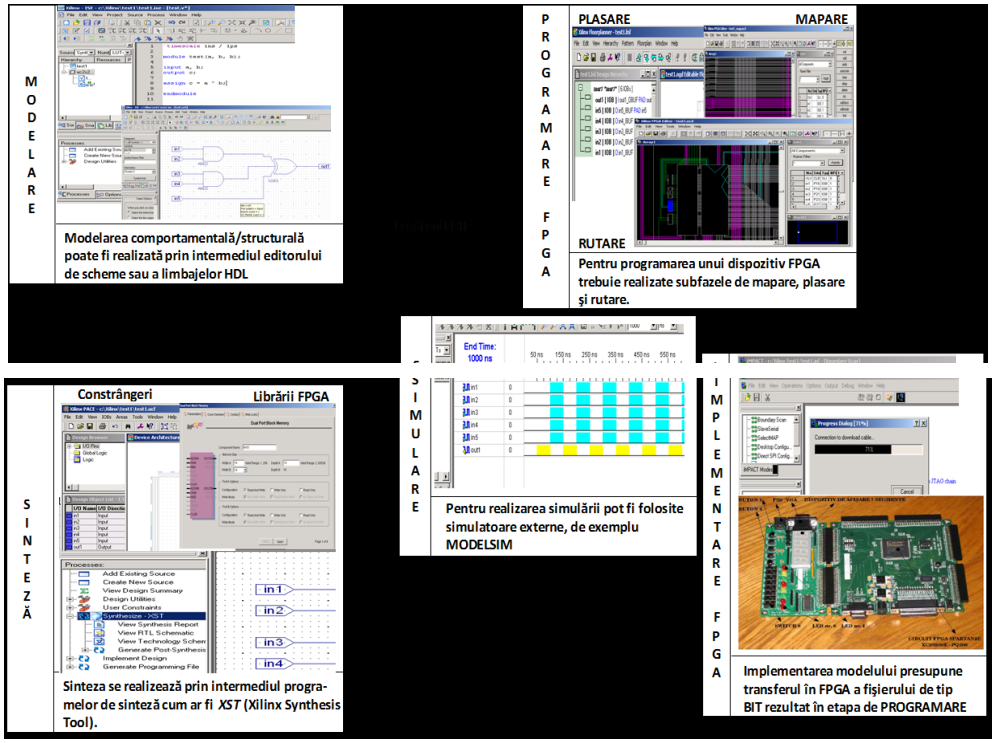


Fig. 3. Fluxul proiectarii cu FPGA

2.1.1 Cine face FPGA-uri?

Cele mai mari companii producatoare de FPGA-uri in toata lumea sunt: Xilinx si Altera.

Xilinx (fig. 4. a), care a inventat FPGA-ul, utilizeaza trasee de interconectare care inconjoara fiecare CLB, la intersectiile traseelor fiind amplasate blocuri de comutatoare (BC) organizate sub forma de matrici, formate din tranzistoare de trecere, controlate pe grila cu ajutorul unor celule de memorie SRAM (fig. 4. b). Celulele de memorie SRAM (fig. 4. c) sunt utilizate si pentru stabilirea functiilor logice implementate de catre multiplexoarele care implementeaza tablourile asociative.

Filosofia generala Xilinx este de a oferi:

- cele mai mari si mai flexibile dispozitive (facilitate completa);
- arhitectura complexa, dispozitive puternice.

Actel propune mai multe segmente de fire avand orientare orizontala fata de cele cu orientare verticala. Astfel, filosofia acestei companii fiind sa pastreze usurinta utilizarii dispozitivelor de catre utilizatori.

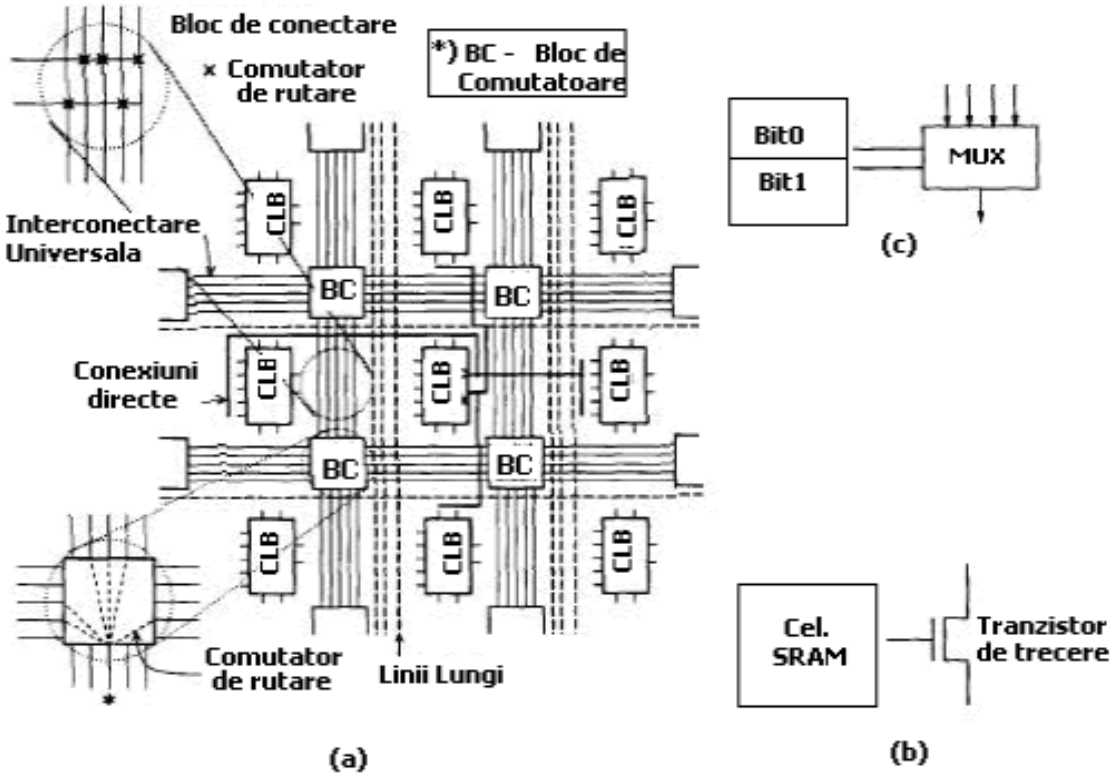


Fig. 4. Arhitectura de interconectare Xilinx

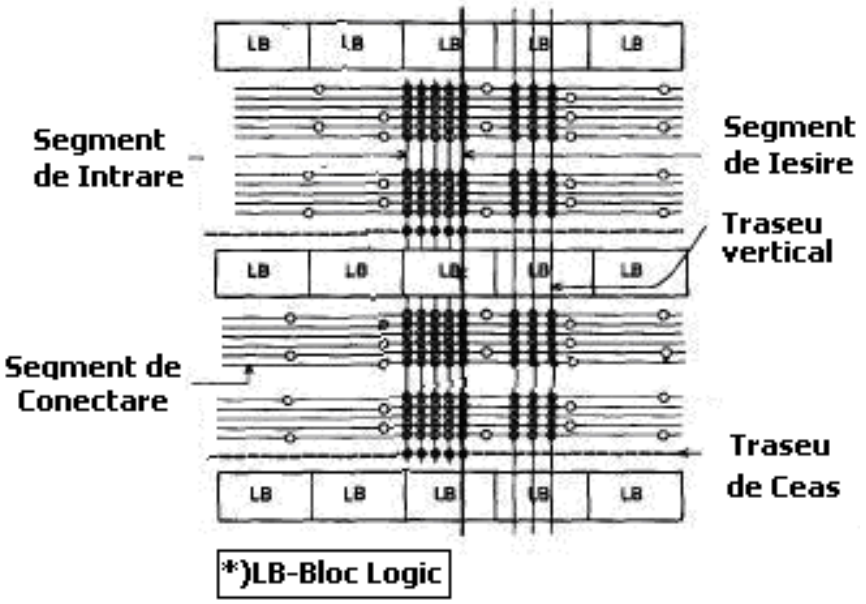
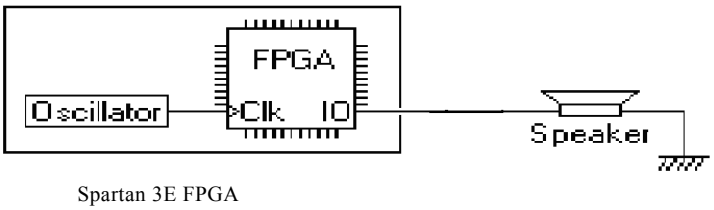


Fig. 5. Arhitectura de interconectare Actel

3 HDL design

Pentru proiectul de fata, am folosit, pe langa placuta si un speaker.



Spartan 3E FPGA

Fig. 6. Reprezentare schematica

Nexys2 bord include un oscilator 50MHz si un socket pentru un al doilea oscilator. Semnale de ceas din oscilatoare sunt conectate la pinii ceasului de intrare pe FPGA astfel încât să poată conduce blocurile de ceas disponibile si sintetizate în FPGA. Sintetizatoarele de ceas (numite DLL sau delay locked loops) asigura capabilităţi care includ amplificari ale frecventei de intrare, chiar divizari ale frecventei de intrare la orice numar intreg şi definiri precise ale fazelor intre diferite semnale de ceas.

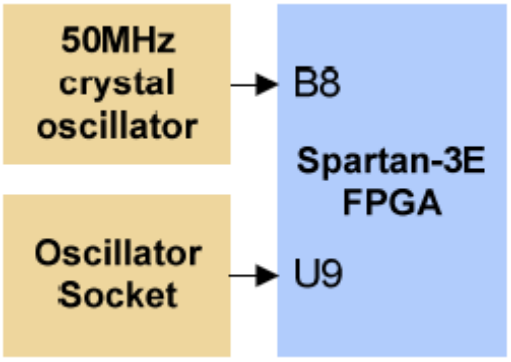


Fig. 7. Nexys2 clocks

In cazul proiectului, frecventele modulate pentru a atinge anumite note, A, A#, B, C, C#, D, D#, E, F, F#, G si G#, sunt purtate catre iesirea IO a FPGA-ului care mai apoi ajung la speaker si ulterior la urechile noastre.

3.1.1 Simplul „beep”

FPGA-ul poate implementa foarte usor un numarator binar in Verilog (Hardware Description Language -HDL). Pentru un simplu beep, folosesc un numarator pe 16 biti. Asta inseamna ca numaratorul va incepe de la 0 pana la pozitia 2^16-1 (65535), adica 65536 valori posibile. Voi folosi acest numarator pentru a imparti frecventa de 50 MHz. Astfel, MSB (most segnificant bit) numaratorului va avea o frecventa de 762 Hz (50MHz/65536).

Codul sursa arata asa:

```
module beep(clk, speaker);
input clk;
output speaker;

// numarator pe 16 biti
reg [15:0] counter;
always @(posedge clk) counter <= counter+1;

// assignez MSB
assign speaker = counter[15];

endmodule
```

LSB (less significant bit), adica counter[0], va avea o frecventa de 25 MHz, counter[1] o frecventa de 12.25 MHz si asa mai departe. Un „frumos” semnal de 762 Hz se va auzi din output-speak-urului.

Daca dorim un semnal de o anumita frecventa, acesta este descris foarte usor. De exemplu, eu am vrut un semnal de 440 Hz, echivalentul notei „A”. In loc sa impart 50 MHz la 65536, o sa-l impart la 113636. Am adaugat un parametru „clkdivider” pentru a imparti output-ul la 2 si astfel a avea un 50 % **duty cycle** (the ratio of time at which one pulse of a clock remain logic high to its period).

```
module nota_A(clk, speaker);

input clk;

output speaker;

parameter clkdivider = 50000000/440/2;
```

```
reg [14:0] counter;
always @(posedge clk) if(counter==0) counter <= clkdivider-1; else counter <= counter-1;
reg speaker;
always @(posedge clk) if(counter==0) speaker <= ~speaker;
endmodule
```

Note	Frequency (Hz)	Wavelength (cm)
A	440.00	78.4
$A^{\#}$	466.16	74.0
B	493.88	69.9
C	523.25	65.9
$C^{\#}$	554.37	62.2
D	587.33	58.7
$D^{\#}$	622.25	55.4
E	659.26	52.3
F	698.46	49.4
$F^{\#}$	739.99	46.6
G	783.99	44.0
$G^{\#}$	830.61	41.5

Fig. 8.

Trebuia sa fac sa alterneze 2 tonuri, asa ca de data aceasta am folosit un numarator pe 24 de biti pentru a produce un semnal dreptunghiular. MSB bit, adica tone[23], va avea o frecventa de aproximativ 3 Hz. Folosim acest bit pentru a schimba 2 frecvente ale aceluiasi numarator si sa alterneze intre 2 tonuri.

Demonstratie a unei sirene de ambulanta:

```
module ambulance(clk, speaker);
input clk;
output speaker;
parameter clkdivider = 50000000/440/2;
reg [23:0] tone;
always @(posedge clk) tone <= tone+1;
reg [14:0] counter;
always @(posedge clk) if(counter==0)
counter <= (tone[23] ? clkdivider-1 : clkdivider/2-1); else counter <= counter-1;
reg speaker;
always @(posedge clk) if(counter==0) speaker <= ~speaker;
```

```
endmodule
```

3.1.2 Cantarea notelor

Ideea mea a fost sa pot canta pe note un cantec si sa fiu in stare, plecand de la orice tabulatura, sa cant „hardware” cantecul respectiv. Cum s-a intamplat sa cant notele?

Pentru o nota am folosit 6 biti deci voi avea 64 de note (2^6). Sunt 12 note pe o octava, asa ca 64 de note mi-ar da 5 octave, destul pentru un mic cantecel.

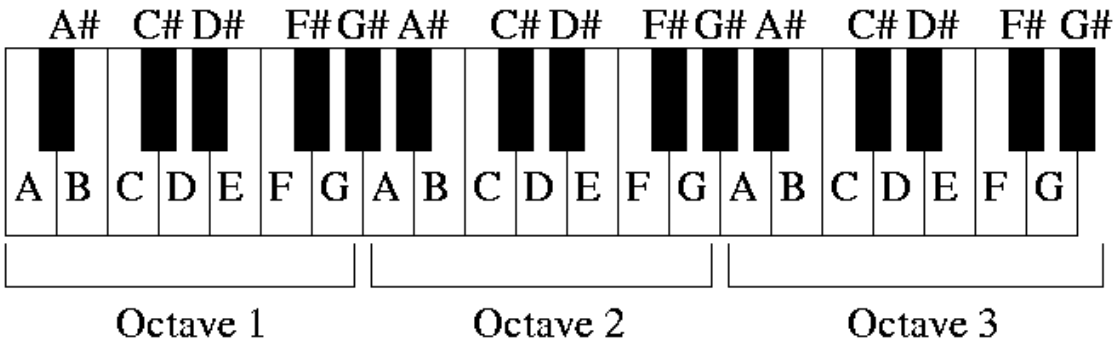


Fig. 9.

Pentru o suita de note, instantiez un numarator pe 28 biti de unde extrag the most segnificant 6 bits pentru nota pe care o vreau. Daca impartim printr- o valoare mai mica, obtinem o mai buna frecventa a notei.

```
module music(clk, speaker);

input clk;

output speaker;



reg [27:0] tone;

always @(posedge clk) tone <= tone+1;

wire [5:0] fullnote = tone[27:22];

wire [2:0] octave;

wire [3:0] note;

divide_by12 divby12(.numer(fullnote[5:0]), .quotient(octave), .remain(note));

reg [8:0] clkdivider;
```

```
always @(note)
case(note)
  0: clkdivider = 512-1; // A
  1: clkdivider = 483-1; // A#/Bb
  2: clkdivider = 456-1; // B
  3: clkdivider = 431-1; // C
  4: clkdivider = 406-1; // C#/Db
  5: clkdivider = 384-1; // D
  6: clkdivider = 362-1; // D#/Eb
  7: clkdivider = 342-1; // E
  8: clkdivider = 323-1; // F
  9: clkdivider = 304-1; // F#/Gb
  10: clkdivider = 287-1; // G
  11: clkdivider = 271-1; // G#/Ab
  12: clkdivider = 0; // nu ar trebui sa existe
  13: clkdivider = 0; // nu ar trebui sa existe
  14: clkdivider = 0; // nu ar trebui sa existe
  15: clkdivider = 0; // nu ar trebui sa existe
endcase

reg [8:0] counter_note;
always @(posedge clk) if(counter_note==0) counter_note <= clkdivider;
                                          else counter_note <= counter_note-1;
reg [7:0] counter_octave;
always @(posedge clk)
```

```
if(counter_note==0)

begin

 if(counter_octave==0)

  counter_octave <=
(octave==0?255:octave==1?127:octave==2?63:octave==3?31:octave==4?15:7);

 else

  counter_octave <= counter_octave-1;

end

reg speaker;

always @(posedge clk) if(counter_note==0 && counter_octave==0) speaker <= ~speaker;

endmodule
```

4 Posibilitati de imbunatatire

Pe viitor as dori punere în aplicare a unui sistem multi-ton electronic folosind o tastatura PS/2 si un sistem audio. Figura 10 reprezinta intocmai ideea mea.

As dori sa folosesc tastatura PS/2 ca si o tastatura de pian iar placuta ar folosi ca sintetizator System on Chip (SOC) pentru a genera muzica si sunete. Monitorul VGA conectat la placa de dezvoltare ar arata ce nota/tasta este „jucata” in timpul melodiei.

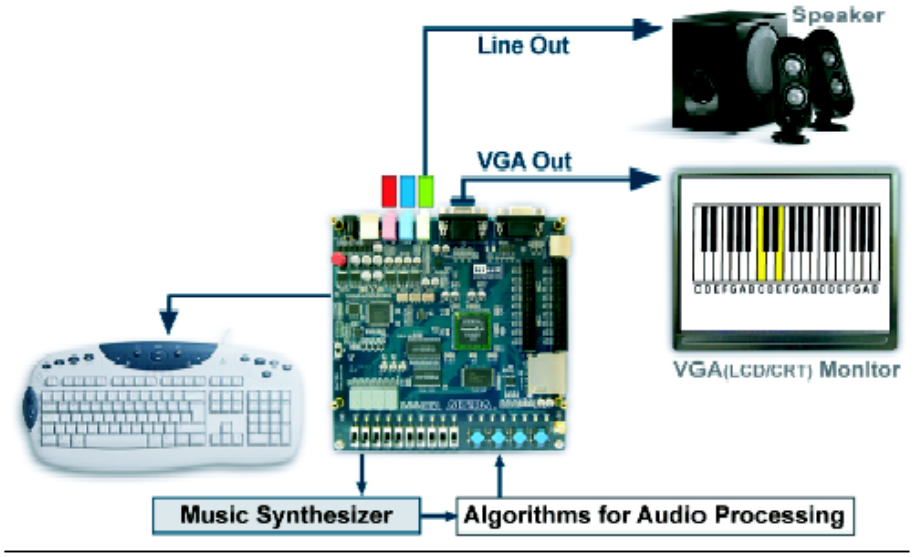


Fig. 10. Music synthesizerdemonstration setup

Bibliografie

[1] ***, *Getting Started With the NEXYS2 Spartan 3E Kit and Xilinx ISE Webpack A Beginner's Tutorial*, http://www.echelonembedded.com/
[2] www.fpga4fun.com
[3] www.altera.com
[4] www.csit-sun.pub.ro
[5] http://esd.cs.ucr.edu/labs/music/music.html
[6] http://www.asic-world.com/verilog/veritut.html
[7] Digilent Inc., *Digilent Nexys2 Board Reference Manual*,http://www.digilentinc.com
[8] Xilinix, http://www.xilinx.com

DRAGOMIR Alexandra Elena
Universitatea Politehnica Bucuresti
Facultatea de Inginerie in Limbi Straine (FILS)
Electronica aplicata
Splaiul Independentei, 290
ROMANIA
E-mail: puttel_chermir@yahoo.com

Sesiunea Naţională de Comunicări Ştiinţifice a Studenţilor
"Imaginaţie, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Human Brain Capillary Recognition

Florin-Robert Drobotă
Vlad Monescu

Abstract

The modern medicine uses more and more the new technologies, hardware and software. Challenged by this human – computer connection, in which the computer helps humans to self-understand his nature, I chose this theme of recognizing the capillaries from an given image using image processing techniques and modern programming languages and technologies. The combination of algorithm programming and user interfaces that works with real digital images that were taken from real human bodies have that practical element that makes you feel you are working on something important and maybe sometime it will save lives. Until then this project comes with features in speed and accuracy in next eventual studies. The main idea of this project started behind the analysis of the medic team Dr. Ioana Dumitru and Dr. Sebastian Toma within the neurology department of Brasov's Judetean Hospital, who decided that it is necessary to make a software product that could analyze high definition digital images of microscopically tissue of brain, and then identify, count, and extract information for helping them into their research.

1 Context — date teoretice

1.1 Introducere

Pentru a înţelege mai bine care este legătura dintre anatomie şi calculator, trebuie să înţelegem problema. În medicina există o ramură numită neurologie ce se ocupă cu diagnosticul şi tratamentul bolilor organice care afectează sistemul nervos central sau periferic. Structurile organice ce ţin de domeniul neurologiei sunt - pe de o parte - creierul, măduva spinării (reprezintă sistemul nervos central), structurile înconjurătoare, precum şi vasele sanguine care le hrănesc, - pe de altă parte - nervii cranieni, rădăcinile nervoase şi ganglionii spinali, nervii periferici, după ieşirea din canalul spinal, inclusiv legăturile cu muşchii scheletici (reprezintă sistemul nervos periferic).

Deci raza de acţiune a bolilor neurologice este situată între creier, măduva spinării şi nervi periferici. De-a lungul timpul s-a încercat tratarea acestor boli, care de obicei duc la invaliditate(totală sau parţială), dar fără rezultate curative. Aşadar evoluţia medicinei, mai precis a neurologiei, este în continuă expansiune. De aceea este loc pentru noi cercetări în acest domeniu. Statisticile arată că "*accidentele vasculare cerebrale reprezintă a treia cauză de mortalitate în lume, după bolile cardiovasculare şi cancer şi cea mai gravă cauză de invaliditate, la persoanele de vârstă activa*", preciza prof. univ. dr. Marcel Pereanu, şeful Clinicii de Neurologie, din cadrul Spitalului Clinic Judeţean Sibiu.

1.2 Accident vascular cerebral

Un accident vascular cerebral(`AVC`) apare atunci când un vas de sânge (o arteră) care furnizează sânge la nivelul unei zone a creierului(prin capilare) se sparge sau este blocat de un cheag sangvin. În câteva

minute, celulele nervoase din acea zonă sunt afectate şi ele pot muri în câteva ore. Ca rezultat, acea parte a corpului care este controlată de zona afectată a creierului nu mai poate funcţiona adecvat. Dacă o leziune vasculară cerebrală îşi menţine efectele mai mult de 24 de ore, atunci atacul vascular cerebral este confirmat. În cazul în care apar simptome ale unui AVC este necesar un tratament de urgenţă, exact ca şi în cazurile de infarct miocardic. În cazul în care tratamentul medical este început cât mai curând după apariţia simptomelor, cu atât mai puţine celule nervoase vor fi afectate permanent.

1.3 Investigaţii

Timpul este critic în diagnosticarea unui accident vascular cerebral. Un diagnostic pus rapid poate duce la administrarea de medicamente care asigură o recuperare mai bună. Prima prioritate va fi să se determine dacă accidentul vascular cerebral este ischemic sau hemoragic. Această distincţie este critică deoarece medicamentele administrate pentru un AVC ischemic (cauzat de un cheag de sânge) poate fi ameninţătoare de viaţă dacă accidentul vascular cerebral este hemoragic (cauzat de o sângerare). Doctorul va trebui de asemenea să ia în considerare şi alte afecţiuni care pot da simptome asemănătoare unui AVC şi să vadă dacă există complicaţii.

Observaţie Dezavantajul acestor investigaţii este că se fac după ce pacientul a suferit atacul vascular cerebral, iar acurateţea localizării exacte a vasului lezat nu este atât de mare. Având în vedere că timpul este critic în asemenea situaţii, descoperiri ce ar putea asocia simptomele cu locul AVC la nivel atomic, ar fi de un real folos.

1.4 Studiul

Aproximativ 5% din accidentele vasculare cerebrale implică diencefalul. Din acestea 58% afectează bărbaţii iar 42% femeile(diferenţele între sexe nu sunt semnificative statistic deoarece sunt aproape egale), iar vârsta medie este 61,5 ani. Sindroame clinice asociate au fost clasificate în sindroame cu deficit hemisenzorial, hemiataxie şi mişcări involuntare, sindroame cu simptomatologie senzitivă pură şi sindroame senzitivo-motorii, secundare. Acestea sunt sindroame pure talamice adică nu mai au altă leziune a creierului, deci se poate face o corelaţie directă între simptome(ceea ce descrie pacientul), semne(ceea ce găseşte doctorul patologic când îl examinează), sindrom(semne + simptome) şi locul leziunii.

Prin realizarea unui studiu pur ştiinţific pe pacienţi decedaţi care au prezentat sindroamele clinice asociate mai sus se poate determina dacă există o corelaţie între numărul de capilare prezente la baza talamusului şi tipul sindromului.

Astfel datele de intrare sunt piese(*lame*) de la baza talamusului (prelucrate) ce aparţin pacienţilor decedaţi în urma unui atac vascular cerebral, cărora li se cunosc sindroamele. Dacă se determină numărul de capilare asociate talamusului, apoi se poate stabili dacă există o corespondenţă între pacienţii cu un anumit tip de sindrom şi un număr aproximativ egal de capilare la un anumit nivel atomic, care e dat de numărul lamei.

Lama este o "felie" de mici dimensiuni la nivelul creierului şi se obţine prin prelevare pieselor, fixarea lor în formol 10%. Piesele formate se vor prelucra prin tehnica includerii la parafină, apoi prin tehnica colorării cu hematoxilină-eozină, şi alte tehnici specifice de colorare. Aceste lame se pot fotografia (transformate în imagini digitale) cu ajutorul unor dispozitive speciale asemănătoare microscoapelor ce au ataşate aparate de fotografiat de rezoluţii înalte. Ea poate conţine mai multe elemente, specifice acelui nivel atomic precum celulă glială , capilară şi neuroni.

Odată ce obiectivele au fost stabilite şi materialele/datele de intrare au fost pregătite, totul se rezumă la crearea aplicaţiei.

2 Aplicaţia — date tehnice

2.1 Introducere

Din considerente practice am ales platforma .NET şi limbajul C# ce s-au dovedit a fi de un real folos, îmbunătăţind considerabil rapiditatea dezvoltării proiectului dar şi a timpilor de rulare. Aşadar plecând de la un `Windows Form`, s-a realizat un program simplu de lucru cu imagini. Printr-un `OpenDialogBox` se poate deschide orice imagine în format `.jpg`, `.png`, `.tif`, `.bmp`, `.gif` apoi afişate în `PictureBox`-ul aplicaţiei.

2.2 Strategii de dezvoltare

Din considerente de optimizare s-a adoptat o strategie de analizare a imaginilor bazata pe stratificare. Iniţial imaginea va fi divizată cromatic în două noi imagini. Prima imagine v-a conţine doar petele întunecate printre care şi elementele sangvine, iar a doua va conţine numai petele deschise printre care şi peretii capilarelor în sine. Aparent imaginile nou create nu aduc nici o îmbunătăţire, dar împreuna pot face ca această căutare să fie una informată(în strategiile de căutare informată, se utilizează cunoştinţe specifice problemei pentru a găsi soluţiile în mod eficient). Având la dispoziţie coordonatele elementelor sangvine dar nu numai (în urma diferenţierii între pixeli sunt validaţi şi pixelii care nu fac parte din capilare) se poate verifica mai rapid în a doua imagine dacă în jurul coordonatelor găsite se află pixeli validaţi ca făcând parte din capilara în sine. Astfel căutarea se face doar în jurul elementelor care "merită efortul". Timpul de analizare a unei imagini scade semnificativ, pentru o imagine de mici dimensiuni procesarea se face sub 2 secunde.

2.3 Proceduri

S-a făcut referire până acum la clasa care se ocupă de prelucrarea imaginilor. Ea se numeşte `ProcesareLame`, şi conţine metode variate ce vor fi detaliate în următoarele subcapitole.

2.3.1 AlbSauNegru

`AlbSauNegru` are rolul de a returna o imagine formată prin transformarea imagini primite într-una `alb-negru`, mai exact într-o imagine `binară`. Transformarea se produce folosind o filtrare gen `alb sau negru`. Astfel pixelii ce au nuanţele `Red`, `Green` şi `Blue` incluse în intervalele definite de variabilele `IntRange`, se vor transforma în pixeli albi(`R:255,G:255,B:255`), iar pixelii care au nuanţe ce nu aparţin intervalelor `IntRange` devin pixeli negrii(`R:0,G:0,B:0`). Imaginea modificată nu este cea originală ci o clonă, care va fi ulterior trimisă ca rezultat al metodei.

În prima parte se creează o variabilă de tip `Color` numită `currentColor`, pentru a putea fi utilizată ulterior. Se creează apoi o clonă a imaginii primite ca parametru pentru a i se putea face modificări. Sunt create două `for`-uri ce parcurg toata matricea de pixeli ai imaginii clonate, începând de la coordonatele `X=0` şi `Y=0`. La fiecare pixel i se capturează culoarea în variabila `currentColor`, folosind metoda `GetPixel(X,Y)` a imaginii clonate. Odată stabilită culoarea pixelului, adică nuanţele ce formează culoarea, se verifică printr-o instrucţiune `if` dacă acestea aparţin intervalelor primite ca parametri. Dacă aparţin, atunci se apelează metoda `SetPixel(X, Y, Color.White)` pentru imaginea nou creată, dacă vreuna din nuanţe nu aparţine intervalului corespunzător atunci se apelează aceeaşi metodă dar cu un parametru diferit: `SetPixel(X, Y, Color.Black)`.

Imaginea clonată este iniţial identică cu cea originală, dar după filtrarea pixelilor în albi sau negrii, imaginea conţine doar 2 culori, dar ea este totuşi în formatul imaginii originale. Acest lucru trebuie corectat, deci convertirea imaginii în format `Format8bppIndexed`. În transformarea imaginilor color în imagini alb-negru, mai precis, în imagini cu nuanţe variate între alb şi negru(scala gri) se foloseşte un filtru `Grayscale` de parametrii 0.2125 , 0.7154 , 0.0721 ce are la bază algoritmul consacrat BT709 la o frecvenţă mult mai mare. Formula de calcul a culorii unui pixel transformat prin `GrayscaleBT709` este:

$$y = R * 0.2125 + G * 0.7154 + B * 0.0721 \quad (1)$$

unde `y` este noua nuanţă rezultată, iar `R`, `G` şi `B` sunt valorile nuanţelor roşu(`red`), verde(`green`) şi albastru deschis(`blue`) ai pixelului transformat.

Aşadar urmărind formula de calcul a algoritmului `GrayscaleBT709`, se poate deduce că un pixel de culoare neagră(RGB:255,255,255) din imaginea filtrată anterior va avea culoarea:

$$y = 0 * 0.2125 + 0 * 0.7154 + 0 * 0.0721 = 0 + 0 + 0 = 0 \tag{2}$$

ceea ce înseamnă că în noul format(`Format8bppIndexed`) pixelul va avea culoarea paletei cu valoarea 0, adică tot negru, iar dacă deducem şi culoarea unui pixel alb(RGB:0,0,0) vom obţine:

$$y = 255 * 0.2125 + 255 * 0.7154 + 255 * 0.0721 = 54,187 + 182,427 + 18,3855 = 255 \tag{3}$$

ceea ce înseamnă că în noul format(`Format8bppIndexed`) pixelul va avea culoarea paletei cu valoarea 255, adică tot alb.

Deci, în final se aplică filtrul `GrayscaleBT709` asupra imaginii clonate, rezultând imaginea necesară următoarelor prelucrări, aceasta fiind returnată ca valoare a procedurii `AlbSauNegru`.

2.3.2 Filtrare

În această procedură se va folosi o clasa a pachetului `AForge.NET` şi anume:

BlobsFiltering aparţine spaţiului de nume `AForge.Imaging.Filtering`. Această clasă are rolul de a detecta elementele **albe** dintr-o `imagine binară` şi apoi să le filtreze în funcţie de mărime. Această filtrare se face în funcţie de proprietăţile setate înainte de a fi aplicată. Aşadar proprietăţi ca `MinWidth`, `MinHeight`, `MaxWidth` şi `MaxHeight` trebuie completate pentru ca filtrarea să aibă succes. Ele de fapt stabilesc intervalul lăţimii şi cel al înălţimii între care un element este valid în urma filtrării. O altă proprietate importantă este `CoupledSizeFiltering` ce stabileşte dacă lăţimea **şi** înălţimea trebuie să fie incluse în interval, sau doar una din ele este de ajuns pentru ca elementul filtrat să fie valid. După setarea proprietăţilor, se poate aplica filtrul pentru imaginea dorită, iar ca răspuns se va returna imaginea filtrată. Se aplică filtrul `blobFilter` asupra imaginii primite ca parametru, iar rezultatul se returnează ca valoare a procedurii `Filtrare`.

2.3.3 Marchează

Procedura de faţă e creată pentru a marca printr-un cadran grafic elementele întâlnite în imaginea primită ca parametru `Bitmap target`, în imaginea văzută de utilizator primită tot printr-un parametru `sourceImage`. În această procedură se vor folosi clasele din `AForge.NET`:

BlobCounter face parte din spaţiu de nume `AForge.Imaging`. Clasa extrage obiecte independente dintr-o imagine binară, împreună cu imaginea lor şi alte informaţii utile, folosind algoritmul `connected components labeling`. Acest algoritm tratează toţi pixelii negrii ca fundal, nu ca un obiect. Asta înseamnă că toate obiectele localizate de algoritm trebuie să aibă altă culoare decât negru. Pentru căutarea `blob`-urilor(blob = obiect localizat de clasa `BlobCounter`), clasa suportă numai imagini în format `8 bpp indexed grayscale`. Din această cauză imaginile filtrate anterior trebuie convertite în formatul `Format8bppIndexed`.

Blob face parte din spaţiu de numele `AForge.Imaging`. Ea preia un obiect detectat de clasa `BlobCounter` şi încapsulează imaginea acestuia, informaţiile aferente lui şi poziţia în imaginea părinte.

La începutul procedurii `Marchează` se clonează imaginea `sourceImage` trimisă ca parametru, sub numele de `clonedImage`, pentru a nu altera imaginea iniţială. Apoi se iniţializează un obiect de tip `BlobCounter`, în care `Blob`-urile localizate vor fi ordonate prin setarea proprietăţii `ObjectOrder` cu atributul `BlobCounter.XY`, adică după poziţia lor în imaginea scanată.

Se apelează metoda `ProcessImage` a obiectului de tip `BlobCounter`, cu parametru imaginea `target`(imaginea binară, cea care nu este văzută de utilizator). În urma apelului acestei metode s-au localizat toate elementele albe din imaginea trimisă ca parametru. Pentru a avea acces la `Blob`-urile detectate se va crea un şir de obiecte `Blob`, numit `blobs`. Apoi `blobs` va primii colecţia de obiecte returnată de metoda `GetObjects` a clasei `BlobCounter` aplicată pe imaginea `target`. Având posibilitatea în acest moment de a manevra obiectele de tip `Blob` se creează o instrucţiune de tip `foreach` cu ajutorul căreia se va accesa fiecare blob din şirul `blobs`.

2.3.4 OnInteraction

Funcţie internă ce primeşte ca parametri: referinţă la imaginea ce trebuie modificată cu nume de parametru `ImageToModify`, două variabile `PointX` şi `PointY` de tip `int` corespunzătoare coordonatelor din imagine, unde utilizatorul a apăsat butonul mouse-ului. Funcţia va returna o valoare de tip `Boolean`, care va fi `true` dacă s-a efectuat cu succes modificarea şi false în cazul în care nu s-a efectuat nici o modificare deoarece utilizatorul nu a făcut clic pe o capilară marcată.

Se iniţializează un nou obiect de tip `BlobCounter`, apoi se setează proprietatea `ObjectsOrder` ca fiind sortare după poziţia `XY`. Se apelează metoda `ProcessImage` cu parametrul `ImageToModify`, apoi se creează un şir de `Blob`-uri, numit `blobs` ce va prelua rezultatul returnat de metoda `GetObject` aplicată obiectului `BlobCounter` având ca parametru imaginea `ImageToModify`, adică colecţia de `Blob`-uri. Apoi se cează o variabilă booleană `bloblOchit` iniţializată cu valoarea `false`, cu scopul de a o returna la finalul funcţiei. Urmează o instrucţiune `foreach` care parcurge toate elementele colecţiei `blobs` pentru a se verifica pe care din ele utilizatorul a făcut clic. Se testează printr-o instrucţiune `if` dacă utilizatorul a apăsat pe butonul mouse-ului cu cursorul deasupra unui `blob`. Adică se verifică dacă parametri `PointX` şi `PointY` definesc un punct în interiorul imaginii `Blob`-ului curent. Dacă utilizatorul nu a făcut clic pe un `Blob` atunci nu se va întâmpla nimic. Dacă el a făcut clic pe un `Blob`, atunci înseamnă că vrea să-l elimine, caz în care variabila `blobOchit` va lua valoarea `true`.

Se va apela din nou la o procedură de schimbare a formatului imaginii, deoarece pentru a şterge un element din imagine trebuie apelată metoda `SetPixel` a acesteia, metodă ce funcţionează numai pe imagini cu format neindexat, adică `Rgb`. Conversie se va face cu ajutorul clasei `GrayscaleToRGB` ce aparţine numelui de spaţiu `AForge.Imaging.Filters`. Aşadar se creează un obiect de tip `GrayscaleToRGB`, numit `backCol`, apoi se aplică conversia pe imaginea `ImageToModify`, deci vom obţine o imagine alb-negru, dar într-un format neindexat, `Rgb`.

Odată realizată conversia se va parcurge prin două instrucţiuni `for` imaginea `Blob`-ului curent şi se va seta culoarea neagră pentru toţi pixelii(folosind metoda `SetPixel` a imaginii `ImageToModify`.

După ce `Blob`-ul a fost eliminat, imaginea trebuie readusă la forma ei iniţială, adică în format `Format8bppRgb`. Se aplică imaginii `ImageToModify` un filtru te tip `GrayscaleBT709`, creat în prealabil. În final este returnată valoarea variabilei boolean-e `blobOchit`.

2.3.5 DrawBlob

Metodă ce vine în rezolvarea problemei de corecţie – "Adaugă capilară". Pentru ca utilizatorul să adauge o capilară va trebui, ca şi în cazul funcţiei `OnInteraction` să se opereze pe imaginea creată în scopul modificărilor.

Procedura `DrawBlob` este o procedură `internal` ce nu returnează nimic. Ea primeşte ca parametri referinţă la imaginea `Bitmap ImageToModify`, două valori de tip `int`(`xCenter` şi `yCenter`) asociate coordonatelor unde utilizatorul a făcut clic în cadrul imaginii şi al patrulea parametru este `raza`, adică valoarea de tip `int` a razei cercului ce trebuie "desenat". Prin urmare va trebui să se realizeze un cerc de rază `raza`, cu centrul în punctul de coordonate `xCenter`, `yCenter`.

Iniţial se face convertirea imaginii `ImageToModify` în format `Rgb`. Se creează o instanţă a filtrului `GrayscaleToRGB`, apoi se aplică pe imaginea `ImageToModify`.

Pentru crearea unui cerc se va folosi următorul algoritm. se creză o variabilă de tip `int` ce va lua valoarea `raza`^{2}, apoi printr-o instrucţiune `for` se vor parcurge 2*`raza`+ iteraţii, plecând de la valoarea -`raza` până la valoarea `raza`, cu pasul de incrementare de 1. Pentru fiecare iteraţie se va calcula variabila

nou creată de tip `int`, numită `y`, astfel:

$$y = \sqrt{r^2 - x^2} + 0.5 \tag{4}$$

unde `r` este valoarea variabilei `raza`, iar `x` este indexul instrucţiunii `for`.

După ce a fost calculată valoarea variabilei `y`, se apelează metoda `SetPixel` a imaginii `ImageToModify` de două ori: odată cu parametri `xCenter + x`, `yCenter + y` şi `Color.White`, iar a doua oară, cu aceeaşi parametri, mai puţin cel al coordonatei `y` care va fi `yCenter - y`.

În final procedurii, se creează filtrul `GrayscaleBT709` şi este aplicat imaginii `ImageToModify`, astfel se realizează conversia inversă.

2.3.6 ShowDrawingBlob

Această procedură diferă doar în partea de declarare faţă de procedura anterioară. Diferenţele constau în cum preiau parametri, şi ce valori returnează. Deoarece este procedura care se ocupă cu afişarea capilarelor nou create în `pictureBox`-ul utilizatorului, trebuie evitată modificarea imaginii `sourceImage`, de aceea ea nu este trimisă ca parametru referinţă, ci doar valoarea ei(imaginea). Din această cauză, procedura `ShowDrawingBlob` trebuie să întoarcă un rezultat, acela al imaginii `sourceImage` modificată.

2.3.7 ValidareCorectare

Este o procedură internă ce returnează o variabilă de tip `Bitmap`. Poate lua ca parametri două imagini numite `peteInchise` şi `peteDeschise`, iar ultimul parametru este de tip `int` denumit `Probabilitate`.

`ValidareaCorectare` are unul din rolurile de a valida elementelor din imaginea `peteInchise`, ce au în jurul lor un anumit procentaj(valoarea variabilei `Probabilitate`) de pixeli albi, doar că verificarea se face în cadrul imaginii `peteDeschise`. Apoi "corectarea" înseamnă că acele elemente care nu au un procentaj de pixeli albi în jurul lor, vor fi corectate, adică eliminate.

Iniţial se creează o imagine de tip `Bitmap`, numită `returnedImage` căreia îi este atribuită clona imaginii `peteInchise`. Apoi se creează o instanţă a clasei `BlobCounter`, ca după aceea să se apeleze metoda `ObjectOrder` pentru a fi ordonate `Blob`-urile după poziţia ocupată în imaginea de unde provin. Se apelează metoda `ProccessImage` a aceleiaşi clase, cu parametrul `returnedImage`. Odată ce imaginea a fost procesată se creează un şir de obiecte de tip `Blob`, unde este stocată colecţia de `Blob`-uri rezultată în urma apelării metodei `GetObjects` a obiectului de tip `BlobCounter` pentru imaginea `returnedImage`.

Odată capturaţi `Blob`-ii, trebuie pregătită imaginea `returnedImage` pentru modificări, deci se creează un filtru de tip `GrayscaleToRGB` şi i se aplică acesteia, făcându-se astfel transformarea din formatul `Format8bppIndexed` în format neindexat, `Rgb`. Urmează o instrucţiune `foreach` ce va parcurge toate elementele `Blob` din colecţia capturată anterior.

În acest moment începe validarea propriu-zisă. Se iniţializează mai multe variabile de tip `int`: `whitePixelsCounter` va lua valoarea 0, adică variabila se ocupă de numărarea pixelilor albi, iar pentru acest `Blob` curent numărul de pixeli albi din jurul lui este iniţial 0, apoi două variabile numite `searchWidthCoef` şi `searchHeightCoef` ce sunt nişte coeficienţi care ajută la determinarea suprafeţei din jurul elementelor din imaginea `peteInchise`, acestea luând valoarea jumătatea lăţimii blobului curent, respectiv jumătatea înălţimii, iar ultima variabilă iniţializată este `blobAreaSearch` ce determină aria suprafeţei de căutare, necesară pentru a se stabili procentajul pixelilor albi. `blobAreaSearch` se calculează cu formula:

$$blobAreaSearch = 2 * blob.Width * 2 * blob.Height = 4 * blob.Width * blob.Height \tag{5}$$

Această formulă este de fapt formula ariei dreptunghiului personalizată pentru cazul de faţă, unde lăţimea, respectiv înălţimea sunt calculate adăugând surplusul coeficienţilor lor. După calcularea suprafeţei ariei, trebuie număraţi pixelii albi din aceeaşi arie. Acest lucru se realizează prin intermediul a două instrucţiuni `for` imbricate, care parcurg toata aria de pixeli vizată. Aşadar primul for pleacă de la diferenţa coordonatei `X` a `Blob`-ului şi coeficientul `searchWidthCoef`, până la coordonata limită superioară a `Blob`-ului însumată cu acelaşi coeficient `searchWidthCoef`. Apoi se verifică, înaintea derulării celui de-al doilea `for`,

dacă nu cumva coordonata curentă a lui `X` depăşeşte limitele imagini. Numai în cazul în care această coordonată este în limitele imaginii se întră în al doilea `for`, dacă nu se continuă iteraţia `for`-ului precedent. Similar şi pentru coordonata `Y`, plecând de la diferenta coordonatei `Y` şi coeficientul `searchHeightCoef`, până la limita superioară a coordonatei `Y` a `Blob`-ului însumată cu acelaşi coeficient `searchHeightCoef`. Iar după instrucţiunea `for` se verifică din nou dacă coordonata `Y` nu a depăşit cadrul imaginii, caz în care se continuă iteraţia `for`-ului precedent.

Dacă totuşi se ajunge în blocul valid al ultimei instrucţiuni `if` însemnă că pixelul curent se află în imaginea `sourceImage` respectiv şi în `peteInchise` şi `peteDeschise`. Deci se poate face verificarea dacă pixelul curent găsit în `Blob`-ul ce aparţine imaginii `peteInchise`, este de culoare albă(`Color.White`) în acelaşi loc(la aceleaşi coordonate) în imaginea `peteDeschise`. Se face verificarea printr-o alta instrucţiune `if`, iar dacă se găseşte un pixel alb, este incrementată cu o unitate variabila `whitePixelsCounter`, ce se ocupă de numărarea pixelilor albi.

La părăsirea celor două `for`-uri, dar tot în cadrul `foreach`-ului, variabila `whitePixelsCounter` conţine numărul total de pixeli albi din `Blob`-ul curent. Pentru calculul procentajului "*cât la sută din numărul total de pixeli ai `Blob`-ului(asociat variabilei `blobAreaSearch`) reprezintă numărul de pixeli albi găsiţi (reprezentat de `whitePixelsCounter`)*" am folosit formula:

$$Procentaj = (whitePixelsCounter * 100)/blobAreaSearch) * 4 \quad (6)$$

unde `Procentajul` este un rezultat între 0% şi 100%, `whitePixelsCounter` este numărul de pixeli albi, iar `blobAreaSearch` este numărul total de pixeli. Acea înmulţire din final reprezintă calibrarea validării. Numărul "4" reprezintă coeficientul rezultat în urma calibrării validării pe o imagine creată în acest scop.

După ce a fost calculată valoarea `Procentaj`, printr-o nouă instrucţiune `if` care verifică dacă ea este mai mică decât variabila `Probabilitate` (cea setată de utilizator). Prin această validare se vor păstra doar `Blob`-urile care au valoare `Procentaj` mai mare ca valoarea variabilei `Proabbilitate`. Prin urmare vor intra pe ramura validă a instrucţiunii `if` doar `Blob`-urile care au un procentaj mai scăzut de pixeli albi decât este setat de utilizator, ceea ce înseamnă că el trebuie eliminat din imaginea ce va fi returnată(`returnedImage`).

Eliminarea `Blob`-ului se va face eliminând toţi pixelii albi ai acestuia. Deci folosind din nou două `for`-uri imbricate ce parcurg toată matricea de pixeli ai imaginii `Blob`-ului, ce face parte din imaginea părinte `peteInchise`. Se creează la fiecare iteraţie o variabilă de tip `Color` ce preia culoarea pixelului curent rezultat în urma apelării metodei `GetPixel` de coordonatele `X`(indexul primului `for`) şi `Y`(indexul celui de-al doilea `for`). Se testează printr-un nou `if` dacă culorile(codurile nuanţelor `RGB`) corespund, iar dacă acestea sunt egale pixelul este setat negru(`SetPixel(X, Y, Color.Black)`).

De specificat este că există posibilitatea de optimizare a acestei secvenţe ce elimină `Blob`-ulri, prin a refuza să se testeze fiecare pixel pentru a se verifica dacă el are culoarea alb. Se puteau alege varianta setării tuturor pixelilor din cadranul `Blob`-ului în culoarea negru. Principalul motiv pentru care nu am ales această formă de optimizare a procedurii, este securitatea imagini, deoarece există posibilitatea ca lângă acel `Blob` să existe unul foarte apropiat, iar prin setarea tutror pixelilor din cadranul `Blob`-ului, să fie setaţi şi pixeli din `Blobul alăturat`.

La sfârşitul procedurii se revine la formatul `Format8bppIndexed` prin aplicarea filtrului nou creat `GrayscaleBT709` pe imaginea `returnedImage`, care în final este returnează ca valoare a procedurii. După procedura `AlbSauNegru` şi `Filtrare`, imaginea va trece şi prin partea de validare.

2.3.8 GenerateResultFileFor

Procedura are rolul de a scoate toate informaţiile posibile dintr-o imagine procesată, validată şi corectată. Ea nu are nici un rol de editare/modificare a imaginii, ci doar o analizează. Parametrul procedurii este de tip `Bitmap`, numit `imageReadyForAnalyse`. Rolul procedurii nu este de a detecta dacă elementele din imagine sunt corecte, acest lucru trebuie realizat înaintea apelării procedurii. Informaţiile extrase în urma analizei sunt stocate într-un fişier cu aceeaşi denumire ca imaginea analizată, numai că tipul fişierului va fi de tip text(`.txt`).

Obiectivele procedurii sunt aflarea numărului total de capilare, aflarea coordonatelor fiecărei capilare din imaginea dată, aflarea diametrului fiecărei capilare în microni, aflarea suprafeţei fiecărei capilare în

microni^2 şi mm^2, determinarea numărului de capilare şi a suprafeţei totale pe zone(Nord-Vest, Sud-Vest, Nord-Est, Sud-Est), determinarea zonei cu cea mai mare suprafaţă, determinarea zonei cu numărul cel mai mare de capilare şi determinarea mediei diametrelor.

La începutul procedurii au loc iniţializări de componente. Se declară variabila constantă `OneInchInMM` de tip `double` ce are valoarea `25,4`. Această valoare reprezintă câţi milimetrii are un inch.

Se iniţializează un şir de patru variabile `double`, numit `SuprafeteZonale`, menit să reţină în fiecare element suprafaţa totală din zona respectivă asociat acestuia. Pentru a rezolva problema zonelor, s-a hotărât că indexul elementelor şirului să fie asociate zonelor astfel: 0–Nord-Vest, 1–Sud-Vest, 2–Nord-Est, 3–Sud-Est. Spre exemplu: dacă se găseşte o capilară în zona Nord-Est atunci se va adăuga valoarea suprafeţei acesteia la `SuprafeteZonale[2]`.

Similar se creează un şir de tip `int`, numit `ContorZonal`, tot cu patru elemente, iar după cum reiese şi din denumire, el numără capilarele pe zone. Spre exemplu: dacă se găseşte o capilară în zona Nord-Vest atunci se incrementează cu o unitate `SuprafeteZonale[0]`.

Urmează o secvenţă tipică de cod, în care sunt colectate toate `Blob`-urile găsite de metoda `ProcessImage` a clasei `BlobCounter`, având ca parametri imaginii `imageReadyToAnalyse` apoi extrase folosind metoda `GetObject` pe aceeaşi imagine, în şirul de obiecte de tip `Blob`.

Pentru a putea genera un fişier este nevoie de o variabilă de tip `TextWriter`, numită `GeneratedFile`, care în momentul creării se iniţializează cu un nou obiect de tip `StreamWriter` ce are ca parametri calea fişierului capturat de `openDialogBox`, numai că îi este modificată extensia în `txt`.

Combinaţia `TextWriter` – `StreamWriter` de fapt asociază fişierului specificat un obiect specializat în scrierea fişierelor text. Deci prin intermediul `GeneratedFile`, se pot adăuga texte în fişierul specificat. De precizat este că dacă fişierul nu există iniţial, el va fi creat, în caz contrar el va fi rescris.

Pentru a evita scrierea repetată în fişier pentru orice informaţie, se alege varianta de a se crea o variabilă temporară de tip `String` ce va fi încărcată pe parcursul procedurii cu anumite informaţii grupate, apoi scrise în fişier.

Pentru început se va scrie în fişier numărul total de capilare găsite în imagine apelând metoda `ObjectsCount` a obiectului `BlobCounter`.

Pentru calculul diametrelor şi a suprafeţelor, este necesară convertirea dimensiunilor din pixeli în microni. Pentru această operaţie sunt create mai multe variabile de tip `double`:
`ImageVerticalResolution` şi `ImageHorizontalResolution` reţin rezoluţia verticală, respectiv cea orizontală. `Px_per_squareMilimeter` este de fapt convertirea densităţii pixelilor pe inch^2 la densitatea pixelilor pe mm^2, conversie realizată folosind formula:

$$Ppsmm = vr \times hr/25,4^2; \tag{7}$$

unde
`Ppsmm` reprezintă valoarea lui `Px_per_squareMilimeter`, `vr` simbolizează `ImageVerticalResolution`, `hr` reprezintă `ImageHorizontalResolution`, iar `25,4` constanta `OneInchInMM`.

Se mai creează şi o variabilă de tip `int`, numită `AverageDiameterPerImage`, responsabilă cu stocarea tuturor diametrelor calculate pentru capilare, urmând ca la final să fie împărţită la numărul total de capilare pentru a afla media diametrelor.

În secvenţa următoare cu ajutorul instrucţiunii `foreach` se parcurg toate elementele şirului de `Blob`-uri, pentru fiecare fiind necesară determinarea suprafeţei şi al diametrului. Aşadar se iniţializează cu 0 variabila de tip `int` numită `PixSup` cu rolul de a număra pixelii albi ai `Blob`-ului curent. Prin două `for`-uri imbricate ce parcurg toţi pixelii imaginii `Blob`-ului curent se testează culoarea pixelului, iar dacă acesta este alb se incrementează cu o unitate valoarea lui `PixSup`. La ieşirea din cele două instrucţiuni `for` se cunosc numărul de pixeli albi ai `Blob`-ului, deci trebuie determinată suprafaţa. Aceasta va fi reţinută în variabila de tip `double`, numită `SuprafataReala`, calculată după formula:

$$SuprafataReala = PixSup/Px\_per\_squareMilimeter/factorZoom \tag{8}$$

unde valoarea variabilei `SuprafataReala` reprezintă suprafaţa capilarei calculată în mm^2, `Px_per_squareMilimeter` reprezintă densitatea de pixeli într-un mm^2, iar `factorZoom` este parametrul setat de utilizator, ce influenţează dimensiunile, denumit şi factor de mărire.

Observaţie Pentru determinarea diametrului unei capilare trebuie avut în considerare faptul că o capilară nu este neapărat de formă circulară, ceea ce înseamnă că nu are un diametru constant. De aceea se consideră diametrul capilarei media dintre lăţimea şi înălţimea acestuia.

Se calculează lăţimea (asociată variabilei `xDimension`, de tip `double`) şi înălţimea(asociată variabilei de tip `double`, numită `yDimension`) capilarei utilizând formulele:

$$\ell = \ell Pix/hr \times 25,4 \times 1000/factorZoom; \tag{9}$$

unde ℓ reprezintă lăţimea în microni(valoarea lui `xDimension`), ℓ `Pix` este lăţimea în pixeli, `hr` rezoluţia orizontală, `25,4` valoarea în milimetrii a unui `inch`, `1000` coeficient(1 milimetru = 1000 micrometrii) de transformare de la milimetrii la microni.

$$I = IPix/vr \times 25,4 \times 1000/factorZoom; \tag{10}$$

unde `I` reprezintă înălţimea în microni(valoarea lui `yDimension`), `IPix` este lăţimea în pixeli, iar `hr` rezoluţia verticală.

Odată aflate valorile lăţimii şi înălţimii se poate calcula, făcând media lor, valoarea diametrului aproximat `AverageDiameter`, variabilă de tip `double`.

Se introduc datele despre `Blob`-ul curent, formatate şi aliniate, în `string`-ul `toFileText`, apoi se scriu în fişier. De precizat este că suprafaţa în microni este calculată prin înmulţirea valorii `SuprafataReala` cu coeficientul de conversie de la milimetru^2 la micromentru^2 care este $1000000 = 10^6$.

Tot în cadrul `foreach`-ului se face reparţizarea `Blob`-ului curent în funcţie de poziţia sa în imaginea `imageReadyForAnalyse`. Pentru a determina corect poziţia elementelor din imagine, trebuie aflate coordonatele centrului lor.

Observaţie Coordonatele unui `Blob` înseamnă coordonatele punctului cel mai din stânga-sus a dreptunghiului în care se află `Blob`-ul. Se iniţializează două variabile de tip `int`: `blobCenterX` ia valoarea coordonatei `X` a `Blob`-ului însumată cu jumătate din lăţimea lui, respectiv `blobCenterY` ce ia valoarea coordonatei `Y` a `Blob`-ului însumată cu jumătate din înălţimea lui. Aceste două variabile reprezintă coordonatele centrului unui `Blob`.

Se verifică pe rând fiecare zonă(N-V, S-V, N-E, S-E) dacă include punctul de coordonatele determinate anterior. În momentul în care s-a găsit zona din care face parte centrul `Blob`-ului, se adaugă suprafaţa lui transformată în microni(înmulţită cu 10^6) la şirul `SuprafeteZonale` ce are ca index valoarea asociată acelei zone, şi se incrementează cu o unitate şirul `ContorZonal` cu acelaşi index.

În momentul părăsirii instrucţiunii `foreach` se pot scrie datele statistice în fişier. Aşadar se pot afişa într-un format organizat numărul de capilare, cât şi aria totală pe zone.

Pentru evidenţierea zonelor speciale din punct de vedere al vascularizării, se caută în şirurile anterioare valorile maxime, ca mai apoi să fie tipărite în fişier.

Se calculează şi media diametrelor, prin împărţirea valorii variabilei denumită semnificativ `AverageDiameterPerImage` la numărul capilarelor din imaginea curentă, apoi se scriu în fişier.

Salvarea datelor scrise cu ajutorul obiectului `GeneratedFile` în fişierul text se face numai la închiderea sesiunii de scriere, fapt realizat prin metoda `GeneratedFile.Close()`.

În final se afişează un mesaj ce confirmă că fişierul a fost generat, alături de numărul total de capilare detaliate.

3 Utilizare

3.1 Deschiderea unei imagini

Odată pornită aplicaţia, putem opta pentru două moduri concrete de a deschide o imagine. Fie folosim butonul rapid `Open...` de pe bara ajutătoare de sub bara de meniu, fie accesăm bara de meniu, iar în cadrul opţiunii `File` putem accesa `Open...`. Odată ajunşi în fereastra `Open...` putem deschide imagini în formate `.jpg`, `.png`, `.tif`, `.bmp` sau `.gif`. După ce am selectat imaginea dorită, şi aceasta a fost afişată putem începe analizarea imagini.

3.2 Capturare parametri

Pentru ca recunoaşterea capilarelor să fie posibilă, utilizatorul trebuie mai întâi să captureze nuanţele de culori necesare identificării unei capilare. Acestea sunt împărţite în nuanţe ce provin din elementele sangvine, care au o tentă închisă şi nuanţe ce provin din capilare, care tind spre alb. Pentru a captura nuanţele elementelor sangvine sau a capilarelor, utilizatorul trebuie să
folosească unealta de captură. Aceasta se găseşte în meniu la secţiunea `Parametri > Capturează culoare(element sangvin)` sau `Parametri > Capturează arie de culori(element sangvin)` sau `Parametri > Capturează culoare(capilară)` sau `Parametri > Capturează arie de culori(capilară)`

Aşadar selectând unul din modurile prezentate, utilizatorul va observa că cursorul mouse-ului a intrat în mod de captură, transformându-se în cruce/chenar, iar pe bara de comenzi rapide apar date utile privind coordonatele cursorului în imaginea deschisă: X, Y, paleta de culori (R, G, B) a pixelului peste care este cursorul mouse-ului în acel moment, precum şi zona în care se află. Datele se schimbă instant la fiecare mişcare de mouse. În acest moment utilizatorul poate face clic oriunde în imaginea deschisă pentru a captura paleta de culori (RGB-ul) pentru pixelii selectaţi. Se poate observa că, făcând clic, în partea dreaptă va apărea un mic container în care sunt afişate date legate de rezultatele capturilor. În cazul în care acest container deranjează vizibilitatea utilizatorului, el poate fi mutat prin procedura de `Drag&Drop`.

După ce au fost capturate nuanţele predominante reprezentative pentru elementele sangvine şi pentru capilare trebuie să setăm şi dimensiunile între care sunt admise atât elementele sangvine cât şi capilarele în sine. Acest lucru se poate face din panoul de parametri(detaliat în subcapitolul următor), dar în ajutorul completării corecte a datelor, utilizatorului i se pune la dispoziţie un mod de capturare a dimensiunilor. Folosind `Parametri > Capturează dimensiuni` se pot determina distanţe în cadrul imaginii deschise. Aşadar ţinând apăsat butonul stâng al mouse-ului şi trăgând un diametru imaginar vertical sau orizontal deasupra unei capilare, în momentul eliberării butonului stâng se va afişa distanţa capturată. Se încearcă capturarea dimensiunilor semnificative, în vederea stabilirii corectă a intervalului de măsurători, deci se urmăreşte capturarea celor mai mici dimensiuni, şi respectiv celor mai mari. Datele vor fi apoi introduse în panoul de parametri.

3.3 Parametri

Toţi parametri aplicaţiei se află într-un panou separat, numit `Parametri`. Acesta poate fi accesat din bara de meniu astfel: `Parametri > Arată`
`Parametri...` dar şi de pe bara de comenzi rapide făcând clic pe butonul cu *cheiţă*.

3.3.1 Parametri elemente sangvine şi Parametri capilare

Aceaste grupări au rolul de a reprezenta parametri elementelor sangvine şi a capilarelor. În prima parte sunt parametri ce ţin de paleta de culori între care este validat un pixel dintr-un element sangvin sau capilară. Datele sunt reprezentate pe 3 linii, fiind asociate cu paleta de culori minimă şi maximă(RGB-ul minim şi maxim). Aceste date sunt capturate cu ajutorul uneltelor prezentate în subcapitolul anterior, iar introducerea lor manuală nu este recomandată doar în cazul în care valorile parametrilor se cunosc.

În momentul salvării datelor, două blocuri ajutătoare afişează grafic culoarea pixelului format din paleta de culori minimă, respectiv cea a pixelului format din paleta de culori maximă.

În a doua parte, trebuie introduse datele ce ţin de intervalul dimensiunilor în care este valid un element sangvin sau o capilară. Conform schemelor ajutătoare trebuie introduse: lăţimea maximă, lăţimea minimă, înălţimea maximă, înălţimea minimă. Elementele sangvine, capilarele sau alte elemente ce depăşesc intervalele stabilite nu sunt luate în considerare.

În gruparea capilarelor mai există încă un parametru important în validarea corectă a capilarelor. `Procentaj aparitie` reprezintă procentajul rezultat din cât la sută din elementul sangvin este înconjurat de capilară în sine, mai exact, dacă în vecinătatea petelor închise(elementelor sangvine) se află un anumit număr de pixeli ce ţin de petele deschise(capilare).

3.3.2 Parametri generali

În cadrul parametrilor generali, există două opţiuni. Prima este `Factor de zoom` parametru utilizat în calcularea cât mai corectă a dimensiunilor. Toate datele generate, legate de dimensiuni, trebuie împărţite la factorul de zoom(sau factorul de mărire) produs de microscop şi aparatul de fotografiat.

Ultima opţiune în cadrul parametrilor este aceea de `Arată stadii intermediare`. Această opţiune ajută la corectarea greşelilor apărute în urma parametrilor eronaţi. Aşadar activând această opţiune, se pot urmării stadiile intermediare prin care trece imaginea în momentul prelucrărilor făcute de acţiuneaDetectează şi marchează ce va fi prezentată în cele ce urmează.

Observaţie Trebuie avut grijă ca după capturarea tuturor datelor, limita maximă a intervalului de paletă din cadrul elementelor sangvine să nu fie mai mare decât limita minimă a intervalului de paletă din cadrul parametrilor din cadrul capilarelor. Acest lucru duce la o analiză incorectă, respectiv la identificări incorecte.

3.4 Acţiuni

Toate metodele de procesare/editare a imaginilor se găsesc în bara de meniu la secţiunea `Acţiuni`. Acestea trebuie efectuate pentru a obţine rezultate optime într-o ordine. Iniţial se porneşte cu "Detectează şi marchează" care este principala metodă de analizare, detectare, recunoaştere a capilarelor, apoi "Corecţii" pentru cazul în care sunt erori de procesare. Ca rezultat a acestor acţiuni putem apela "Generează fişier date" iar fişierul de date este generat. Vizualizarea lui se poate face prin actionarea "Arată fişier date". Opţional, putem efectua şi "Medie" pentru a face media capilarelor la nivelul atomic corespunzător. Aşadar se poate determina în medie care este calitatea vascularizaţiei a unui individ pe nivelul studiat. Apoi la apelarea `Acţiuni > Media capilarelor/set imagini...` se va deschide o fereastră asemănătoare celei de la `Deschiderea unei imagini`. În cadrul acestei ferestre se pot face *selecţii multiple*. Odată încărcate fişierele selectate, se vor culege datele necesare mediei, apoi se va afişa rezultatul final: `Media capilarelor`.

3.5 Concluzie

Crearea aceastei aplicaţii menită să simplifice şi să înbunătaţească munca departamentelor de cercetare în medicină cu un plus de rapiditate si acurateţe s-a dovedit a fi o adevarata incursiune in lumea medicinei si a microscopicului, în care similitudinile dintre corpul omenesc şi un calculator atât software cât şi hardware devin din ce în ce mai interesante cu cât conştinentizăm dependenţa reciprocă, dusă până în extreme, unde putem vorbi de vieţi salvate, consecinţă a puterii de calcul a calculatorului.

Ca posibile dezvoltări viitoare, aplicaţia poate ajunge la optimizări mult mai complexe, cu timpi de rezolvare mai buni. Un mare pas ar fi excluderea modului de captură, si înlocuirea lui cu un modul automat de setare a parametrilor, bazat pe interpretarea histogramelor, şi ajustarea corespunzătoare a intervalelor. De asemenea se poate ajunge la o implemetare hardware, în care "microscopul viitorului" va face "LIVE" toate aceste analize, urmând ca la sfârşit să listeze rezultatele finale.

Bibliografie

[1] Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing*, Prentice Hall, Inc. Upper Saddle River, New Jersey 07458, 2th edition, 2002, 0-201-18075-8

[2] Kenneth R. Castleman: *Digital Image Processing*, Prentice Hall; 2nd edition, September 2, 1995.

[3] Mark Galer, Les Horvat: *Imaginea Digitala*, AD LIBRI, 2004, 973-86781-6-1.

[4] Bahram Javidi: *Image Recognition and Classification*, CRC; 1st edition, June 15 2002, 978-0824707835.

[5] C. Arseni, I. Petrovici: *Bolile vasculare ale creierului şi ale măduvii spinării*, Editura Medicală, Bucureşti 1965.

[6] V. Popescu: *Neurologie pediatrică*, Editura Teora, Bucureşti 2004.

[7] Andrew Kirillov: *http://www.aforgenet.com/framework/docs/*

La sfarsitul articolului vor fi pozitionate detaliile cu privire la afilierea fiecarui autor. Daca sunt mai multi autori se va folosi un tabel cu coloane cu latimea de 8.2 cm pentru 2 autori, 5.6 cm pentru trei autori, Times New Roman, 8pt., spatiul dintre caractere este de 0.2 pt

DROBOTĂ Florin-Robert
Universitatea Transilvania
Braşov
Informatică aplicată
Adresa Str. Iuliu Maniu, nr. 50,
Cod postal: 500091, Brasov
ROMÂNIA
E-mail:
robert.drobota@gmail.com

Sesiunea Națională de Comunicări Științifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

3D animation method from a fixed angle

Iulian Flester, Ivascu Carina
Coordonator: Lector Dr. Anca Vasilescu

Abstract

The theme of this project is to find a method to make a 3D animation of a picture,from a fixed angle. Our intention was to use few rendering resources and to creat display interfaces(for Napier's rods) and animation interfaces(for Schickard's machine).We have used a 2D programming language. We have had as an example the disadvantage that OpenGL and DirectX have, by using many resources and from this reason some objects can not be rendered at a good quality on some mobile phones or PC's. From this reason, we have tried to find an alternative, that can create an ilusion of an animated 3D object,seen from a fixed angle.

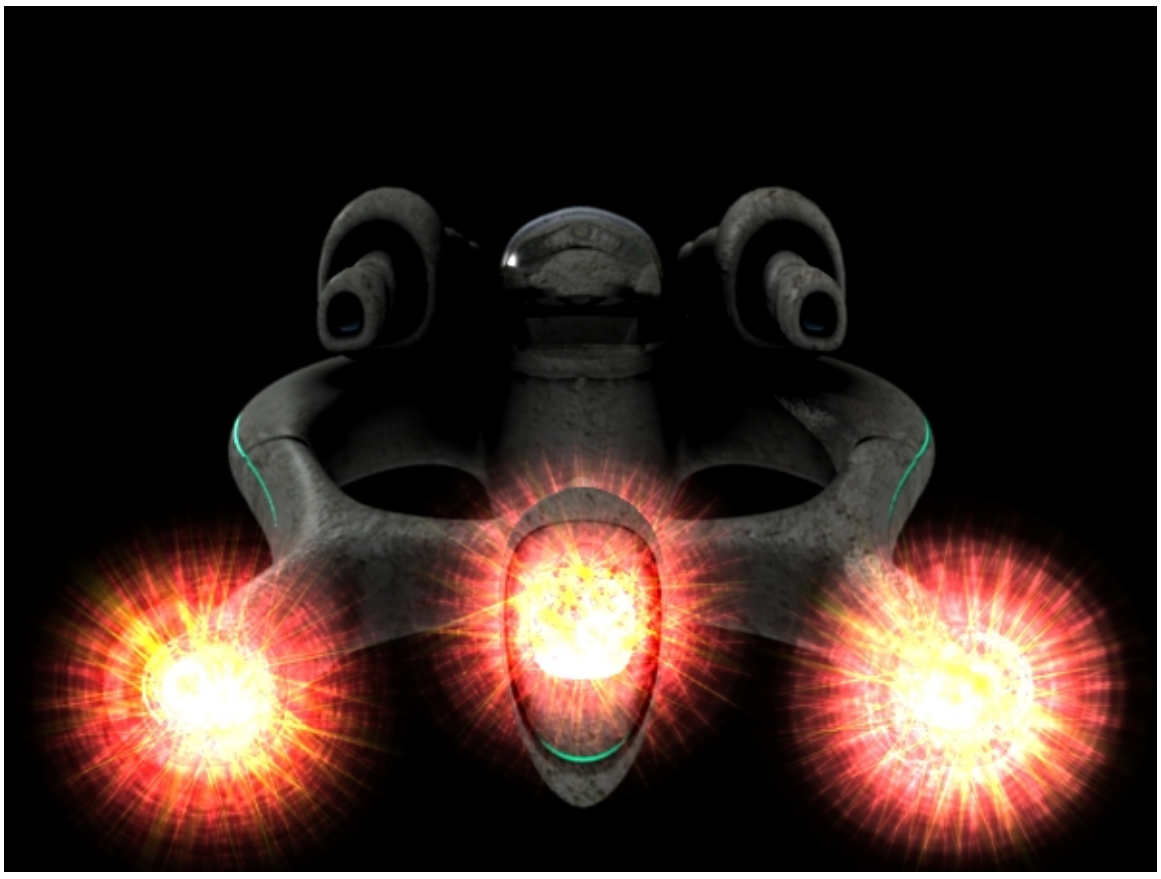


Fig 1: A photo of a 3d model.

Metoda de animatie 3D dintr-un unghi fix

Iulian Flester, Ivascu Carina
Prof.Coordonator: Anca Vasilescu

Abstract

Tema si scopul acestei lucrari au fost, de a gasi o modalitate de a face o animatie 3D, dintr-un unghi fix, la o calitate de randare cat mai mare, plecand de la o imagine.Am dorit sa folosim cat mai putine resurse de randare.De asemenea am folosit un limbaj de programare 2D.

Plecand de la aceasta idee, am dorit sa cream interfete de afisare interfete de animare(pentru masina lui Schickard).

De asemenea, am plecat de la ideea ca DirectX si OpenGL pot randa obiecte 3D, folosind coordonatele punctelor din care este alcatuit obiectul.Acestea au dezavantajul ca folosesc multe resurse si din aceasta cauza pe unele calculatoare sau telefoane mobile, anumite obiecte nu pot fi randate la o calitate buna.Din aceast motiv, am cautat sa gasim o alternativa care sa permita crearea unei iluzii de animatie 3D, a unui obiect vazut dintr-un anumit unghi fix.

Oasele lui Napier este un dispozitiv inventat in anul 1600,folosit pentru a inmulti si imparti doua numere.Acest dispozitiv s-a bucurat de o popularitate mare in acea vreme atat in Marea Britanie,cat si pe continent.

Masina lui Schickard,inventata la inceputul secolului al XVII-lea de catre Wilhelm Schickard,putea sa adune,sa scada si sa inmulteasca numere; inmultirea putea fi realizata folosind oasele lui Napier. Din pacate, înainte ca dispozitivul sã fie pe jumãtate realizat, fabrica unde era adãpostit a ars. Wilhelm Schickard si întreaga sa familie au murit de ciumã, înainte ca el sã construiascã alt model. Mult mai târziu, un mecanic a recreat instrumentul având ca ghid vechile scrisori ale lui Schickard. Dacã aceastã inventie ar fi fost disponibilã inventatorilor din acea vreme, ea ar fi putut schimba în totalitate istoria calculatoarelor.

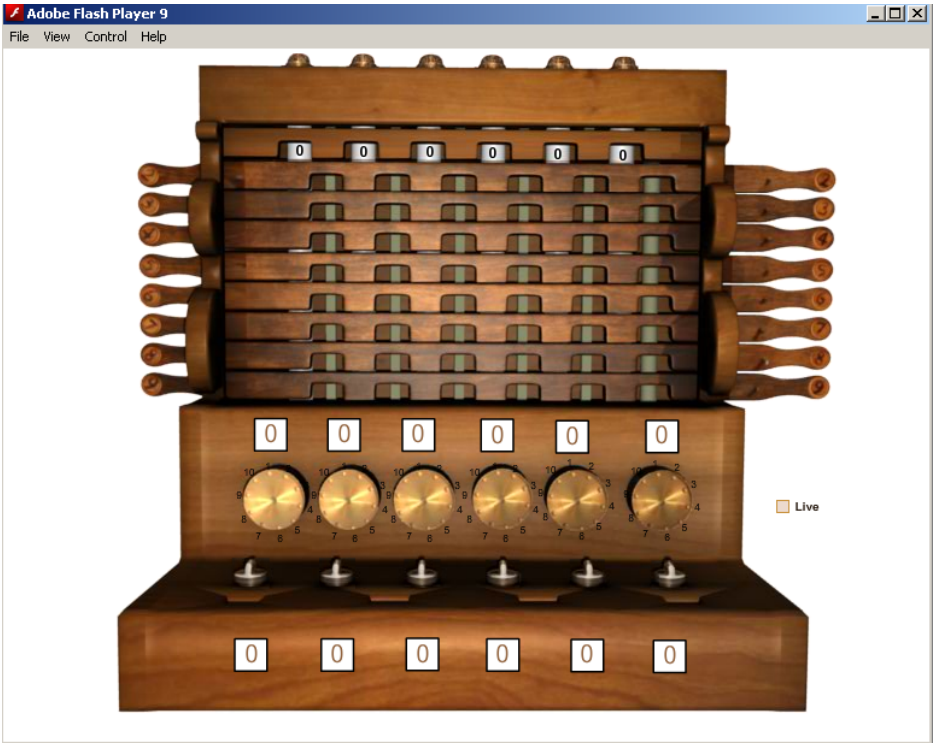


Fig 2:Masina lui Schickard

Lucrarea cuprinde 2 sectiuni, in care sunt prezentate modul de functionare al celor doua dispozitive,precum si modul de realizare al celor doua simulatoare.

1 Oasele lui Napier

Acest dispozitiv era format din 10 cilindrii de fildes, fiecare fiind format din 9 patrate.
Pe primul cilindru erau trecute in cele 9 patrate,cifrele de la 1 la 9(scrise vertical).Pe celelalte 9”oase” erau trecute in patratul de sus cifrele de la 1 la 9,iar in patratele de jos multiplii cifrelor respective.Fiecare patrat,cu exceptia celui de sus,era impartit de o diagonala,care avea urmatorul rol:cifra unitatilor a multiplului respectiv era trecuta sub diagonala,iar cifra zecilor deasupra diagonalei.
Multiplul era calculat inmultind cifra aflata pe linia respectiva,a primului ”os”(cel cu cele cu cifrele de la 1 la 9,scrise vertical)cu cifra aflata in primul patrat de pe fiecare ”os”.

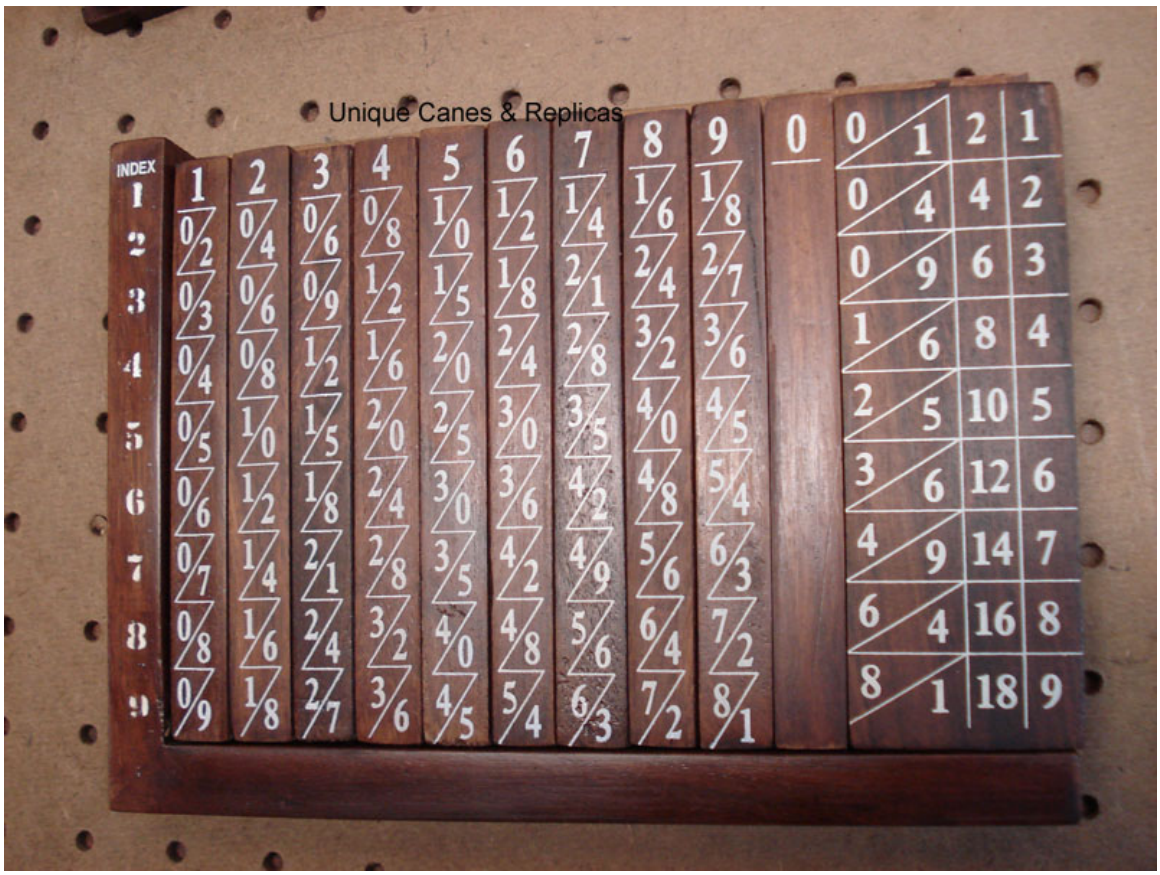


Fig. 3:Oasele lui Napier

Daca se dorea sa se inmulteasca numarul 4138 cu 567,se proceda astfel:

- Se fixau oasele lui Napier,astfel incat sa ramana oasele corespunzatoare cifrelor deinmultitului si se citea ce scrie pe fiecare linie corespunzatoare cifrelor inmultitorului
- Se lua fiecare cifra a inmultitorului si se inmultea cu deinmultitul.

Se lua 5 si se inmultea cu 4138 de la dreapta,utilizatorul se uita pe oase in dreptul liniei 5 si vedea in dreptul lui 4,1,3 si 8 ce valori aveau multiplii.

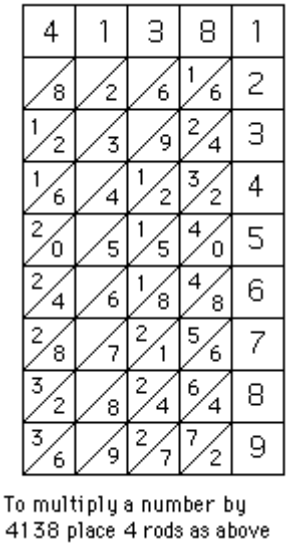
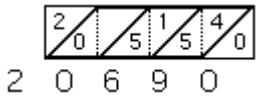
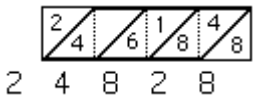


Fig 4.

Aceste patrate se obtin din Oasele lui Napier,inmultind 5 cu 4138:
Si se aduna de la dreapta spre stanga cifrele ce erau in fiecare paralelogram



- La fel se proceda si pentru celelalte cifre ale inmultitorului,adica pentru 6 si 7,astfel se mai obtineau:



corespunzatoare liniei 6

Iar pentru 7 se obtinea numarul 28966

- Rezultatul final se obtinea adunand aceste 3 numere,dar in felul urmator:

20690+
24828
28966

Fiecare numar este situat sub cel de dinaintea lui,dar cu o unitate mai la dreapta

Rezultatul inmultirii este astfel :

20690 +
24828
28966
2346246

Pentru a realiza inmultirea a doua numere,folosind oasele lui Napier,am utilizat mai multe functii:”extragcif”,”extragcifprod”,
“afisare” si “formarenr.”
Functia “extragcif” extrage cifrele unui numar si le pune intr-un vector si returneaza numarul cifrelor.
Functia “extragcifprod” extrage cifrele unui produs,pune cifrele zecilor in vectorul z,iar cifrele unitatilor in vectorul u.
Functia “afisare” afiseaza elementele unui vector.
Functia “formarenr” aduna cifrele din vectorii z si u,conform regulii de la oasele lui Napier,se aduna cifrele din paralelogram,de la dreapta spre stanga.

2 Masina lui Schickard

Fig 5:Masina lui Schickard

Masina putea fi folosita pentru a aduna,a scadea si inmulti doua numere.
In partea de sus a masinii, se pot observa 6 rotite care sunt folosite pentru a afisa deinmultitul in casutele de dedesubt.Pentru a realiza inmultirea se foloseste principiul folosit la oasele lui Napier (deoarece masina are inauntru cei 9 cilindrii corespunzatori multiplilor cifrelor de la 1 la 9).Astfel, se trag placile de pe lateral, corespunzatoare cifrelor inmultitorului si apar multiplii scrisi ca la oasele lui Napier.De exemplu, daca se doreste sa se inmulteasca 524 cu 48,se vor trage placutele corespunzatoare cifrelor 8 si 4 dupa ce de la rotite s-a fixat numarul 524.
Cele 6 discuri dedesubtul ferestrelor folosite pentru inmultire,sunt folosite pentru adunari si scaderi.Deasupra lor se afla niste ferestre mici care indica rezultatele.Aceste discuri sunt corespunzatoare cifrelor unitatii,zecilor,sute s.a.md.. Acestea sunt construite in asa fel incat la depasirea cifrei 9 la un disc,in fereastra deasupra discului urmator din stanga,va aparea o unitate in

plus fata de cifra ce era inainte. Acelasi lucru este valabil si pentru scadere, numai ca in sens invers se rotesc discurile.
In realizarea simulatorului acestei masini, am facut o clasa pe nume player care preia doua argumente.Unul MovieClip si nr-ul de frame-uri. Aceasta este compusa din 2 membri(mc si frs) si 3 metode(ms Down,inainte si inapoi).
Mc este numele MovieClip-ului si preia primul parametru din constructorul clasei.
Acesta este un obiect care poate contine n(frs)imagini.Al doilea parametru al functiei precedeaza numarul de imagini(frs=n).
In constructorul functiei este adaugata o metoda de tip ascultator (mc.addEventListener) care "asculta" cand un buton este apasat.Cand este detectata apasarea unui buton al mouse-ului, este apelata automat metoda ms Down.

Metoda "ms Down"

- Inlatura ascultatorul pentru a permita transmiterea animatiei.
- Monitorizeaza ce imagine se afiseaza.Daca s-a ajuns la prima imagine, inseamna ca animatia trebuie sa se desfasoare inainte,in caz contrar animatia trebuie sa se desfasoare inapoi.

Metoda "inainte"

- Adauga o functie de tip Timer care se apeleaza de cate imagini sunt in obiectul mc la un interval precizat.
- Daca s-a ajuns la numarul maxim de imagini, se avanseaza la prima imagine, in caz contrar se trece la urmatoarea imagine prin incrementare.

Metoda "inapoi"

- Adauga o functie de tip Timer care se apeleaza de cate imagini sunt in obiectul mc la un interval precizat.
- Daca s-a ajuns la prima imagine, se avanseaza la ultima imagine,altfel se trece la imagina precedenta prin decrementare.

La ambele functii de tip Timer se verifica daca nu s-a ajuns la sfarsitul celor n imagini.In acest caz, se sterg obiectele de tip Timer si se adauga evenimentul de ascultare a mouse-ului.
In cazul placutelor se instanteaza clasa Player.
Toate animatiile sunt bazate pe aceasta clasa.
La discuri se folosesc aceleasi principii, cu precizarea ca nu se animeaza si se deplaseaza cate o imagine de fiecare data cand este apasat butonul corespunzator.
Cand este apasat butonul "+" se trece la urmatoarea imagine.
Cand este apasat butonul "-" se trece la imaginea precedenta.
La sistemul de adunare au fost adaugate suplimentar constrangerile de rotatie, folosind instructiuni de tip "if-else".

3 Concluzii

Sistemul ar putea fi extins pe un numar mai mare de unghiuri, in asa fel incat animatia sa fie facuta cat mai spatial, in detrimentul ocuparii unui spatiu cat mai mare pe disc.
Este foarte important sa se obtina un sitem de animatie cat mai bun calitativ, care sa poata fi rulat pe majoritatea calculatoarelor si altor dispozitive care permit folosirea acestor principii.

BIBLIOGRAFIE

[1] *http://www-history.mcs.st-andrews.ac.uk/history/Extras/Napier_rods.html*
[2] *http://en.wikipedia.org/wiki/Napier's_bones*
[3] *http://history-computer.com/MechanicalCalculators/Pioneers/Schickard.html*
[4] *http://ro.wikipedia.org/wiki/Istoria_ma%C5%9Finilor_de_calcul*
[5] *http://er.adrianmoisei.com/hardware/Istoria_calculatoarelorI.html*
[6] *http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/*

FLESTER Iulian si Carina IVASCU
Universitatea Transilvania Brasov
Facultatea de Matematica-Informatica
Str.Iuliu Maniu,nr 50, Brasov
ROMANIA
flester.iulian@gmail.com

Sesiunea Națională de Comunicări Științifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Arhitectura orientata pe servicii in e-marketing

Valentina Lazar
Coordonator: Lect. Univ. Drd. Ralf Fabian

Abstract

„Internet marketing, also referred to as i-marketing, web-marketing, online-marketing, Search Engine Marketing (SEM) or e-Marketing, is the marketing of products or services over the Internet.

The Internet has brought media to a global audience. The interactive nature of Internet marketing in terms of providing instant response and eliciting responses, is a unique quality of the medium. Internet marketing is sometimes considered to have a broader scope because it not only refers to the Internet, e-mail, and wireless media, but it includes management of digital customer data and electronic customer relationship management (ECRM) systems.

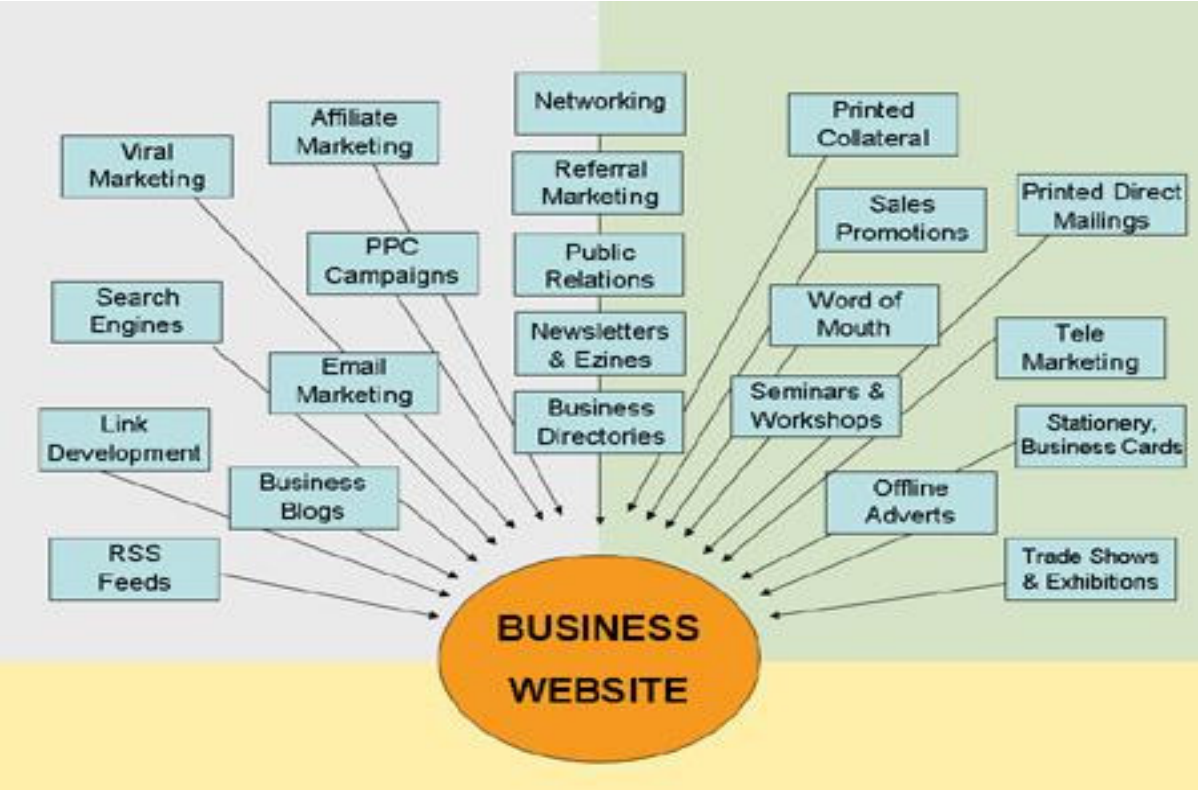
Internet marketing ties together creative and technical aspects of the Internet, including: design, development, advertising, and sales.Internet marketing also refers to the placement of media along many different stages of the customer engagement cycle through search engine marketing (SEM), search engine optimization (SEO), banner ads on specific websites, e-mail marketing, and Web 2.0 strategies."

1 E-marketing. Un raspuns la cat mai multe intrebari.

Odata cu inceputurile Internetului un nou tip de afacere s-a profilat din ce in ce mai pronuntat pe plan international: comertul online. Au aparut nenumarate site-uri si firme care se ocupa cu vanzarile en-gros de produse prin Internet, cu organizarea de lictitatii online, cu furnizarea accesului contracost la pagini cu informatii, intr-un cuvant abordeaza o forma sau alta de comert electronic, un domeniu aflat in plina expansiune. Avantajul comertului online este evident: piata pe Internet creste exponential de la un an la altul. Un site bine lucrat, cunoscut si care ofera produse de calitatela preturi bune va avea mai multi vizitatori, mai motivati si mai inzestrati financiar decat "Consignatia" din coltul strazii.

Magazinul virtual si comertul electronic din Romania inregistreaza un ritm de dezvoltare deosebit de dinamic. La nivelul anului 2005 este de asteptat ca cifra tranzactiilor sa depaseasca valoarea de 50 de milioane de euro.Cauzele acestei evolutii pozitive nu sunt greu de identificat, avantajele comertului on-line, fie ca e vorba de cumparator sau de vanzator, fiind clare.

Avantajul comertului on-line este evident: piata pe Internet creste exponential de la un an la altul. Un site bine lucrat, cunoscut si care ofera produse de calitate la preturi bune va avea mai multi vizitatori, mai motivati si mai inzestrati financiar decat in cazul formelor clasice de vanzare.



Acestea fiind spuse se poate observa o reala imbunatatire a tehnicilor de marketing:

Mijloacele e-marketing:

- fara intermediari;
- segmente foarte reduse de piata;
- marketing " viteza";
- cresterea concurentei;
- confidentialitatea;
- cunoasterea pietei;
- marketingul din perspectiva utilizatorului;

Mijloacele marketingului clasic:

- necesitatea existentei intermediarilor;
- segmentul de piata este incert;
- "viteza" mica de marketing'
- nu permite cunoasterea pietei;
- marketingul din perspectiva utilizatorului;

2 Analiza mediului si acoperirea cat mai multor nevoi.

2.1 Analiza client-firma.

Magazinul virtual si comertul electronic din Romania inregistreaza un ritm de dezvoltare deosebit de dinamic. Cauzele acestei evolutii pozitive nu sunt greu de identificat, avantajele comertului on-line, fie ca e vorba de cumparator sau de vanzator, fiind clare.
Aceasta analiza presupune un raspuns la fiecare din urmatoarele intrebari:

1. Ce grad de siguranta ofera comertul electronic clientilor si comerciantilor
2. Ce dimensiune economica are comertul electronic?
3. Care sunt cele mei mari bariere in comertul electronic?
4. Ce putem cumpara prin Internet?

In continuare trebuie urmarite:

2.1.1 Stabilirea cerinţelor clienţilor

- marketingul este responsabil pentru *identificarea cerinţelor consumatorilor* inclusiv a cerinţelor calitative
- marketingul trebuie să joace un *rol conducător* în următoarele *domenii*
- *stabilirea necesităţilor privind produsul* sau serviciul
- *stabilirea pieţei* aferente produsului sau serviciului şi *analiza ofertelor concurenţei* (inclusiv a aspectelor juridice privind reglementările aferente)
- *stabilirea detaliată a cerinţelor clienţilor,* inclusiv a acelora indirecte şi/sau implicite;
- *stabilirea cerinţelor şi caracteristicilor calitative speciale.*

Aceste *activităţi trebuie efectuate* nu numai în faza de început a dezvoltării unui produs (serviciu) ci în mod *continuu* şi pentru produsele deja existente.

2.1.2 Realizarea produsului preliminar

- marketingul trebuie să realizeze, în cadrul unor proceduri *produsul preliminar*
- *o culegere provizorie de specificaţii* pe care marketingul o transmite celorlalte departamente ale firmei
- trebuie *corelate* în mod direct cu *posibilităţile interne ale firmei*
- constituie *baza proiectării produsului* sau serviciului → elaborarea unor *specificaţii şi caracteristici de calitate formale (*detaliate) = *referinţă* pentru *evaluarea output-ului*
- puncte de vedere ce *trebuie luate în considerare,* de exemplu:
- caracteristici privind performanţele precum şi cele percepute prin intermediul organelor senzitive;
- prescripţiile legilor şi ordonanţelor;
- referinţe privind punerea în funcţiune şi de utilizare
- ambalarea
- factori ai controlului şi asigurării calităţii.

2.2 Analiza mediului de proiectare.

Aceasta etapa resupune o serie de pasi ce trebuie urmariti:

1. **Intelege**

- obiectivele propriei afaceri pe termen scurt, mediu si lung
- rolul site-ului web ca parte a strategiei globale de afaceri
- contextul concurential online si offline in care se plaseaza afacerea

2. **Identifica**

- piata si audienta;
- competitia in cadrul segmentului de piata vizat;
- modul in care competitorii abordeaza drumul spre succes, atat online cat si offline;
- tendintele de evolutie a pietei si audientei;
- tehnologiile de utilizat in munca de implementare;
- punctele tari si punctele slabe ale viitoarei oferte;
- timpul, efortul si bugetul alocat etapelor de proiectare, promovare si intretinere a site-ului;

3. **Proiecteaza**

- continutul propriu-zis al site-ului web, in acord cu toate standardele unei prezentari profesionale;
- structura site-ului web, in asa fel incat navigarea sa fie usoara, clara si a tot cuprinzatoare;
- design-ul site-ului, de asa natura incat elementele grafice sa nu lipseasca dar nici sa nu abunde, totul in armonie;

4. **Valideaza**

- corectitudinea sintactica si morfologica a textelor (validare gramaticala);
- structura semantica a intregului continut
- alcatuirea codului (HTML, CSS, JavaScript, Java, PHP, etc...)
- linia generala de prezentare si ofertare
- dreptul de autor asupra produsului finit.

3 Prezentarea aplicatiei.

Aplicatia urmareste analiza si constructia unui simulator bancar care, din punct de vedere al clientului, raspunde cat mai bine cerintelor, iar din punct de vedere al specialistului in marketing, ofera solutii rapide si eficiente si, mai ales, poate acoperi o gama cat mai larga de intrebari.
In acest sens s-a urmarit dezvoltarea unui site Web , ca interfata cu utilizatorul/clientul, si a unei baze de date cat mai complexe care sa poata sa raspunda nevoilor celor ce se ocupa cu dezvoltarea produselor si cresterea vanzarilor.

Bibliografie

[1] Quentin Zervaas, *Practical Web 2.0Applications withPHP*, Apress, 2008.
[2] William Ballad , Tricia Ballad,*Securing PHP Web Applications,* WoweBook.com.
[3] Cristian Darie and Emilian Balanescu. *BeginningPHP and MySQL ,E-Commerce*.Apress.
[4] http://www.w3schools.com/default.asp
[5] Seminar: *Asigurarea calitatii in marketing.* www.upm.ro/...in-**marketing**.../Seminar%20CM%205.doc

LAZAR VALENTINA
Facultatea de Stiinte din Sibiu
Informatica
Adresa: Str. Ion Rațiu
Nr.5-7, Sibiu, 550012, România
ROMANIA
E-mail: lyah_valy@yahoo.com

Sesiunea Națională de Comunicări Științifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Computational methods in medical imaging analysis

Grigore Lupescu
Coordonator: Lector univ. Dr. Mircea Olteanu

Abstract

This paper presents several methods to optimize the computation of high complexity algorithms used in medical imaging analysis. These imply maximizing CPU (replicated workers model, MPI) and GPU (OpenCL) resources locally or across a network. Though some or all may be used in many analysis algorithms any example will be related only to fractal 3d box counting algorithm. All of the methods presented can be combined into a single complex one – distributed heterogeneous computing - and such a setup is presented briefly.

1 Complexity in medical imaging analysis

1.1 Medical Imaging

Medical imaging is the technique and process used to create images of the human body for clinical purposes (medical procedures seeking to reveal, diagnose or examine disease) or medical science (including the study of normal anatomy and physiology).

Imaging technology includes : Electron microscopy, Radiographic, Magnetic resonance imaging (MRI), Nuclear medicine, Computed Tomography, and Ultrasound. For instance Computed tomography (CT) is a medical imaging method employing tomography created by computer processing. CT produces a volume of data which can be manipulated, through a process known as "windowing", to demonstrate various bodily structures.

1.2 Fractal Analysis

In nature, there exist complex forms that cannot be evaluated neither by standard geometry or by topology. A fractal is a rough or fragmented geometric shape that can be split into parts, each of which is (at least approximately) a reduced-size copy of the whole, a property called *self-similarity.* The basis of fractal analysis comes from the self-similarity property of them.

In fractal geometry, the *fractal dimension*, D, is a statistical quantity that gives an indication of how completely a fractal appears to fill space, as one zooms down to finer and finer scales. It is a well known fact that some tissues have a fixed fractal dimension. Most human tissues have a fractal-like structure.

The box-counting dimension is defined as:

$$\dim_{\text{box}}(S) := \lim_{\varepsilon \to 0} \frac{\log N(\varepsilon)}{\log(1/\varepsilon)}$$

Fractal analysis algorithm does the following :
A) builds a 3d geometric area based on the image height map (grayscale levels) B) for each point in the image selects an area of a given length and does 3d box counting on that region. C) the result is then stored in a matrix of numbers ranging from 0 to 999 representing the mobile part of the number (ex 2.784->784) D) builds a histogram corresponding to the resulted matrix

Fractal 3d box counting analysis yields poor results on single core CPUs. A comparison of the needed time for a complete analysis on a 640x480 image (**Fig.1**) is presented bellow (lower is better)

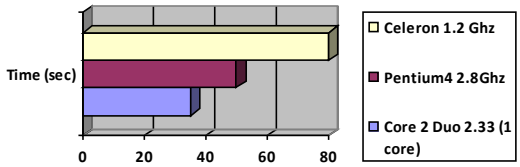


Fig.1 3d box counting algorithm performance

The obtained matrix from fractal analysis can be used to find out the structural differences in closely positioned tissue. In other words similar structure-like tissue should have a corresponding similar color.

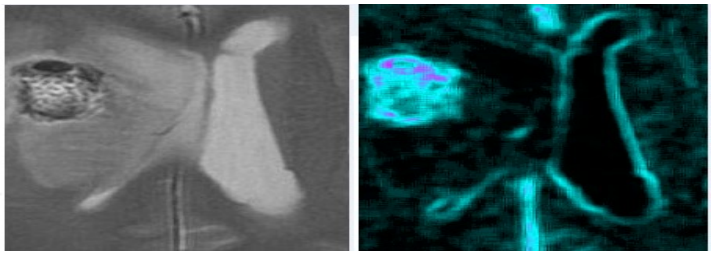


Fig.2 Brain Tumor CT

1.3 Computation Complexity

Due to needed detail in the analyzed data and the complexity of some algorithms, parallel and distributed computing approaches are taken into consideration. Consider medical analysis algorithm that is too complex to be executed on a single processor, be it multi core. Let's assume the it can take advantage of a multi core architecture. We can thus harvest the whole power of a computer (given the

problem can be divided) by working a parallel/distributed solution. There are several problems to this : A) Given different architectures full potential to parallel or distribute the computations will be hard or impossible to achieve, B) Process flow / synchronization problems are difficult to spot and correct, C)Large data can pose difficulties in efficient management.

2 Maximizing CPU resources

For local execution the "replicated workers" model has been considered. Thread pooling enables you to use threads more efficiently by providing your application with a pool of worker threads that are managed by the system. One thread monitors the status of several wait operations queued to the thread pool. When a wait operation completes, a worker thread from the thread pool executes the corresponding callback function. The "replicated workers" model is also implemented in other languages (for instance Java) . This is helpful because we are using several platforms for simplifying the workload.

Fractal 3D algorithm is well suited for the "replicated workers" model because all the operations are done on different areas of the image (zones around each point). Hence we first break the image into N strips depending on the number of available processing units. Each processing unit will work out its strip and finally the result will be the unification of those strips. Our tests have shown a substantial performance gain over the single threaded computational method.

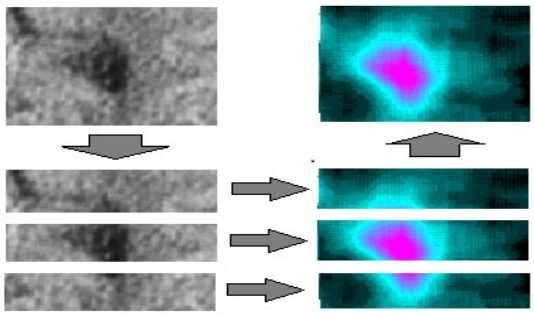


Fig.3 Replicated workers model optimization

Optimizations for local image processing are thus available through models like "replicated workers" model. These scale well with the CPU capabilities in a system. Selecting the right platform for resolving a problem can be a difficult task. Each posses certain advantages and disadvantages while we only have limited access in adding new software features to the computing units. In other words we have little control over the platforms we develop on. This is a performance drawback which we will try to limit by periodically updating the code and so, taking advantage of the new features introduced by the frameworks.

The code has been written so to benefit from Net. Framework 4.0- includes the Task Parallel Library (TPL), a library of objects that make it easier to write code that takes advantage of multiple cores. Another improvement was toward the thread pool, giving a performance boost not only TPL programs, but to all programs that make use of .NET threading services.

Another platform (Java) has been considered but not yet used. JDK 5.0 - added major new support for developing concurrent applications, including JVM changes, new low-level synchronization utilities, and higher-level, thread-safe, high-performance concurrency classes

3 Distributing computations using a process flow chart

To simplify task assignment a method to represent job relationship has been developed. Thus define graphically, data and processes, and interconnect them as to obtain from an input and using a set of processes a desired output.

The graph corresponding to the fractal analysis is presented bellow. Fractal module reads input from D1 source (in our case images) works on them and outputs data in D2. Data in D2 is taken by the Histogram process and outputs it in D3 data. Finally the Compare process tries to see if there are connections between histograms and certain problems and either restarts the cycle with different settings (arguments) /different data (extracts irrelevant data), or finishes with certain statistical data.

Job Collections are sent to the server which in turn distributes them to the clients. Every computer is connected to a share point. The server application only distributes jobs based on execution commands given in a shell . This decreases the band which use dramatically. There is also a drawback to the fact that the data is centralized. The distribution is made based on the client's demand. In other words a client which already has work to finish will not post a job demand for the server until he is done.

The platform deals with this problems by allocating a certain time for each job (TTL) and a certain number of tries (REDO). So to speak, if the program is defective or fails repeatedly it will be cancelled. Problems that may occur include : program execution on client freezes, client shuts down or becomes unreachable, program crashes and emits wrong output. Every single event that involves a failed job, redistribution of tasks, defective modules is reported in an event log.

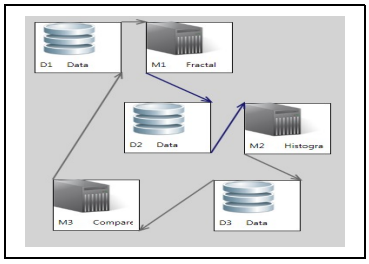


Fig.4 Process flow chart

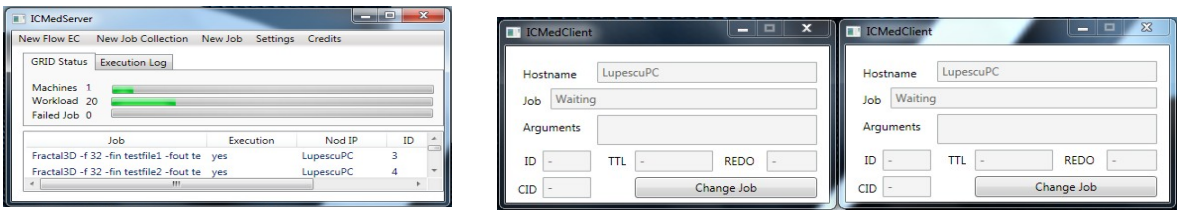


Fig.5 Server App(left), 2Clients App(right)

4 Heterogeneous computing

GPUs are massively multithreaded many-core chips :A) Hundreds of cores, thousands of concurrent, threads, B)Huge economies of scale, C)Still on aggressive performance growth

While CPUs range in number of cores from 1 up to 8-16, GPUs can reach hundreds of cores. GPUs are ideal for massive parallel computations. General-purpose computing on graphics processing units (GPGPU, also referred to as GPGP and to a lesser extent GP) is the technique of using a GPU, which typically handles computation only for computer graphics, to perform computation in applications traditionally handled by the CPU. The GPGPU programming landscape has rapidly evolved over the past several years, so that now there are several approaches to programming GPUs.

OpenCL (Open Computing Language) is a framework for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs, and other processors. OpenCL gives any application access to the Graphical Processing Unit for non-graphical computing. Thus, OpenCL extends the power of the Graphical Processing Unit beyond graphics (General-purpose computing on graphics processing units).

```
cl_uint num_devices_returned;
cl_device_id devices[2];
err = clGetDeviceIDs(NULL, CL_DEVICE_TYPE_GPU, 1,
                     &devices[0], num_devices_returned);
err = clGetDeviceIDs(NULL, CL_DEVICE_TYPE_CPU, 1,
                     &devices[1], &num_devices_returned);

cl_context context;
context = clCreateContext(0, 2, devices, NULL, NULL, &err);

cl_command_queue queue_gpu, queue_cpu;
queue_gpu = clCreateCommandQueue(context, devices[0], 0, &err);
queue_cpu = clCreateCommandQueue(context, devices[1], 0, &err);
```

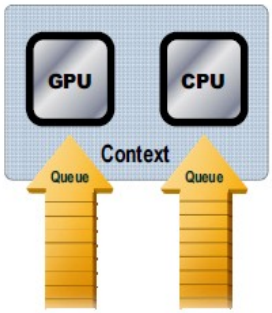


Fig.6 Example code (left) and distribution scheme (right)- OpenCL

5 Distributing heterogeneous computations using a flow chart

In the platform that is at work a solution that wraps around OpenCL and other distributed frameworks is taken into consideration. The full potential of every computer node in a network can be used if the computation can be distributed with respect to the architecture.

Tools that would be able to take the load off of the developer such as diagrams that generate process and data flow have been designed and tested. Also a solution that would provide a share point for all of the computer nodes takes a lot of the data distribution problems. A system capable of monitoring the state of each process and capable of making decisions is mandatory.

An illustration of using heterogeneous computing is illustrated bellow. Main idea is to distribute computations using a flow chart (having a setup as presented in **Section 3)** and making associations between nodes – jobs based on hardware capabilities and job

requirements. It is used in computer clusters and supercomputers. Though for the applications in **Section 3)** WCF (windows communication foundation) has been used for pc-pc communication a better approach is to use MPI. Message Passing Interface (MPI) is a specification for an API that allows many computers to communicate with one another. In the scheme bellow MPI is used.

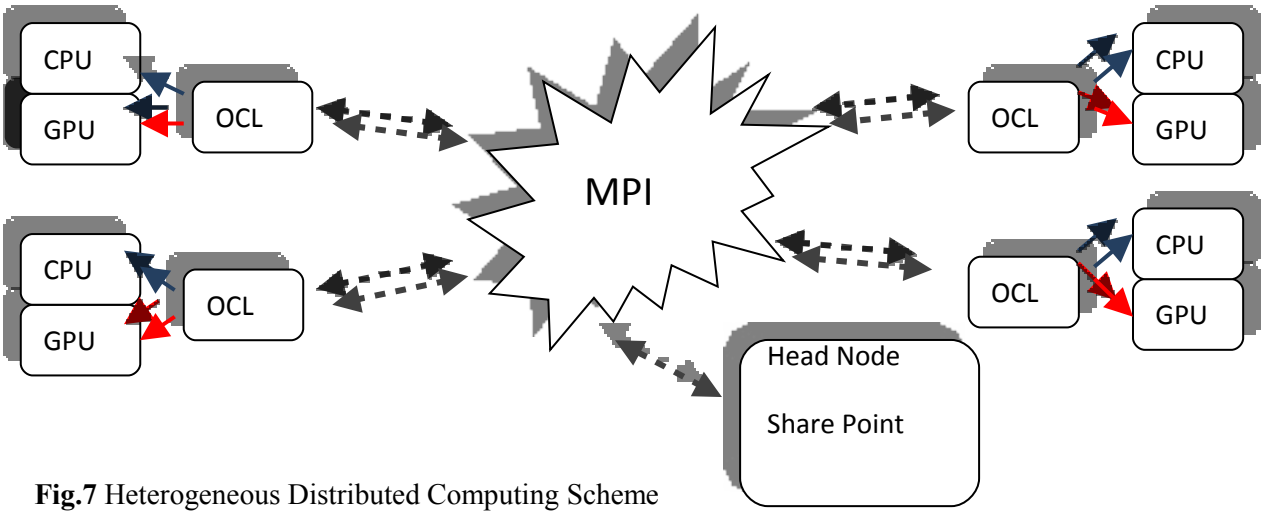


Fig.7 Heterogeneous Distributed Computing Scheme

6 References

Jeffrey Dean and Sanjay Ghemawat, MapReduce: *Simplified Data Processing on Large Clusters*

Rodica Dragomir, M. Olteanu, M. Tanase, *The Analysis Of CT and MR Brain Images using Box Counting* - Type Methods, Proceedings of IAFA'2003, Bucharest, Romania, May 2003, p. 267-272.

Ciprian Dobre, Florin Pop, *Sisteme de programe pentru retele de calculatoare* ,Politehnica Press,2008, p. 39-72

Radu Dobrescu,Dan Alexandru Iordache, *Modelarea Complexitatii*, Politehnica Press,2008

P. Luzi, G. Bianciardi, C. Miracco, M.M. De Santi, M. Del Vecchio, L.Alia, P.Tosi, *Fractal analysis in human pathology*. Ann N Y Acad Sci 879, 255-257, 1999

B. Mandelbrot, *The Fractal Geometry of Nature*, Academic Press, 1975, New York.

T. Mattfeldt, *Nonlinear deterministic analysis of tissue texture: a stereological study on mastopathic and mammary cancer tissue using chaos theory.* J. Macrosc. 185, 47-66, 1997

GRIGORE LUPESCU
FILS Politehnica Bucuresti
Computer Science, 2nd year
ROMANIA
lupescu_grigore@hotmail.com

Sesiunea Națională de Comunicări Ştiințifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Optimizarea sarcinilor cotidiene

Laura Nechifor, Daiana Teona Negulici
Coordonator: Asist univ. drd. Ing. Alexandru RADOVICI

Abstract

The project consists in an apllication for everyone overcome by the numerous number of simple but annoying daily tasks. In a few words, it is an Android project for mobile devices that helps with all these by grouping your daily tasks in a way that makes you finish faster and get more time for the important issues or for yourself.

1 Introducere

Fiecare dintre noi are de mers macar o dată pe zi la bancă, la supermarket, la benzinărie sau la o instituţie publică. Însă de cele mai multe ori lasăm la o parte câte un detaliu, un mic aspect ce pare nesemnificativ la prima vedere sau puţin impotant, dar ce se dovedeşte a fi într-un final imperios necesar, omitem să luăm în considerare orarul instituţiilor şi locurilor în care avem nevoie să mergem sau uităm pur şi simplu că mai avem ceva de făcut în aceeaşi locaţie.
Şi atunci totul devine complicat, insuportabil de lung, meticulos, iar de cele mai multe ori, nervii noştri, puşi din greu la încercare, cedează. Ne întrebăm dacă toate eforturile noastre sunt într-adevăr cu folos, dacă timpul nu a fost pierdut în zadar, dacă nu există o cale mai uşoară şi mai eficientă.
Răspunsul la întrebare este mai totdeauna nu.
De aceea, noi ne-am gândit să venim în ajutorul tuturor celor care preţuiesc timpul şi care au nevoie de el pentru a-l utiliza în scopuri mai bune, mai importante.
Astfel, am dezvoltat o aplicaţie care-şi aminteşte pentru dvs. toate lucrurile mărunte pe care le aveţi de îndeplinit, unde anume trebuie să mergeţi şi dacă nu cumva aţi putea să mai şi împuşcaţi doi iepuri deodată. De ce? Fiind propriul dvs. ghid, dar şi propriul dvs. contabil în acelaşi timp, ea te avertizează când benzinăria şi notarul sunt vecini, când sunteţi aproape de supermarketul de lângă bancă sau când primăria, punctul de plată al facturilor al distribuitorului dvs. şi piaţa sunt în conferinţă.

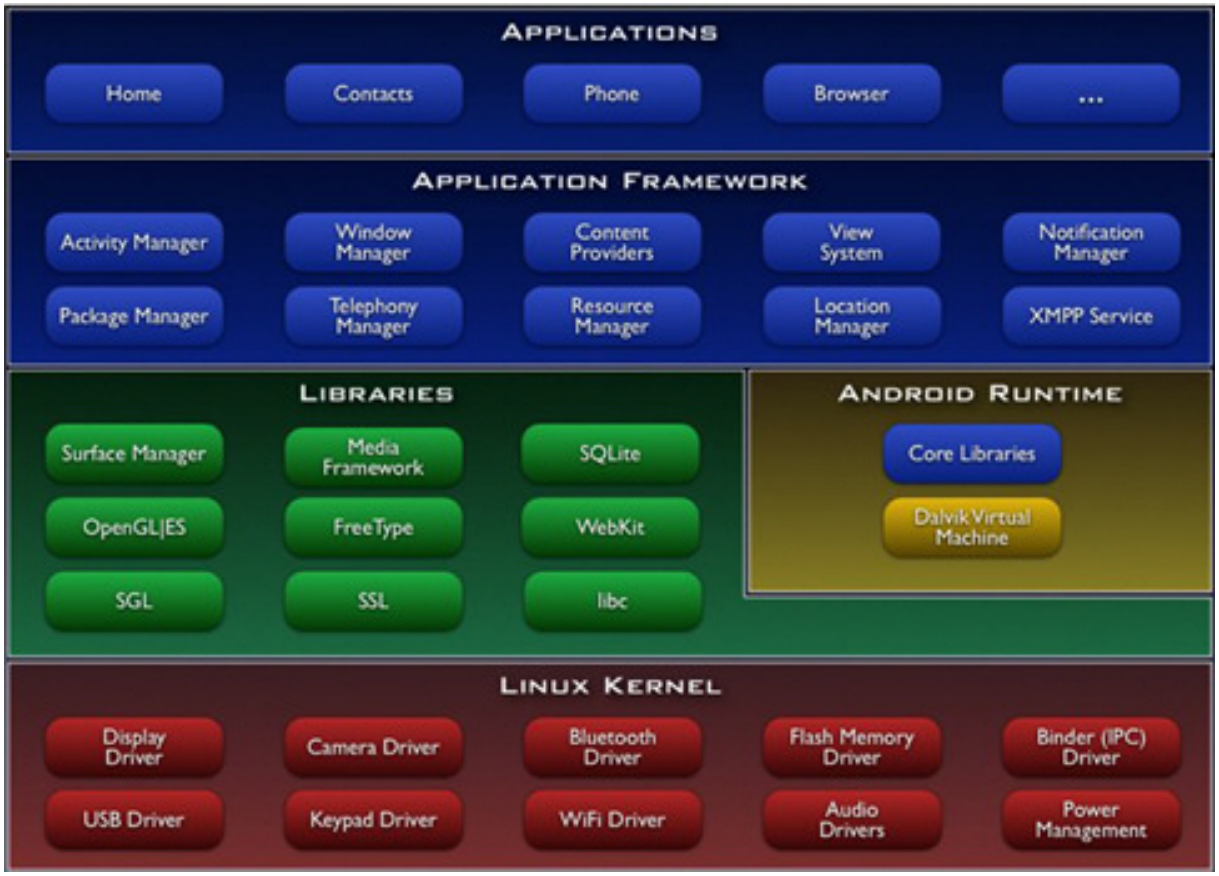
2 Realizarea proiectului

2.1 Android

Proiectul este realizat în Android, un sistem de operare pentru telefoanele mobile.
Android este o platformă dezvoltată de Google, bazată pe Linux (fapt care îi va da ocazia să fie portat pe o varietate de platforme în viitor), folosind o maşină virtuală Java optimizată, Dalvik. Un aspect important este reprezentat de statutul pe care acesta îl are, de open-source.

Aplicaţiile disponibile sunt de mai multe tipuri: activităţi, servicii, baze de conţinut, intenţii şi receptori.
Înainte de a începe scrierea de cod, este necesară instalarea Java, a unui IDE şi a Android SDK. Kit-ul SDK funcţionează pe Linux, Windows şi Mac OS X. Aplicaţiile realizate desigur funcţionează pe orice dispozitiv Android.
Limbajul folosit este Java. JDK 5 sau 6 este necesar. Instalarea unui mediu de dezvoltare Java este de asemenea necesară. Am folosit Eclipse, fiind gratuit şi susţinut de creatorii Android de la Google.
Pentru rularea programului Android se foloseşte fereastra Package Explorer, prin click dreapta pe HelloAndroid Project şi selecţie Run as – Android Application. Dacă simulatorul nu ştie ce dispozitiv vom folosi pentru simulare, acesta trebuie creat. Asta înseamnă să creem un dispozitiv virtual Android (AVD). Odată obţinut acest AVD, fereastra simulatorului Android se va deschide şi va încărca sistemul de operare Android. Rularea unui program Android pe un dispozitiv fizic este aproape identică cu rularea pe emulator şi se face prin conectarea telefonului la calculator cu un cablu USB şi instalarea unui driver special. Cât timp telefonul este conectat, fereastra simulatorului trebuie închisă pentru ca aplicaţia să ruleze direct pe telefon.
Cîteva părţi din Android ar putea fi familiare, precum Linux Kernel, OpenGL şi baza de date SQL, respectiv SQLite.

Arhitectura acestei platforme este complexă, dar bine organizată, aşa cum se poate observa din imaginea de mai jos.



2.2 Proiectul propriu-zis – aplicaţie Android

Proiectul reprezintă o aplicaţie pentru Android, care va da şansa utilizatorului de a-şi sorta sarcinile zilnice pentru a le îndeplini cu succes.
Folosind baza de date SQLite, evenimentele pe ordinea de zi vor fi stocate în funcţie de prioritate, locul unde acestea se vor desfăşura şi alte criterii.
Utilizatorul poate adăuga sau şterge în orice moment lucrurile ”de făcut”, modifica prioritatea sau adăuga locaţii în care acestea se pot desfăşura. Folosind Google Maps, locaţiile sunt marcate de utilizator prin introducerea coordonatelor exacte sau prin marcaj pe hartă. În momentul în care telefonul detectează apropierea de unul din aceste locuri, utilizatorul va primi o înştiinţare în acest sens.
Alertele vor atenţiona asupra expirării evenimentului, incapacităţii de găsire a locaţiei marcate de utilizator sau incapacităţii de conectare la Sistemul GPS.
Utilizatorul îşi poate organiza necesităţile şi bifa problemele rezolvate. Baza de date se va actualiza.

Utilizatorul va fi atenţionat de fiecare dată cu câteva minute înaintea expirării evenimentului, culoarea atenţionării schimbându-se din verde în galben şi în cazul în care acesta expiră fără ca rezolvarea sa să fie înregistrată, el va fi semnalat în continuare în cod roşu.

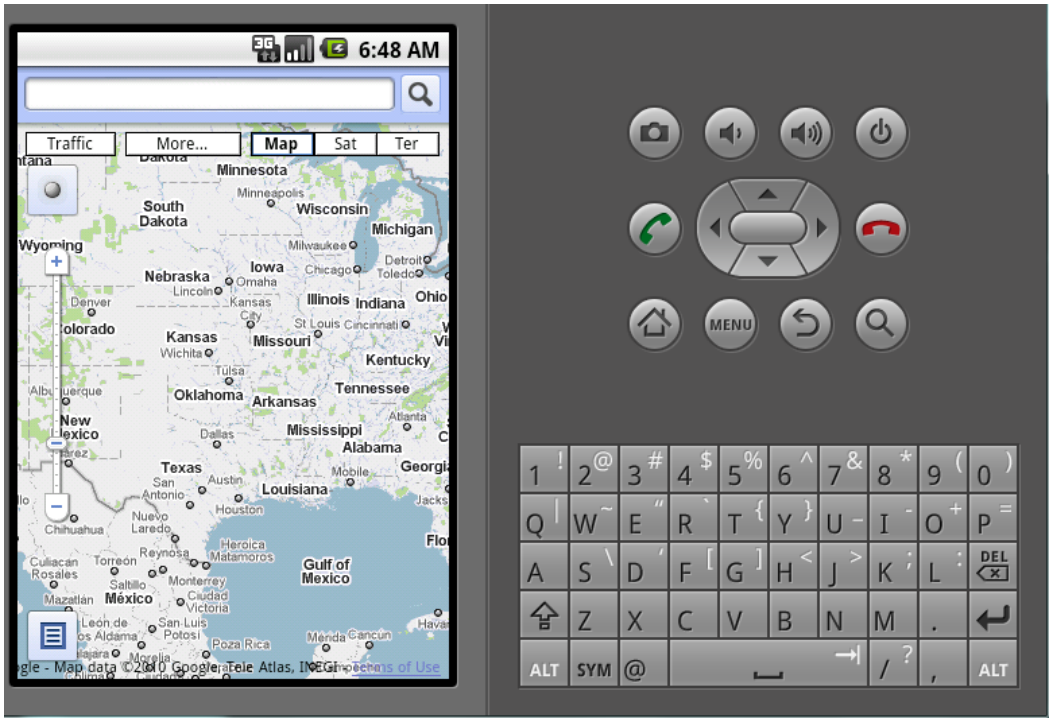
3 Codul sursă a programului

Folosind un program de editare pentru platforma Android, respectiv Eclipse se reuşeşte implementarea unui widget.

Interfaţa realizează accesul utilizatorului la aplicaţie, permiţându-i acestuia să:

- Vizualizeze sarcinile existente şi neexpirate
- Introducă sarcini de făcut pe listă
- Ştergă sarcini din listă
- Să actualizeze sarcinile din listă (sarcina, locuri favorabile rezolvării, stadiul rezolvării, data expirării etc.)
- Să acceseze Google maps pentru a
    - Introduce coordonatele locului în care poate rezolva o sarcină, găsirea locului şi salvarea lui
    - Găsi locul în care poate rezolva o sarcină prin marcare pe hartă şi salvarea lui
    - Cere realizarea celui mai scurt traseu până la rezolvarea unei sarcini, în functie de locul pe care îl are salvat
    - Simula un traseu până la locul rezolvării unei sarcini şi estimarea timpului necesar

- Să afle locul cel mai apropiat în care poate rezolva sarcina în funcţie de locurile salvate în baza de date



De asemenea, utilizând SQLite, baza de date pentru Android şi accesând Google Maps, programul devine deosebit de util. Baza de date salvează sarcinile utilizatorului, data la care expiră sarcina, locuri în care ea poate fi îndeplinită, statusul acesteia (cele expirate nu se mai afişează), stadiul acesteia (îndeplinită sau nu).

Bibliografie

[1] Mark MURPHY, *Beginning Android*, Apress, 2009
[2] Ed BURNETTE, *Hello Android: Introducing Google's Mobile Development Platform*, Pragmatic Bookshelf, 2009
[3] Hello Views, http://developer.android.com/guide/tutorials/views/index.html
[4] Android.widget, http://developer.android.com/reference/android/widget/package-summary.html
[5] Creating Menus, http://developer.android.com/guide/topics/ui/menus.html
[6] Android Application Fundamentals, http://developer.android.com/guide/topics/fundamentals.html

NECHIFOR Laura
Universitatea Politehnica din Bucuresti
Electronica aplicata
Splaiul Independentei 313, Bucuresti
ROMANIA
lauri_mari@yahoo.com

NEGULICI Teona Daiana
Universitatea Politehnica din Bucuresti
Electronica aplicata
Splaiul Independentei 313, Bucuresti
ROMANIA
teodaia@yahoo.com

Sesiunea Naţională de Comunicări Ştiinţifice a Studenţilor
"Imaginaţie, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Sistem de management şi gestiune a parcărilor

Dragoş Iulian Obancea
Coordonator: Lect. Dr. Lucian Sasu

Abstract

One of the main problem around the world regarding the travel system is the management of the parking places. Unfortunately the solutions used until now are not as effective as they wanted to be. That is why it is of utmost importance the management and the organization of the areas where parking places can not be built.This paper is aiming to describe a new management system of the overcrowded parking places and much more. This concept is presented through a web application that permites the administrators to change parts from the existing parking places in some that can be reserved. In this way, the users have the possibility to make online reservations, and so they avoide losing their time in traffic. There is an advantage not only for the ones that need the same parking place every day, but also for the others. To search a parking place is very easy. Google Maps is used in the application, it receives and gives the required information. This approach is feasible because all the parking places are collected in a single data base.

1 Introducere

Una din marile probleme cu care se confruntă administraţiile locale pe de o parte şi conducătorii auto pe de alta, o reprezintă locurile de parcare. Ritmul galopant de creştere a numărului de autovehicule din ultimul deceniu, a sugrumat multe din îngustele drumuri şi străzi ale marilor oraşe, proiectate pentru un volum mult mai scăzut al traficului. Aceasta, împreună cu gestionarea neeficientă a autorităţilor competente au condus la un adevărat haos pe domeniile publice din marile centre ale ţării şi nu numai.

Conducători auto sau pietoni, cu toţii suntem conştienţi de amploarea acestei probleme. Cu toţii am fost puşi în situaţii dificile provocate de: autovehicule parcate pe toată suprafaţa trotuarului, benzi de circulaţie blocate de maşini parcate ilegal sau de şoferi ce caută cu disperare un loc de parcare liber, conflicte iscate între locatari de bloc şi nu numai, cu privire la dreptul de a parca autovehicolul într-un anumit loc şi multe astfel de situaţii deranjante.

Dacă privim cu atenţie, nici peste graniţă situaţia nu este foarte îmbucurătoare. Chiar şi marile oraşe ale lumii, cum ar fi New York, Sidney, oraşe ce se bucură de străzi şi parcări numeroase şi încapătoare, se confruntă cu aceeaşi problemă. Ideile de rezolvare sunt multe şi diverse, unele ridicole, altele chiar foarte bune. Se întâlnesc, de la parcări subterane cu lifturi pentru autovehicule, până la locuri de parcare blocate cu stâlpi hidraulici acţionaţi la telecomandă sau la introducerea unui cod de verificare. Punerea în practică a acestor idei a fost însă mai greu de realizat şi, în general, soluţiile care s-au dovedit a fi utile au fost şi cele mai costisitoare. Este adevarat că ar fi mult mai comod pentru o persoană să folosească un loc de parcare blocat de

stâlpi acționați hidraulic, însă construirea acestor mecanisme este costisitoare şi greu de imaginat, deoarece nu putem îngrădi fiecare metru pătrat din spațiul public. Nu se poate ca la fiecare conflict apărut din pricina unor spații de teren să se mai construiască un gard.

Conform statisticilor, țările cele mai afectate de problema parcărilor sunt cele aflate la un nivel mediu de dezvoltare, categorie din care face parte şi România. În ultimii 20 de ani în aceste state numărul autovehiculelor a crescut exponențial, însă numărul locurilor de parcare a rămas aproape acelaşi. Motivele sunt diverse, de la lipsa fondurilor alocate pentru acest sector pană la eliminarea locurilor de parcare, cauzată de construirea a noi imobile în zonele cele mai aglomerate, în ciuda faptului că în legislația actuală există obligativitatea amenajării de locuri de parcare la toate cladirile noi.

În concluzie, singura soluție rămasă, aplicabilă în toate oraşele lumii, indiferent de nivelul de dezvoltare la care se află, este una simplă, accesibilă şi necostisitoare, dar care din păcate nu a fost găsită sau popularizată până în ziua de astăzi.

2 Formularea problemei

Obiectivul inițial al acestei lucrări a fost următorul: *Să se gasească un nou concept de organizare şi gestionare a parcărilor şi să se construiască un sistem informatic pe baza acestuia, care să ajute la rezolvarea problemei locurilor de parcare din țară şi nu numai*. De remarcat folosirea verbului ”să ajute” deoarece nici un sistem informatic, oricât de evoluat ar fi, nu poate soluționa această problemă, cum o poate face crearea a noi locuri de parcare.

Uşurința acestei cerințe ascunde, însă, un număr mare de obiective pe care o aplicație bazată pe un concept nou, trebuie să le atingă. Dintre aceste obiective, le amintim pe cele mai importante: accesibilitate, flexibilitate, universalitate, simplitate, rapiditate.

2.1 Accesibilitate

2.1.1 Din punct de vedere al proprietarilor şi administratorilor de parcări

Probabil aspectul cel mai important din punct de vedere al proprietarilor şi administratorilor de parcări este cel al costului investițiilor. În general, pentru orice investiție se analizează mai întai suma de bani disponibilă şi abia apoi, în funcție de aceasta, obiectivele finale. De aceea noul sistem de management şi gestiune a parcărilor va trebui să implice costuri minime.

2.1.2 Din punct de vedere al coducătorilor auto

Succesul unui sistem este dat, în general, de durata de timp în care persoanele aflate în anumite categorii se adaptează acestuia şi de uşurința cu care îl folosesc. Astfel, este foarte important ca noul sistem de management şi gestiune a parcărilor să dispună de un suport informatic uşor accesibil care să poată fi utilizat fără a avea nevoie de instalarea de programe suplimentare (exemplu: aplicație web).

2.2 Flexibilitate

De-alungul istoriei, sistemele ce nu au demonstrat că sunt capabile să suporte schimbări au fost primele înlocuite. Drept urmare, flexibilitatea va fi una dintre caracteristicile de bază a noului concept, adică pe lângă faptul că în aplicație se vor putea adăuga permanent zone cu parcări, în cazul în care se modifică un aspect legat de conceptul de funcționare, această modificare să poată fi făcută cu uşurință şi in aplicație.

2.3 Universalitate

În lumea întreagă există la fel de multe moduri de abordare a problemei, câte parcări există. Însă foarte puține dintre acestea sunt folosite într-un număr semnificativ de orașe și cu atât mai puține în țări diferite. În consecință, la construcţia noului sistem de management şi gestiune a parcărilor se are în vedere această proprietate prin realizarea unei aplicații care să fie accesibilă în țări aflate în diferite regimuri și nivele de dezvoltare și care să se poată adapta în funcție de medile în care sunt folosite.

2.4 Simplitate

Se spune că soluțiile cele mai bune sunt, în general, și cele mai simple. O aplicație bazată pe un astfel de concept va trebuie să fie simplă, rapidă și ușor de folosit, atât pentru utilizator cât și pentru administrator. Aceasta înseamnă că administratorul va putea gestiona cu ușurință parcările iar pentru conducătorul auto, crearea unei rezervări nu va necesita cunoștințe deosebite legate de utilizarea calculatorului.

2.5 Rapiditate

În general, sistemele noi ce le înlocuiesc pe cele vechi au proprietatea de a fi mai rapide. Dacă această nouă abordare a problemei nu permite conducătorului auto să își asigure un loc de parcare într-un timp foarte scurt, atunci există mari șanse ca sistemul să fie considerat un eșec.

3 Rezultate principale

Este o opinie răspândită faptul că, o lucrare prezentată schematic, în care ideile sunt scurte și la obiect, determină ascultătorul/cititorul să fie mult mai receptiv la conceptul descris în interiorul acesteia, decât o prezentare lungă și obositoare. De aceea, în continuare, lucrarea va fi structurată pe un număr reletiv mare secțiuni de lungime medie și mică. Astfel, rezultatele obținute în urma finalizării acestei lucrări pot fi împărțite în două entități diferite:

- Un concept care descrie un nou sistem de management și gestiune a parcărilor.
- O aplicație menită să demonstreze posibilitatea punerii sistemului în practică și ușurința cu care aceasta se poate realiza.

3.1 Descrierea la nivel de concept

Cu toții știm, dupa cum am amintit și în introducere, că una din problemele care apare în momentul în care se folosește autovehicolul ca mijloc de transport, către o zonă aglomerată din punct de vedere al traficului este găsirea unui loc de parcare liber. Acest lucru necesită timp, iar în cazul în care acest gen de activitate se desfășoară zilnic, devine frustrant. Statistic vorbind, o persoană care lucrează într-o zonă aglomerată din centrul unui oraș de dimensiunile Bucureștiului, pierde aproximativ 5-10 minute pe zi căutând un loc de parcare, adică 30-60 minute pe săptămână respectiv 15-30 ore pe an.

Acest nou concept își propune să elimine această perioadă de timp inutilă prin crearea unui sistem care oferă posibilitatea conducătoriilor auto să își rezerve un anumit loc de parcare pe un interval de timp bine definit. Un astfel de sistem nu este chiar nou, el fiind folosit de mult timp în industria hotelieră. Astfel, o simplă adaptare va transforma spațiile de parcare existente într-o mulțime de entități bine organizate, pentru fiecare dintre acestea cunoscându-se, în fiecare moment, disponibilitatea și gradul de ocupare.

Pentru ca acest mod de abordare a problemei să se desfăşoare cu succes a fost nevoie de îndeplinirea următoarelor condiţii:

- ”Colectarea” tuturor parcărilor şi locurilor de parcare aferente într-o singură bază de date.
- Crearea unui suport informatic care să ofere posibilitatea conducătorilor auto, de a face rezervări şi administratorilor, de a-şi efectua cu uşurinţă procesul de gestiune.
- Singura condiţie impusă celor din urmă este de a numerota locurile de parcare, pentru a permite referirea lor atât de aplicaţie cât şi de şoferi.

Astfel se conturează două puncte de vedere diferite şi anume:

3.1.1 Din punct de vedere al administratorului

Să considerăm următorul scenariu:

În centrul istoric al oraşului Braşov, zonă în care traficul este foarte intens şi există foarte multe clădiri importante, se află parcarea ”Gheorghe Dima”. Această parcare, aflată în administraţia primăriei dispune de 100 de locuri de parcare, pentru care a fost stabilită o taxă de 1 leu pe ora de staţionare. În prezent, puţini dintre cei care intră în această parcare găsesc efectiv un loc liber unde să poată staţiona maşina. Majoritatea încearcă la următoarea parcare şi aşa mai departe. Dacă însă, se foloseşte noul sistem de management şi administatorul transformă 50 din cele 100 de locuri de parcare în locuri de parcare cu rezervare, situaţia va arăta total diferit. Şoferii pregătiţi, ce vor să ajungă undeva în zona centrului istoric şi au rezervarea făcută din timp, pot veni liniştiţi, siguri că vor găsi un loc de parcare liber, toată acţiunea fiind organizată.

Am presupus alocarea a doar 50 de locuri din cele 100 pentru rezervări, deoarece este necesară o perioadă de tranziţie între cele două sisteme. Dacă se încearcă direct transformarea tuturor locurilor de parcare, vor exista şoferi mai puţin informaţi care vor intra fără rezervare şi se va ajunge la conflicte şi nemulţumiri. Dar dacă trecerea se face treptat şi se alocă suficient timp pentru informare şi adaptare, probabilitatea încheierii cu succes a acestei tranziţi va fi mult mai mare.

Celelalte aspecte însă vor rămâne neschimbate: autovehiculele parcate ilegal şi cele ce nu au rezervare vor fi ridicate de pe domeniul public, iar taxele de staţionare vor putea fi în continuare aplicate, plata făcându-se la crearea rezervării.

3.1.2 Din punct de vedere al conducătorului auto

Continuând scenariul prezentat mai înainte, să presupunem că domnul Popescu, posesor al unui permis de conducere, categoria B, doreşte să ajungă cu maşina personală în apropierea centrului istoric al oraşului. În condiţiile sistemului actual, este foarte probabil ca domnul Popescu să ajungă în parcarea ”Gheorghe Dima”, să caute fără succes un loc liber timp de 5 minute, iar apoi să îşi continue drumul spre o altă parcare în care va relua procesul de căutare. În schimb, dacă se aplică acest nou concept, domnul Popescu îşi va permite să plece de acasă cu 10 minute mai târziu decât în mod normal, deoarece este sigur că, odată ajuns în parcarea ”Gheorghe Dima”, va găsi la numărul 5, un loc de parcare rezervat numai pentru dumnealui, deoarece a avut grijă să petreacă două minute în faţa calculatorului şi să îşi rezerve acel loc.

După cum se poate observa, scenariile prezentate mai sus, nu sunt extrase din poveşti ştiinţifico-fantastice, ci reprezintă o situaţie tipică în care se află milioane de conducători auto în fiecare zi.

Ţin să menţionez faptul că sistemul prezentat la nivel de concept atinge toate obiectivele prezetate în formularea problemei: accesibilitate, flexibilitate, simplitate, universalitate, rapididate.

3.2 Descrierea aplicației

După cum am precizat și la descrierea conceptului, aplicația atașată acestei lucrări are menirea să demonstreze simplitatea cu care sistemul prezentat poate fi pus în aplicare. De asemenea prin aceasta se demonstrează și cît de realisă și realizabilă este această abordare.

Un prim obiectiv atins de această aplicație este accesibilitatea deoarece constă într-un website ușor de accesat de orice persoană cu conexiune la internet, internet care de altfel a devenit din ce în ce mai abordat de către orice categorie de persoane.

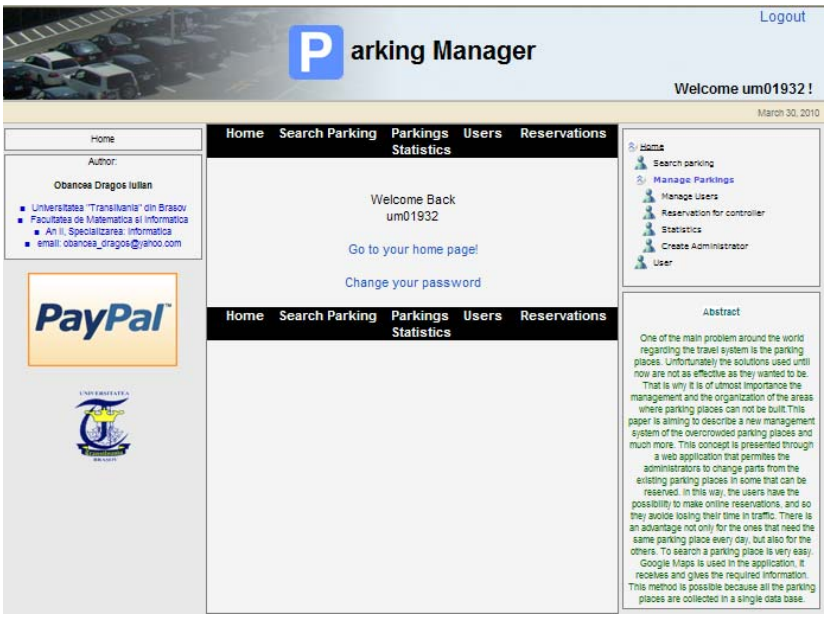
Descrierea aplicației poate fi făcută scurt și la obiect, prezentând succint principalele entități ale acesteia.

3.2.1 Interfața de utilizare

Am optat pentru o interfață utilizator cât mai puțin complicată, deoarece simplitatea este unul din cei mai importanți factori care contribuie la succesul introducerii unui sistem nou. Astfel am reușit să ating un nou obiectiv (simplitate) creând fiecare pagină pe un șablon (”*MasterPage.master*”) care are ca și elemente principale următoarele:

- Un meniu din care pot fi accesate diferite pagini
- O hartă a aplicației
- O scurtă descriere a aplicației
- Unelte de înregistrare și autentificare
- O cale curentă
- Detalii despre autor, data curentă ș.a.m.d.

Conținutul meniului și a hărții aplicației se modifcă în funcție de tipul de utilizator: (anonim, administrator, utilizator).



Menționez că imaginea atașată este orientativă. Aplicația se află în continuă dezvoltare și pot apărea mici modificări, cum ar fi adăugarea unei noi pagini în meniu sau a unei noi unelte în interiorul șablonului.

3.2.2 Modulul pentru utilizator

Modulul pentru utilizator este împărțit în două componente. O primă componentă constă într-o pagină de tipul ”acasă” în care utilizatorul își poate vizualiza/modifica/insera numerele de înmatriculare ale mașinilor personale. Tot în această secțiune sunt afișate toate rezervările făcute de acesta.

Cea de-a doua componentă constă într-o serie de entități structurate liniar ce au rolul de a conduce utilizatorul spre găsirea și rezervarea locului de parcare dorit. Astfel pentru a face o rezervare se vor parcurge următorii pași:

1. Cu ajutorul motorului de căutare și a hărților Google, ce afișează din baza de date, se va găsi și se va selecta parcarea dorită;
2. În cazul în care utilizatorul nu este autentificat i se va cere acest lucru;
3. Se va introduce intervalul de timp în care se dorește a se face rezervarea și în funcție de locurile de parcare disponibile se va alege unul dintre acestea;
4. Urmează o pagină de confirmare în care sunt afișate toate opțiunile selectate de utilizator;
5. Se efectuează plata cu PayPal astfel:
    a. Se face trecerea pe site-ul PayPal la secțiunea plăți;
    b. Se introduc date de autentificare sau date despre persoană și contul bancar;
    c. Se confirmă datele introduse de către utilizator;
    d. Se confirmă plata de către PayPal;
    e. Se revine la aplicația Parking Manager;
6. Se confirmă plata de către aplicație;
7. Se înregistrează rezervarea și se revine la pagina principală a utilizatorului.

Pași prezentați mai sus pot fi parcurși de un utilizator mediu al internetului în aproximativ 3 minute iar în cazul în care persoana în cauză deține și un cont PayPal, plata se face și mai repede. Menționăm că și alte motoare de plăți autohtone pot fi integrate în aplicație (e.g. epayment.ro).

Alegerea lui PayPal s-a făcut pentru că API-ul pus la dispoziție permite simulări în timpul dezvoltării de aplicație și nu în ultimul rând datorită documentației existente. Aplicația confirmă faptul că timpul alocat pentru a rezerva un loc de parcare este mult mai mic decât timpul necesar căutarii efective (zilnice sau nu). Astfel încă un obiectiv propus la formularea problemei (rapiditate) a fost atins.

3.2.3 Modulul pentru administrator

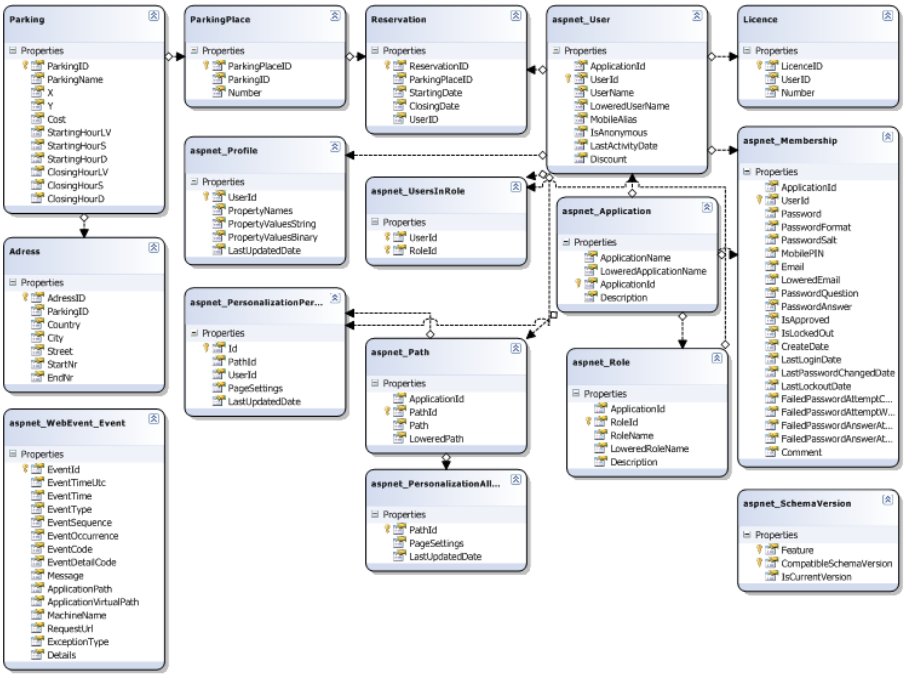
Modulul pentru administrator le oferă acestora o serie de unelte prin care își pot gestiona mai ușor parcările. În continuare amintim o parte dintre acestea:

- Gestionarea conținutului tabelelor din baza de date, adică posibilitatea de a insera, șterge, edita parcări, adrese ale parcărilor, locuri de parcare și rezervările corespunzătoare locurilor de parcare. Menționez că la inserarea parcărilor se folosește Google Maps pentru a determina locația pe hartă a entității obiect;
- Gestionarea utilizatorilor;
- Crearea de noi administratori;
- Crearea a diferite statistici, în mod grafic sau nu, referitor la eficiența sistemului implementat pentru anumite parcări și comparații făcute între toate acestea;
- Crearea unei vederi complexe care să determine toate autovehiculele care au permisiunea de a staționa pe fiecare loc de parcare în parte, structurat pe intervale de timp, care aparțin unei date fixe specificate de către administrator. Acest tabel este destinat controlorilor care pot verifica cu ușurință dacă o mașină este parcată corespunzător din punct de vedere al rezervării, pentru a lua măsurile disciplinare ce

sunt conținute în lege. Figura următoare ilustrează o mică secțiune dintr-un asemenea tabel:

3.2.4 Baza de date

Baza de date este creată în SQL Server 2005 Express Edition ce poate fi folosit gratuit și în cazul unor aplicații comerciale [1, 2]. În continuare voi prezenta printr-o diagramă și o serie de comentarii structura bazei de date și motivele pentru care a fost aleasă o asemenea structură.



- Toate tabelele ce au ca prefix ”*aspnet_*” fac parte dintr-o schemă care este creată automat de *aspnet_regsql*, o aplicație a platformei .NET Framework ce poate fi accesată din Visual Studio 2008 Comand Prompt [3]. Această schemă reprezintă baza

pentru realizarea unei ”*Membership Schema*” necesară aplicației bazate pe utilizator. Singura modificare adusă în această secțiune a fost adăugarea în tabela *aspnet_User* a proprietății *discount* deoarece am considerat că există persoane care sunt scutite de plata taxelor de parcare sau care au reduceri pentru acestea (ex: angajații primăriilor).

- Tabela de la care s-a început dezvoltarea, este *Parking*. Proprietățile acesteia sunt următoarele:
  - ParkingName – numele parcării
  - X, Y – coordonatele pentru Google Map
  - Cost - valoarea taxei de staționare timp de o oră
  - Proprietățile din urmă descriu programul săptămînal al parcării (intervalele de timp în care se percep taxele) pentru: Luni-Vineri, Sâmbătă și Duminică.
- Dimensiunile unei parcări variază, de aceea o parcare cu o suprafață mare poate fi *”întinsă*” pe mai multe adrese. În astfel de condiţii am creat tabela *address*.
- Fiecare parcare are un număr de locuri alocate procesului de rezervare şi din acest motiv a fost introdusă tabela *ParkingPlace*.
- În fiecare loc de parcare există un număr de rezervări pe intervale de timp diferite, care nu se suprapun; şi fiecare rezervare corespunde unui utilizator. Astfel a fost adăugată tabela *Reservation*.
- Tabela *Licence* a fost concepută deoarece orice utilizator poate deține una sau mai multe autovehicule.

Legătura dintre baza de date şi aplicație a fost făcută folosind Linq to SQL [4, 5, 6]. Am ales acest mod de abordare, în detrimentul celorlalte opțiuni, inclusiv a celei automate oferite de platforma .Net Framework 3.5 deoarece putem considera că este un mod eficient din punct de vedere al cantității de cod scrise prin care se poate crea legătura cu un server de baze de date.

3.2.5 Limbajun C#

Limbajul C# a fost preferat datorită faptului că este un limbaj de programare obiect-orientat cu o sintaxă bazată pe C++ care include aspecte ale limbajelor Delphi, Visual Basic şi Java, cu un accent special pus pe simplificare (mai puține simboluri decât în C++, mai puține cerințe decorative decât în Java ș.a.m.d) [8, 9].

Straturile implementate în cadrul aplicației sunt:

- *Business Logic Layer,* format din
  - *Business Objects,* corespunzător lui Domain Model din [7]
  - *Business Services,* corespunzător lui Service Layer din [7]
- *Data Access Layer,* implementat prin Data Mapper [7]

În următoarele secvențe de cod se exemplifică inserarea unui obiect de tip *LicenceBO* în tabela *Licence* a bazei de date:

Clasa LicenseBO:

```
.....
public int LicenceID
{
        get;
        set;
}
public Guid UserID
{
       get;
```

```
        set;
}
public string Number
{
        get;
        set;
}
public override string ValidateMessage()
{
 ........
}
 ........
```

Clasa LicenceServices:

```
 ........
  [DataObjectMethod(DataObjectMethodType.Insert, true)]
   public void InsertLicence(LicenceBO LicenceBO)
 {
    string validationMessage = LicenceBO.ValidateMessage();
    if (validationMessage != string.Empty)
    {
       throw new Exception(validationMessage);
    }
    LicenceDAL LicenceDAL = new LicenceDAL();
    LicenceDAL.InsertLicence(LicenceBO);
  }
 ........
```

Clasa LicenceDAL:

```
 .........
 internal void InsertLicence(LicenceBO LicenceBO)
{
    using (ParcariDBDataContext pdc = new ParcariDBDataContext())
    {
       Licence Licence = new Licence
       {
          UserID = LicenceBO.UserID,
          Number = LicenceBO.Number
       };
       pdc.Licences.InsertOnSubmit(Licence);
       pdc.SubmitChanges();
       LicenceBO.LicenceID = Licence.LicenceID;
    }
 }
 .......
```

3.2.6 Google Maps

Integrarea în aplicație a hărțiilor oferite de cei de la Google a fost unul din considerentele urmărite pentru această aplicație. Folosirea acestei unelte se face atât de către administrator, la introducerea parcărilor cât și de către utilizator la căutarea unui loc de parcare. Utilizarea se face pe baza unui API care permite programatorului să utilizeze hărți oferite de Google Maps într-o aplicație web folosind JavaScript. API oferă o serie de unelte pentru manipularea hărților (asemănătoarea celor de pe pagina http://maps.google.com) și pentru adăugarea a unei mari varietăți de servicii ce duc la realizarea unor hărți robuste pentru proprile aplicații web.

Avantajele oferite de această tehnologie sunt numeroase. Printre acestea se numără:

- Posibilitatea de a introduce pentru o căutare atât o adresă exactă cât şi un punct de reper important (ex: România, Braşov, Primărie);
- Afişarea tuturor parcărilor din baza de date a aplicaţiei împreună cu programul lor;
- Afişarea coordonatelor;
- Posibilitatea de a mări sau micşora imaginea (Zoom);
- Existenţa a trei moduri de vizualizare a hărţii: *Map, Satelite, Hybrid*.
- Un exemplu de interfaţă utilizator Web care foloseşte Google Maps este dat în Figura 1.

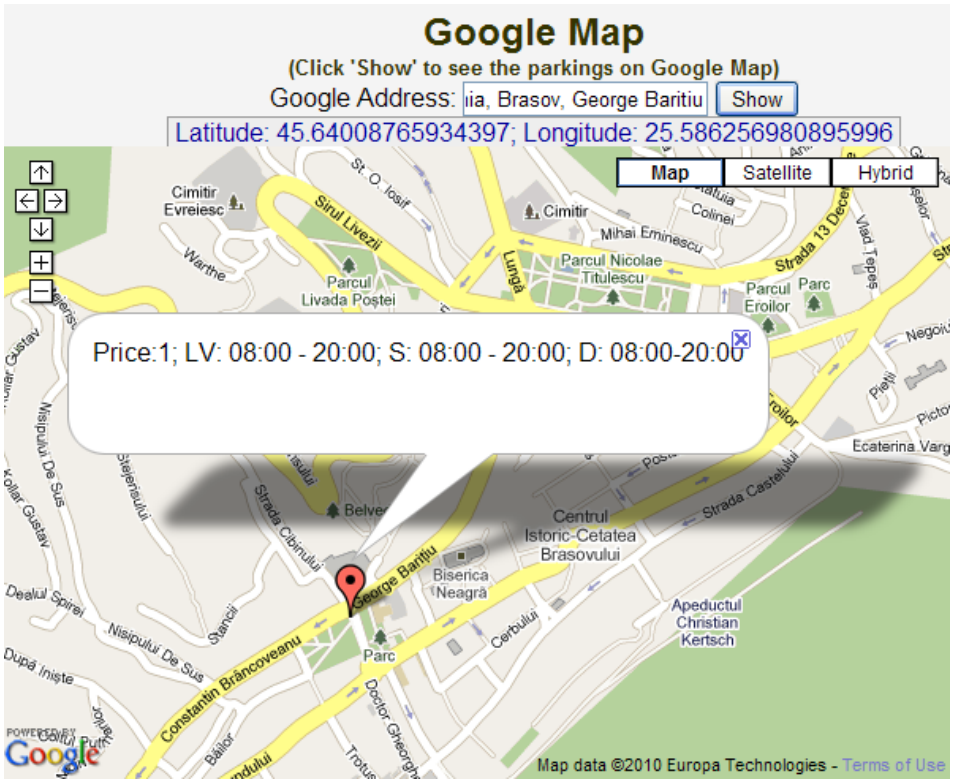


Figura 1. Exemplu de integrare a hărţilor Google Maps in aplicaţie

3.2.7 PayPal

Pentru a realiza plata rezervaţiilor am folosit o tehnologie oferită de cei de la PayPal numită PayPal Sandbox [10, 11]. Aceasta oferă programatorilor posibilitatea de a testa integrarea soluţiei de plată PayPal înainte de a efectua tranzacţiile în mediul de plată real. Sandbox este un duplicat al site-ului PayPal, singura diferenţă dintre aceste două fiind faptul că banii nu se transferă de pe un cont pe altul.

Am ales să folosesc PayPal ca şi metodă de plată pentru că:

- PayPal este unul dintre liderii mondiali în plăţi şi tranzacţii online;
- Este uşor de folosit şi accesibil în majoritatea ţărilor lumii;
- Acceptă plata direct cu cardul sau din contul utilizatorului înregistrat;
- Este rapid şi eficient;

3.2.8 Platforme software

Aplicaţia a fost dezvoltată în Visual Studio 2008 Professional iar baza de date a fost creată cu Microsoft Sql Server 2005 Express Edition. Am facut această alegere având în vedere stabilitatea, viteza de execuţie şi nu în ultimul rând uşurinţa de dezvoltare. Am folosit de asemenea şi biblioteci preluate de la Microsoft, Google, ş.a.m.d. pentru a putea implementa anumite *unelte*: GoogleMap, DataVisualization.Charting, PayPal.

4 Avantaje

Avantajele aplicaţiei realizate sunt numeroase şi diferite pentru cei ce utilizează un astfel de sistem. Cu siguranţă, fiecare dintre noi are un alt punct de vedere asupra unui concept şi prin urmare avantajele oferite de acesta diferă de la o persoană la alta. De aceea, voi prezenta în continuare doar o parte din avantajele pe care le consider ”comune”:

- Costuri mai mici pentru administratori parcărilor dat fiind faptul ca investiţile sunt minime;
- Eliminarea duratei de timp necesare pentru a căuta un loc liber de parcare;
- Confort crescut în ceea ce priveşte efectuarea plăţii rezervărilor de către conducătorii auto;
- Confort crescut în ceea ce priveşte încasarea banilor de către deţinătorii parcărilor;
- O mai bună organizare şi gestionare a parcărilor, respectiv a locurilor de parcare;
- Accesibilitate: Sistemul este accesibil atât administratorilor, datorită costurilor reduse ale investiţilor necesare, cât şi utilizatorilor datorită aplicaţiei care se accesează prin intermediul internetului.
- Universalitate: Sistemul poate fi implementat atât în oraşele ce dispun de resurse materiale mai însemnate, câ şi în cele în care sumele de banii alocate pentru investiţii sunt mai reduse;
- Flexibilitate: În funcţie de condiţiile în care se doreşte dezvoltarea acestui concept, anumite aspecte ale sistemului, respectiv aplicaţiei, pot fi modificate, inclusiv structura bazei de date;
- Simplitate: În general, persoanele sunt mai receptive faţă de ideile simple pe care le pot asimila uşor. Astfel adaptarea conducătorilor auto la un astfel de sistem ar fi mult mai rapidă;
- Rapiditate: Se spune că în ”secolul vitezei” orice concept care aduce rezultate mai rapide este foarte bine primit. De acord, având în vedere că 2-3 minute în faţa calculatorului sunt mai uşor de suportat decât 10 minute în trafic.
- Cu aceste sisteme de parcări pot fi dotate: parcările publice din centrele marilor oraşe, parcările clădirilor administratve, parcările de cartier, parcările hotelurilor de lux, saloanele auto ale reprezentanţelor firmelor producătoare de autoturisme şi aşa mai departe;

5 Concluzii şi dezvoltări viitoare

Atât conceptul cât şi aplicaţia reprezintă o idee nouă, a cărei dezvoltări a ajuns până la un anumit punct. Bine înţeles că ideea nu este fixă, se pot adăuga şi modifica o mulţime de aspecte, sau chiar tot sistemul propus poate fi inclus într-un sistem mai mare, devenind un subsistem al acestuia. Chiar ideea iniţială de la care s-a plecat a fost puţin diferită de rezultatul final obţinut. Am plecat de la o abordare mai simplă în care vroiam doar să creez o aplicaţie în care se puteau face rezervări, iar la confirmarea acestora, utilizatorii primeau un cod pe care îl introduceau în

aparatele de pe marginea parcărilor pentru a primii tichete asemanătoare celor de astăzi. După cum puteți observa aplicația era mai simplă dar sistemul mult prea complex, conținea elemente inutile. Astfel, pe parcursul dezvoltării conceptului am realizat că există o metodă mult mai ușoară, mai puțin costisitoare și mai la îndemână de pus în aplicare folosind numerele de inmatriculare a mașinilor personale ale utilizatorul. De aici, asemenea unor piese de domino, au urmat crearea tabelelor pentru controlori, a statisticilor și așa mai departe.

”Puterea” acestui mod de abordare este dată de posibilitatea continuă de dezvoltare. Astfel poate să țină pasul cu dorința de modernizare din acest sector. Obiectivul propus, de a crea un concept și o aplicație care să poată să crească a fost atins și deja am stabilit o listă de unelte ce pot fi adăugate sistemului și implicit conceptului. Printre acestea se numără:

- Posibilitatea utilizatorului de a-și recupera banii în cazul în care există probleme la locul de parcare rezervat deoarece *”nu putem preveni situațiile neprevăzute”*;
- Posibilitatea utilizatorilor de a avea discounturi în funcție de parcare;
- Posibilitatea selectării unor zile în care să nu se perceapă taxe (ex. în zilele de sărbătoare);
- Posibilitatea determinării clienților fideli și oferirii de bonusuri sub forma de reduceri;
- Posibilitatea de a face reduceri în funcție de parcarea și de timpul dorit pentru rezervare (ex. dacă rezervarea este pe o lună de zile să se facă o reducere de 20%);
- În final, punerea în practică a sistemului.

În concluzie sistemul de management și gestiune a parcărilor prezintă un concept nou, susținut de o aplicație, care reușește să aducă o abordare nouă, simplă și mai ușor de folosit asupra acțiunii de căutare și ocupare a unui loc de parcare. Idea este complet realizabilă și răspunde cerințelor impuse de dezvoltarea sistemului de transport și nu numai.

Bibliografie

[1] ***, *SQL Server 2005 Redistribution Rights*, http://www.microsoft.com/sqlserver/2005/en/us/express-redistribute.aspx
[2] ***, *SQL Server 2005 Express Edition Redistribution EULA*
[3] Matthew MacDonald and Mario Szpuszta, *Pro ASP.NET 3.5 in C# 2008*, Apress, 2007
[4] ***, http://www.asp.net/LEARN/videos/
[5] Fabio Claudio Ferracchiati, LINQ *for Visual C# 2008*, Apress, 2008
[6] Fabrice Marguerie, Steve Eichert, Jim Wooley, *LINQ in Action*, Manning, 2008
[7] Martin Fowler, *Patterns of enterprise application architecture*, Addison Wesley, 2002
[8] Dragoş Iulian Obancea, *Primii paşi cu C#*, Editura Noua Bucureşti, 2009.
[9] Lucian Mircea SASU, *Introducere în limbajul C#,* Editura Universitatii Transilvania din Brasov, 2009
[10] Damon Wiliams, *Pro PayPal E-Commerce (Expert's Voice)*, Apress, 2007
[11] Victoria Rosenborg, Marsha Collier, *PayPal for Dummies*, Wiley Publishing Inc.

DRAGOŞ IULIAN OBANCEA
Universitatea „Transilvania” din Braşov
Facultatea de Matematică şi Informatică
Specializarea Informatică
Strada Iuliu Maniu, Nr. 50, Braşov
ROMÂNIA
obancea_dragos@yahoo.com

Sesiunea Națională de Comunicări Ştiințifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Maze - Joc interactiv

Vasile Nechifor Nicusor, Sorin Radu Daniel
Coordonator: Prof. dr. Dana Simian

Abstract

In this paper we presented an original application which includes algorithms for generating 2D maze, browsing and finding optimal solutions to solve routing problem.

The implementation was done using Mathworks Matlab which gave us an ideal programming environment for development this interactive game.

In the future we want to implement a 3D map of the maze, to improve graphics and algorimii use.

1 Introducere

Jocurile interactive sunt o parte a vieţi cotidiene din zilele noastre. Acestea pot fi simple sau foarte complexe, pornind de la simplul joc Minesweeper şi pană la cele mai avansate jocuri 3D care folosesc tehnologie înaltă de procesare grafică. Acest proiect stă la baza unei idei care a rezultat în urma încercării de a dezvolta un program interactiv într-un mediu în care în general nu se pot dezvolta astfel de proiecte. În această lucrare am prezentat o aplicaţie originală care include algoritmi pentru generarea 2D a labirintului, parcurgerea şi găsirea unei soluţii optime de rezolvare a traseului spre soluţionarea problemei folosind metoda backtracking. Implementarea a fost realizată folosind Mathworks Matlab care ne-a oferit un mediu de programare ideal pentru dezvoltarea acestui joc interactiv.

2 Fundamente teoretice

2.1 Dezvoltarea grafica a jocului

Folosind calculul matricial care este unul dintre avantajele mediului de programare Matlab, am dezvoltat modelul matematic al labirintului astfel:

- se porneşte de la 4 matrici - perete_stanga, perete_dreapta, perete_jos, perete_sus;
- se se initializează cele 4 matrici pentru a defini un “labirint” format numai din celule care au toţi cei patru pereţi;

- cu ajutorul funcţiei *desenare_labirint* se trece la eliminarea pereţilor, atata timp cât sunt celule izolate, adică are toţi cei patru pereţi, apoi se trece la desenarea grafică;
- se trece la parcurgerea labirintului în funcţie de direcţia dorită, direcţie definită cu ajutorul săgeţilor, cu ajutorul funcţiei *parcurgere_labirint.*

2.2 Rezolvarea „problemei labirintului" folosind metoda backtracking

Metoda backtracking foloseşte la rezolvarea multor probleme. Pentru ca o problemă să poată să fie rezolvată cu metoda backtracking, ea trebuie să îndeplinească simultan următoarele condiţii:

- poate avea mai multe soluţii, acestea fiind puse sub formă de vector ca de exemplu $S(x_1,x_2,x_3,...x_n)$, unde $x_1 \in A_1$, $x_2 \in A_2$,..., $x_n \in A_n$

- mulţimile A_1, A_2, A_3, ..., An sunt mulţimi finite, având elementele aflate într-o ordine bine stabilită, ele putând să fie chiar identice.

În acest caz, vectorul care conţine soluţii se numeşte **stivă**.

Se alege primul element x_1 din A_1. Se repetă într-o structură repetitivă până când nu mai există elemente netestate din mulţimea A_1;

- se presupune că am ajuns la elementul x_k din mulţimea A_k şi se doreşte găsirea unui element x_{k+1} din mulţimea A_{k+1}. Acesta trebuie să îndeplinească condiţiile de existenţă a unui astfel de element, impuse de problemă, iar dacă:

- îndeplineşte aceste condiţii atunci trebuie să îndeplinească alte condiţii impuse de problemă, prin care se determină dacă el ar putea face parte din soluţie. Dacă: (1)da, atunci se testează dacă şirul de elemente este o soluţie a problemei; dacă: (2) da, atunci se tipăreşte vectorul care conţine aceste soluţii; (2) nu, atunci se trece pe următorul nivel din şir (k:=k+1); (1) nu, nu îndeplineşte condiţiile de a face parte din soluţie, atunci se trece la următorul element netestat din mulţimea A_k; nu mai poate exista un element pe nivelul k, atunci se trece la nivelul anterior şi se încearcă aici găsirea unui element; repetiţia se termină când am ajuns pe nivelul 0.

2.2 Studiul modalităţilor de implementare

Enunţ: Se dă un labirint sub formă de matrice de m linii şi n coloane. Fiecare element al matricii reprezintă o cameră. Într-una din camerele labirintului se găseşte un om. Se cere să se afle toate soluţiile ca acel om să iasă din labirint, fără să treacă de două ori prin aceeaşi cameră.

Generalizare. Această variantă a problemei este varianta în care fiecare cameră are pereţii proprii în părţile laterale. Există o altă variantă în care fiecare element al matricii este fie un culoar, fie un perete, putându-se trece doar dintr-un culoar în altul. Aici, se poate trece dintr-o cameră în alta, doar dacă între cele două camere nu există perete (camerele sunt imediat apropiate). Prin labirint, putem trece dintr-o cameră în alta doar mergând în sus, în jos, la stânga sau la dreapta, nu şi în diagonală.

Codificare. Principiul backtracking generalizat spune că trebuie codificate direcţiile. În aceste caz vor fi codificate şi combinaţiile de pereţi ai fiecărei camere. Astfel, un element al camerei va fi un element al unei matrici cu n linii şi n coloane, având valori de la 0 la 15. În sistemul binar, numerele 0..15 sunt reprezentate ca 0..1111, fiind memorate pe 4 biţi consecutivi. Vom lua în considerare toţi

cei 4 biţi, astfel numerele vor fi 0000..1111. Fiecare din cei 4 biţi reprezintă o direcţie, iar valoarea lui ne spune dacă în acea direcţie a camerei există sau nu un perete. Vom reprezenta numărul astfel:

nr = b1 b2 b3 b4 (b = bit)

Astfel, b1 indică direcţia nord (sus), b2 indică direcţia est (dreapta), b3 indică direcţia sud (jos) iar b4 indică direcţia vest (stânga).

Valorile unui bit sunt, firește, 0 şi 1. 0 înseamnă că în direcţia respectivă nu există un perete, iar 1 înseamnă că în direcţia respectivă există un perete şi pe acolo nu se poate trece. De exemplu: 0111 - camera aceasta are pereţi în dreapta, în jos şi în stânga, iar în sus este drum liber spre camera vecină. Acest număr este de fapt 7, aşa fiind notat în matricea labirintului.

În ceea ce priveşte direcţiile, vom reţine doar coordonatele unde se află omul din labirint, acestea fiind schimbate în funcţie de drumul pe care-l urmează. Vom avea o stivă triplă astfel:

- coloana 1 va indica direcţia. Aceasta va fi de la 1 la 4, 1 reprezentând sus, 2 reprezentând dreapta, 3 reprezentând jos, iar 4, stânga;
- coloana 2 va reţine indicele de linie al camerelor parcurse în labirint;
- coloane 3 va reţine indicele de coloană al camerelor parcurse în labirint;
- fiecare linie va însemna o cameră, cu indicele de linie şi de coloană.

Astfel, în cazul în care direcţia este 1 (sus), linia se va micşora cu 1, dacă direcţia este 2 (dreapta), coloana se va mări cu 1, dacă direcţia este 3 (jos), linia se va mări cu 1, iar dacă direcţia este 4 (stânga), coloana se va micşora cu 1.

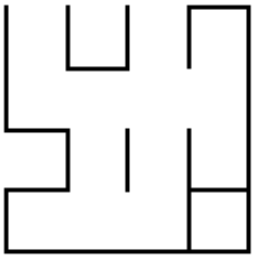
Exemplu: Să presupunem că introducem următoarea matrice:

5 7 5 13

3 8 0 4

14 5 5 7

11 2 6 15

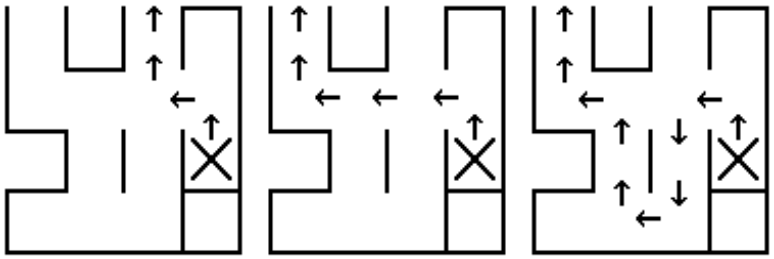


Această matrice corespunde labirintului din desenul de mai sus.

Punctul de pornire din acest labirint este linia 3, coloana 4. Astfel, avea 3 soluţii de a ieşi din labirint, fără a trece de două ori prin aceeaşi cameră:

(3,4)-(2,4)-(2,3)-(1,3)-ieşire

(3,4)-(2,4)-(2,3)-(2,2)-(2,1)-(1,1)-ieşire



(3,4)-(2,4)-(2,3)-(3,3)-(4,3)-(4,2)-(3,2)-(2,2)-(2,1)-(1,1)-ieşire

Rezolvare. După metoda backtracking, trebuiesc găsite toate posibilităţile de a ieşi din labirint. S-a ieşit din labirint când linia este 0, coloana este 0, linia este m+1 sau când coloana este n+1. Trebuie să trecem din cameră în cameră, să verificăm toate posibilităţile de ieşire.

Când trecem dintr-o cameră în alta, reţinem în matricea triplă, direcţia, linia şi coloana respectivă. Va trebui să respectăm regulile preoblemei, adică să nu trecem de două ori prin aceeaşi cameră. Această regulă a fost impusă deoarece dacă am rezolva problema fără ea, atunci ne-am învârti în gol ! Această verificare se face comparând coordonatele camerei în care suntem cu coordonatele camerelor prin care am trecut, cele reţinute de matricea triplă pe coloanele 2 şi 3.

Pentru a verifica dacă între două camere exisă sau nu un perete, va trebui să respectăm un lucru: dacă o cameră are perete în sus, atunci obligatoriu şi camera de sus trebuie să aibă perete jos, iar dacă o cameră nu are perete sus, atunci nici camera de sus nu trebuie să aibă perete jos. Aceasta deoarece dacă ar fi aşa, am trece dintr-o cameră în alta, dar invers nu, ceea ce este imposibil. Imaginaţi-vă că sunteţi într-o cameră. Vreţi să ieşiţi în hol. Din cameră se vede holul şi uşa, dar după ce ieşiţi în hol şi vreţi să reveniţi în cameră, vă întoarce-ţi şi constataţi că în spatele vostru nu există nici o uşă pentru a intra înapoi în cameră, în locul ei fiind doar un perete. Astfel vom verifica după relaţiile:

a[st[k,2],st[k,3]] and 8 = 0 - nu putem merge în sus
a[st[k,2],st[k,3]] and 4 = 0 - nu putem merge la dreapta
a[st[k,2],st[k,3]] and 2 = 0 - nu putem merge în jos
a[st[k,2],st[k,3]] and 1 = 0 - nu putem merge la stânga
a – matricea labirintului st – stiva triplă

Operatorul logic AND verifică biţii corespunzători ai 2 numere întregi, returnând 1 dacă ambii sunt 1 şi 0 în rest. Acesta este rezultatul final. 8 înseamnă 1000, adică dacă camera respectivă are perete sus, rezultatul va fi tot 8, dacă nu, va fi 0.

Când mergem prin labirint, înaintăm în camera respectivă iar apoi verificăm dacă între cele două camere există sau nu perete pentru a putea trece pe acolo. Apoi verificăm să nu fi ieşit din labirint ($l_1<1$ or $l_1>m$ or $c_1<1$ or $c_1>n$) şi dacă prin această cameră am mai trecut odată.

3 Descrierea aplicaţiei

Această aplicaţie prezintă o interfaţă grafică, realizată în Matlab, uşor de folosit, interfaţă ce poate fi lansată cu ajutorul fişierului labirint.fig (nu rulează pe versiuni de Matlab mai vechi de 2008!!!).

După ce datele au fost completate, executant click pe butonul START, este chemată o funcţie maze.m ce are că date de intrare numărul de linii, numărul de coloane şi tipul de generare al labirintuli.

Interfaţa grafică a fost realizată cu ajutorul GUI Builder-ului, pornind de la o fereastă goală (template blank), pe care au fost puse obiectele asupra cărora s-au aplicat diferite proprietăţi şi funcţii.

Fig. 2 Interfata grafica

Interfaţa include două butoane de tip *text box* pentru introducerea dimensiunii labirintului, un meniu pop-up din care utilizatorul alege tipul de generare a labirintului cat şi un buton de *start*.

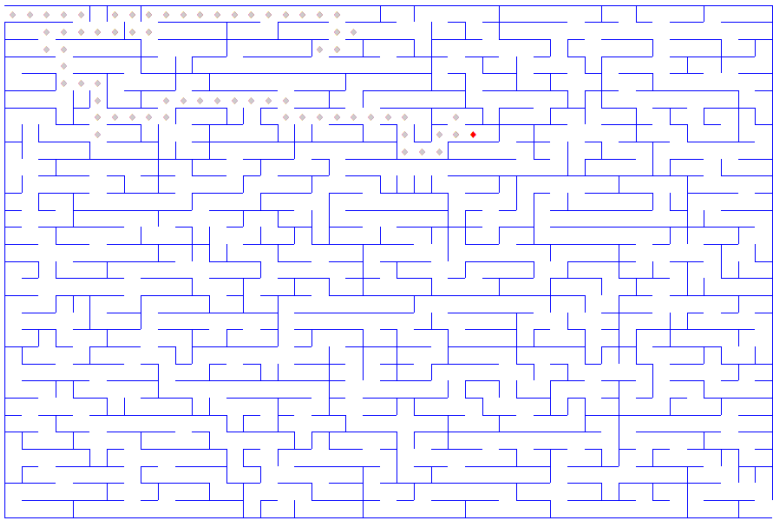


Fig. 3 Labirintul generat

Odată generat labirintul, utilizatorul trebuie să caute soluţia, care este unică deplasându-se prin labirint cu ajutorul săgeţilor. În cazul în care nu găseşte o soluţie, el are posibilitatea de a cere ajutorul programului pentru a găsi drumul spre ieşire.

4 Concluzii

Ca si dezvoltarii viitoare acestui joc interactiv dorim sa-I aducem noi componente cat si o interfata grafica noua.

In concluzie as dori sa specific faptul ca jocurile interactive pot fi dezvoltate in orice mediu fie ele pur matematice sau direct grafic. Utilizarea unui program precum Mathworks Matlab creste probabilitatea de succes a unei incercari de implementare a unui program, fie el joc interactiv, aplicaţie didactică sau aplicaţii de cercetare. Până la urmă toate aceste concepte pot fi implementate în realitatea virtuală care reprezinta o ramura importanta a informatici. Acestea oferă modalitatea prin care calculatorul şi echipamentele specializate modifică modul în care omul percepe realitatea din mediul natural, prin simularea/modelarea unei alte realităţi.

5 Bibliografie

[1] **Dana, Simian**, ***Algoritmi fundamentali şi tehnici de programare***, Editura Universităţii "Lucian Blaga", Sibiu, 2004

[2] **Valer, Roşca,** ***Culegere de exerciţii şi probleme de programare***, Editura Universităţii "Lucian Blaga", Sibiu, 2006

[3] Marin V.,Mircea P., Realitatea Virtuală (Virtual Reality), Tehnologie modernă a informaticii aplicate

[4] www.mathworks.com

[5] www.mathtools.net

VASILE NECHIFOR NICUSOR
Universitatea „Lucian Blaga"
Informatică
Str. dr. I. Ratiu, nr. 5-7, cod 550012, Sibiu
ROMANIA
E-mail: vasilenicusor@yahoo.com

SORIN RADU DANIEL
Universitatea „Lucian Blaga"
Informatică
Str. dr. I. Ratiu, nr. 5-7, cod 550012, Sibiu
ROMANIA
E-mail: radu_s_o_r_i_n@yahoo.com

Sesiunea Națională de Comunicări Ştiințifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Agenda electronica

Raluca Pandaru
Coordonator: Lect. univ. dr. Lucian Sasu

Abstract

This project is developed in C# and SQL Server 2005 and its main purpose is to help people organize their time more efficiently. Thus, the user can record various activities, determine which are the planned activities from a specified date and use the facilities of this application in order to plan some events. The project can automatically plan activities which are introduced by the user so that the resulted planning respects all the constrains imposed by the user.

1 Introducere

Algoritmii genetici sunt folositi pentru rezolvarea unor probleme pentru care nu se cunoaste un algoritm care sa ruleze in timp polinomial. Aceste probleme au pe langa datele de intrare numeroase conditii care trebuie indeplinite de solutia rezultata.

In acest sens, putem considera planificarea timpului ca fiind o problema dificila pe care trebuie sa o rezolvam zilnic, o problema a carei solutie trebuie sa satisfaca numeroase conditii. De aceea, m-am gandit sa realizez un proiect care sa ajute la planificarea timpului. Datorita conditiilor care trebuie indeplinite, am folosit un algoritm genetic care sa determine solutia.

In plus, aplicatia permite inregistrarea de noi activitati si vizualizarea activitatiilor inregistrate.

2 Prezentarea proiectului

Pentru realizarea proiectului am folosit limbajul C# datorita faptului ca acest limbaj de programare este puternic orientat pe obiecte. Programarea orientata pe obiecte este utila pentru acele aplicatii care vor fi dezvoltate in viitor pentru ca presupune efort minim, utilizandu-se ceea ce a fost programat anterior. In plus, pentru inregistrarea activitatilor am folosit SQL Server 2005.

Aceasta prezentare va descrie modul in care aplicatia va planifica activitatile care sunt introduse de catre utilizator.

3 Formularea problemei planificarii timpului

Cerinta:
Fiind date un interval de timp, o multime de evenimente si un set de restrictii, se pune problema gasirii unei planificari cat mai eficiente (timpul sa fie ocupat cat mai bine) a evenimentelor astfel incat sa fie respectate toate conditiile impuse.

Date de intrare:
- fie un interval de timp T dat si E multimea de activitati care trebuie planificate in intervalul T, precum si o multime de intervale de timp R care contine intervalele de timp rezervate
- presupunem ca multimea E contine n evenimente $E=\{e_1, e_2,..., e_n\}$
- fiecare eveniment e_i din multimea E este caracterizat de urmatoarele atribute:
    - o scurta descriere a evenimentului
    - o anumita complexitate care este data de utilizator (aceasta complexitate este un numar de la 1 la 10; ea este aproximativa si este data prin comparatie cu celelalte evenimente din multimea E; aceasta este folosita pentru programarea in timp a evenimentelor)
    - evenimentele de care depinde (este posibil ca un anumit eveniment sa poate fi realizat numai daca a fost indeplinit alt eveniment)
    - un interval de timp $t_i=[t_i^1, t_i^2]$ in care se doreste sa se planifice evenimentul (se poate specifica numai t_i^1 (e_i va fi programat dupa t_i^1) sau numai t_i^2 (e_i va fi planificat astfel incat sa fie realizat inainte de t_i^2); trebuie specificat faptul ca introducerea intervalului t_i nu inseamna ca evenimentul e_i se va desfasura din t_i^1 pana la t_i^2, ci se va desfasura intr-un interval de timp t_i' care este inclus in t_i; acest lucru nu restrange generalitatea problemei pentru ca se pot introduce intervale de timp reservate, adica in care nu se pot face planifiari)
    - un eveniment e_j care va fi planificat imediat dupa evenimentul e_i
    - intervale de timp in care evenimentul nu poate fi planificat
- multimea $R =\{r_1, r_2,..., r_m\}$ contine intervalele de timp in care nu se pot face planificari (adica niciun eveniment nu poate fi planificat in intervalul de timp respectiv), unde r_i este strict inclus in T.

Date de iesire
- o planificare a evenimentelor care sa respecte conditiile impuse prin datele de intrare

Trebuie specificat faptul ca planificarea evenimentelor se va face astfel incat un eveniment sa nu fie divizat. In plus, daca un interval t_i este inclus intr-un interval r_j, algortimul nu va gasi solutie. Adica, obtinerea solutiei depinde de corectitudinea formularii datelor de intrare.

4 Prezentarea algoritmului genetic folosit

Algoritmul genetic folosit are forma simplificata:

```
crearea populatiei initiale;
for (int nr_iteratie = 0; nr_iteratie < 7000; nr_iteratie++)
{
        for (int i = 0; i < 100; i++)
        {
                calcularea valorii de fitness a individului i;
                if (fitnessi == 1)
                {
```

```
                                    am gasit solutie;
                                    iesire din cele 2 cicluri for;
}
            }
selectarea indivizilor care se vor inmulti si a celor care vor fi inlocuiti;
aplicarea operatorului crossover;
aplicarea operatorului unar mutation;
 }
```

Fiecare populatie este formata din 100 de indivizi, un individ este reprezentat printr-un vector a carui dimensiune este data de numarul de evenimente care vor fi planificate, iar evenimentele sunt codificate folosind numere intregi intre 0 si numarul de evenimente - 1. Conditiile impuse evenimentelor sunt stocate intr-o instanta a unei clase Event care codifica un eveniment. Aceste instante sunt folosite pentru a calcula valoarea de fitness a unui individ.

Dupa cum se poate observa, inainte de aplicarea algoritmului genetic se construieste populatia initiala. Aceasta trebuie sa fie cat mai diversificata pentru a creste probabilitatea de a gasi o solutie intr-un timp cat mai scurt. In plus, pentru micsorarea timpului de executie, populatia initiala va fi alcatuita numai din indivizi care respecta conditiile legate de dependenta a unor evenimente de alte evenimente. Pentru construirea populatiei initiale am folosit un algoritm de backtracking optimizat, in sensul ca am impus conditii care elimina functia de verificare si care garanteaza ca fiecare pas k al algoritmului va construi o solutie valida. Tot cu ajutorul acestui algoritm se poate determina, inainte de lansarea in executie a algoritmului genetic, daca exista dependente circulare intre evenimente.

Pentru implementarea functiei de selectie este necesar calculul valorii medii de fitness a polpulatiei si a valorii de fitness minime. Vor fi selectati pentru a face schimb de material genetic toti indivizii care au valoarea de fitness mai mare sau egala cu valoarea medie de fitness a populatiei. In schimb, pentru inlocuire vor fi selectati acei indivizi care au valoarea de fitness mai mica sau egala cu (valoarea de fitness minima + valoarea medie de fitness a populatiei) / 2. Dar, dintr-o populatie nu pot muri mai mult de 20% dintre indivizi. Daca sunt mai mult de 20% de indivizi din totalul populatiei selectati pentru inlocuire atunci se vor alege aleator indivizii care vor fi inlocuiti. In acest fel, se evita constructia unei populatii care sa contina de mai multe ori un individ care are o valoare de fitness mare, dar care nu poate conduce la gasirea solutiei pentru problema data.

Operatorul de crossover folosit este cel propus de Syswerda (detalii in [1] capitolul 10 pagina 219), numai ca numarul de pozitii generate este egal cu lungime_individ div 2. Iar operatorul mutation presupune interschimbarea a doua evenimente intre ele. Acesta se aplica unui singur individ astfel: se parcurg toate evenimentele in ordinea in care sunt stocate de individ si pentru fiecare eveniment se genereaza un numar in cuprins intre 0 si 1. Daca numarul generat este mai mic de 0.001 atunci evenimentul respectiv este interschimbat cu alt eveniment ales arbitrar.

In continuare, voi prezenta un exemplu abstract pentru a arata modul de executie al algoritmului.

Sa presupunem ca intervalul de timp este 04/01/2010 7:30 AM – 04/10/2010 22:30 PM, multimea de evenimente este {ev0, ..., ev24}, iar multimea de intervale rezervate este multimea vida.

Pentru fiecare eveniment se introduc atributele acestuia:
Evenimentul 1
- descriere: ev0
- complexitate: 3
- evenimente de care depinde: ev20
- limita de timp superioara: 04/07/2010 23:30 PM

Evenimentul 2

descriere: ev1
complexitate: 4
evenimente de care depinde: ev4, ev15
limita de timp inferioara: 04/05/2010 7:30 AM
limita de timp superioara: 04/07/2010 23:30 PM

Evenimentul 3
descriere: ev2
complexitate: 5
limita de timp inferioara: 04/03/2010 7:30 AM
limita de timp superioara: 04/07/2010 23:30 PM

Evenimentul 4
descriere: ev3
complexitate: 7

Evenimentul 5
descriere: ev4
complexitate: 1

Evenimentul 6
descriere: ev5
complexitate: 3
evenimente de care depinde: ev16

Evenimentul 7
descriere: ev6
complexitate: 1

Evenimentul 8
descriere: ev7
complexitate: 3
evenimente de care depinde: ev10

Evenimentul 9
descriere: ev8
complexitate: 2
limita de timp inferioara: 04/02/2010 12:30 PM
limita de timp superioara: 04/02/2010 22:30 PM

Evenimentul 10
descriere: ev9
complexitate: 1

Evenimentul 11
descriere: ev10
complexitate: 4

Evenimentul 12
descriere: ev11
complexitate: 5

Evenimentul 13
descriere: ev12
complexitate: 7

Evenimentul 14
descriere: ev13
complexitate: 8

Evenimentul 15
descriere: ev14
complexitate: 9
evenimente de care depinde: ev13

Evenimentul 16
descriere: ev15
complexitate: 1
evenimente de care depinde: ev11, ev14
Evenimentul 17
descriere: ev16
complexitate: 1
evenimente de care depinde: ev10
Evenimentul 18
descriere: ev17
complexitate: 2
Evenimentul 19
descriere: ev18
complexitate: 1
limita de timp inferioara: 04/09/2010 8:45 AM
limita de timp superioara: 04/10/2010 8:50 PM
Evenimentul 20
descriere: ev19
complexitate: 10
Evenimentul 21
descriere: ev20
complexitate: 4
Evenimentul 22
descriere: ev21
complexitate: 6
evenimente de care depinde: ev18
Evenimentul 23
descriere: ev22
complexitate: 2
Evenimentul 24
descriere: ev23
complexitate: 4
Evenimentul 25
descriere: ev24
complexitate: 2

O solutie gasita de algoritm este:
ev20 [4/1/2010 7:30 AM, 4/1/2010 5:13 PM]
ev11 [4/1/2010 5:13 AM, 4/2/2010 5:23 AM]
ev10 [4/2/2010 5:23 AM, 4/2/2010 3:06 PM]
ev6 [4/2/2010 3:06 PM, 4/2/2010 5:32 PM]
ev8 [4/2/2010 5:32 PM, 4/2/2010 10:24 PM]
ev13 [4/2/2010 10:24 PM, 4/3/2010 5:51 PM]
ev2 [4/3/2010 5:51 PM, 4/4/2010 6:00 AM]
ev14 [4/4/2010 6:00 AM, 4/5/2010 3:53 AM]
ev7 [4/5/2010 3:53 AM, 4/5/2010 11:11 AM]
ev4 [4/5/2010 11:11 AM, 4/5/2010 1:37 PM]
ev19 [4/5/2010 1:37 PM, 4/6/2010 1:56 PM]
ev16 [4/6/2010 1:56 PM, 4/6/2010 4:22 PM]
ev0 [4/6/2010 4:22 PM, 4/6/2010 11:44 PM]
ev9 [4/6/2010 11:44 PM, 4/7/2010 2:05 AM]

ev15 [4/7/2010 2:05 AM, 4/7/2010 4:31 AM]
ev1 [4/7/2010 4:31 AM, 4/7/2010 2:15 PM]
ev24 [4/7/2010 2:15 PM, 4/7/2010 7:07 PM]
ev22 [4/7/2010 7:07 PM, 4/7/2010 11:59 PM]
ev17 [4/7/2010 11:59 PM, 4/8/2010 4:50 AM]
ev12 [4/8/2010 4:50 AM, 4/8/2010 9:52 PM]
ev23 [4/8/2010 9:52 PM, 4/9/2010 7:35 AM]
ev5 [4/9/2010 7:35 AM, 4/9/2010 12:27 PM]
ev18 [4/9/2010 12:27 PM, 4/9/2010 5:53 PM]
ev21 [4/9/2010 5:53 PM, 4/10/2010 5:28 AM]
ev3 [4/10/2010 5:28 AM, 4/10/2010 10:29 PM]
Solutia a fost gasita in cadrul generatiei 3927.
Timp de rulare: 00:00:52.53

5 Concluzii

Agenda electronica este o aplicatie care vine in sprijinul celor care doresc sa-si organizeze timpul si ofera posibilitatea planificarii automate a anumitor evenimente, pe baza restrictiilor impuse de utilizator.

Pentru viitor, imi propun sa optimizez algoritmul de planificare a evenimentelor si sa adaug functii noi care sa ajute la organizarea eficienta a timpului pentru a creste utilitatea aplicatiei.

Bibliografie

[1] Genetic Algorithms + Data Structures = Evolution Programs, Zbigniew Michalewicz
[2] diferite carti in format eBook gasite pe internet

Raluca PANDARU
Universitatea Transilvania Brasov
ROMANIA
raluca_pandaru@yahoo.com

Sesiunea Națională de Comunicări Științifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

CineStar – a movie application

Popa Bogdan Constantin
Coordonator: prep. univ. Baicoianu Alexandra

Abstract

The Movie Industry is one of the most exciting and informative business in the world, over 70% of the population goes to cinema or watches a movie at home. Each of us searched on Internet at least one time looking for a specific movie, or for getting in touch with the latest releases. This can be time consuming in our lifes. So why not having an application that does everything with a push of a button? CineStar is the answer, news, information, trailers, cinema schedules and locations, subtitles and downloads right away. You don't have to bother anymore about searching all over the Internet for a specific content, CineStar does all that you need faster and better.

1 Introducere

De mai bine de jumatate de secol industria de entertaiment s-a dezvoltat foarte rapid astfel incat sa fie prezenta aproape peste tot. Acest segment face parte din viata fiecarui om. Astfel nu este de mirare ca o mare parte din petrecerea timpului liber il dedicam filmelor. Astazi, indiferent in ce parte a lumii ne-am afla, daca am intreba la intamplare pe oricine am intalni pe strada: „Ce preferati sa faceti in timpul liber?", nu ar trebui sa ne mire sa constatam ca o majoritate covarsitoare ar spune ca: „Prefer sa ma uit la un film". Industria cinematografica pare sa detina monopolul peste timpul liber al oamenilor. Variatatea de genuri si stiluri face ca cinematografia sa se refere la o gama larga de preferinte ale iubitorilor de film. Daca „unora le place jazzul", celor mai multi dintre noi ne plac filmele.

Multi dintre cinefili prefera sa se informeze pe internet iar intalnindu-se cu situatia in care cauta informatii despre un anumit film, cauta un trailer, o subtritrare sau chiar filmul. Sau vrea sa vada unde poate viziona filmul respectiv, la care cinematograf. In a cauta ne ajuta probabil Google. Dar nu ar fi mai bine sa avem o aplicatie care sa le faca pe toate? Sa sarim peste toate etapele in care dam click-uri peste click-uri si tot nu dam peste o informatie utila si filtrata.

Acest proiect indeplineste dorintele multor dintre cinefili, CineStar propune un sistem rapid de cautare in domeniul filmului. Doar prin cautarea denumirii filmului, aplicatia ofera utilizatorului posibilitatea de a vedea multiple informatii, sa vizioneze un trailer, sa vada locatia si orarul cinematografului care ruleaza filmul, sa descarce subtritrarea sau filmul respectiv. Pentru a avea un pachet complet, CineStar ofera si posibilitatea de a afisa cele mai noi stiri. Astfel aplicatia devine o adevarata „unealta" in mainele celui care o utilizeaza.

Dezvoltarea aplicatiei s-a facut in Visual Studio 2008, Microsoft Expression Blend 3 folosind Windows Presentation Foundation (WPF) alaturi de limbajul C#. Windows Presentation

Foundation ofera un model de programare unificat pentru construirea aplicatiilor Windows de tip „smart client”. Permite grafica 2D si 3D si foloseste XAML, un limbaj declarativ, subset al XML pentru a instantia obiecte. XAML a fost dezvoltat in paralel cu WPF. Am ales WPF deoarece ofera un plus in construirea interfetei vizuale, permitand folosirea diferituri stiluri, template-uri si animatii dand in acest fel un „look and feel” mai deosebit.

De asemeni, in realizarea aplicatiei un punct cheie a fost crearea unei baze de date cat mai bine pusa la punct. Aceasta aplicatie foloseste serverul de baze de date MySQL, fiind in acelasi timp si unul dintre cele mai populare sistem de management pentru baze de date relationale. In baza de date MySQL datele sunt stocate in mai multe tabele separate,fiind imbunatatita viteza si flexibilitatea. Prin serverul de MySQL datele se pot accesa foarte rapid, iar baza de date poate fi intretinuta mult mai usor.

Aplicatia CineStar este alcatuita din mai multe module, situate pe o forma principala, astfel interactionarea cu utilizatorul se face mult mai usor. De multe ori limba afisata de aplicatie poate constitui o problema pentru utilizator atfel am dorit sa implementez o interfata in care se poate alege limba de afisare. In acest moment in aplicatie se poate alege intre urmatoarele limbi: romana,engleza si italiana.

In continuare, in sectiunea urmatoare vom analiza structura aplicatiei, functiile acesteia si modul de operare.

2 Structura aplicatiei

2.1 Interfata vizuala

Intr-o aplicatie o foarte mare importanta o are interfata vizuala, aceasta trebuie sa fie cat mai intuitiva intrucat utilizatorul sa poata folosi functiile acesteia pentru a-si indeplini utilitatea. In continuare vom detalia structura interfetei vizuale pentru care am optat. In figura 1 se observa principalele componente ale aplicatiei CineStar.

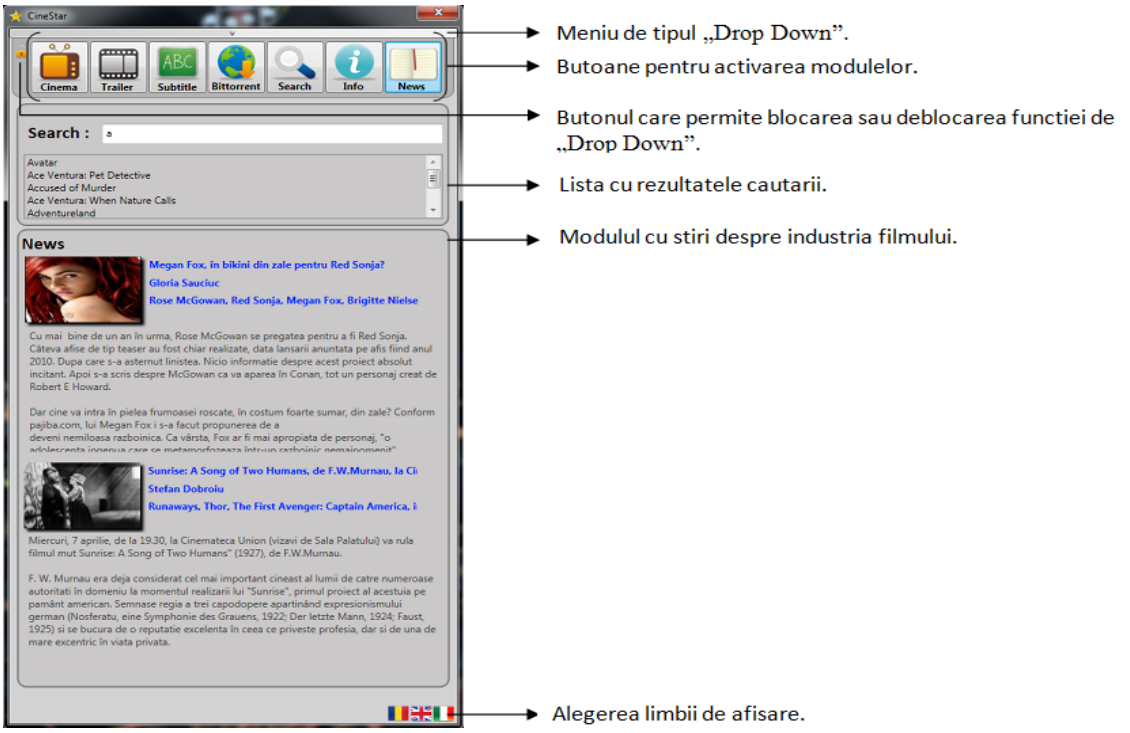


Fig. 1

Dupa cum se vede in figura 1, aplicatia are urmatoarele componente principale:

- Meniul de tip „Drop-Down” care contine butoanele pentru activarea modulelor.
- Modulul „Search” in care se cauta filmul dorit.
- Modulul „News” in care se afiseaza stiri despre industria filmului.
- Modulele „Cinema”, „Trailer”, „Subtitle”, „Bittorrent” si „Info” care pot fi activate dupa o cautare.
- Limba folosita pentru afisare.

In urmatoarele subsectiuni vom discuta despre fiecare dintre aceste componente.

2.1.1 Meniul

Meniul este format dintr-un control de tipul grid, acesta continand butoanele cu care se interactioneaza in forma principala. Pentru a creea efectul „drop-down”, meniului i-au fost atasate metode prin care se verifica daca butonul unlock/lock a fost activat si in consecinta el va afisa continutul sau il va ascunde.

Metoda care ascunde elementele meniului, la parasirea mouse-ului din zona acestuia:

```
private void Grid_MouseLeave(object sender, System.Windows.Input.MouseEventArgs e) {
        if (!lck){
            button1.Content = ">";
            button2.Visibility = Visibility.Hidden;
            button3.Visibility = Visibility.Hidden;
            button4.Visibility = Visibility.Hidden;
            button5.Visibility = Visibility.Hidden;
            button6.Visibility = Visibility.Hidden;
            button7.Visibility = Visibility.Hidden;
            button8.Visibility = Visibility.Hidden;
            rectangle1.Visibility = Visibility.Hidden;
            lock_image.Visibility = Visibility.Hidden;
            Thickness thick = new Thickness(4.5, 15, 1.5, 3);
            grid2.Margin = thick;
            ytubeplayer.Visibility = Visibility.Hidden;
            ytubeplayer.Visibility = Visibility.Visible;
        }

    }
```

In meniul aplicatiei, la apasarea unui element se va crea o lista de obiecte. Aceste obiecte vor fi afisate pe forma in ordinea apasarii acestora si in functie de dimensiunea ferestrei. Pentru a avea acesta posibilitate am contruit urmatoarele metode:

```
private List<Grid> x = new List<Grid>();
private void move(object sender) {
   if (!x.Contains(((Grid)sender)))
      x.Add(((Grid)sender));
   else{
      x.Remove(((Grid)sender));
      x.Add(((Grid)sender));
   }
   double height=0;
   for (int i = 0; i < x.Count(); i++)
      height += x[i].Height;
   while (height > 600) {
      height -= x[0].Height;
```

```
            x.RemoveAt(0);
        }
        Thickness thick = new Thickness(0, 160, 3, 0);
        for (int i = x.Count() - 1; i > -1; i--)
        {
            x[i].Visibility = Visibility.Visible;
            x[i].Margin = thick;
            thick.Top = thick.Top + x[i].Height + 4;
        }
    }
    private void setVisible(object sender)
    {
        grid_news.Visibility = Visibility.Hidden;
        grid_info.Visibility = Visibility.Hidden;
        grid_cinema.Visibility = Visibility.Hidden;
        grid_bittorrent.Visibility = Visibility.Hidden;
        grid_subtitle.Visibility = Visibility.Hidden;
        grid_trailer.Visibility = Visibility.Hidden;
        ((Grid)sender).Visibility = Visibility.Visible;


    }
```

2.1.2 Componentele „News”, „Info”, „Subtitle” , „Bittorrent” , „Trailer”

Aceste componente, cu exceptia „News” pot fi activate dupa ce se efectueaza o cautare. Componenta „News” se poate activa indiferent daca s-a facut o cautare sau nu, aceasta va afisa informatii cu privire la ultimele stiri in legatura cu industria filmului. „Info” afiseaza informatii cu privire la filmul cautat: tipul filmului, un sumar al actiunii, regizorul, scriitorul etc. Acesta componenta se poate vizualiza in figura 2. Componenta „Subtitle” va oferi linkuri catre o subtritrare a filmului, acest link se va deschide automat in browserul default si va va propune sa salvati subtritrarea. Modul de operare al componentei „Bittorrent” este asemanator cu al celei de mai sus,diferenta constand in faptul ca la salvare, vi se va oferi un fisier .torrent care se poate utiliza printr-o aplicatie client de tip torrent pentru a descarca filmul. Componentele „Subtitle” si „Bittorrent” se regasesc in figura 3. Modulul „Trailer” gazduieste continut media furnizat de youtube si ne ofera posibilitatea de a putea viziona secventa din filmul cautat. Acesta se regaseste tot in figura 3.

Pentru fiecare dintre aceste module am folosit cate o clasa separata pentru a conecta aplicatia de baza de date MySQL. Un exemplu de clasa este clasa Info.cs :

```
class Info{
    public String rating, director, writer, genre, plot, awards, cast,img;
    private String search;
    public Info(String search) {
        this.search = search;
        getInfo();}
    private void getInfo(){
        QueryDB qdb = new QueryDB(search);
        MySqlDataReader reader = qdb.info();
        while (reader.Read())
        {
            img = reader["picture"].ToString();  rating = reader["rating"].ToString();
            director = reader["director"].ToString();  writer = reader["writer"].ToString();
            genre = reader["genre"].ToString();  plot = reader["plot"].ToString();
            awards = reader["awards"].ToString();  cast = reader["cast"].ToString();
        }
        reader.Close();
    }
```

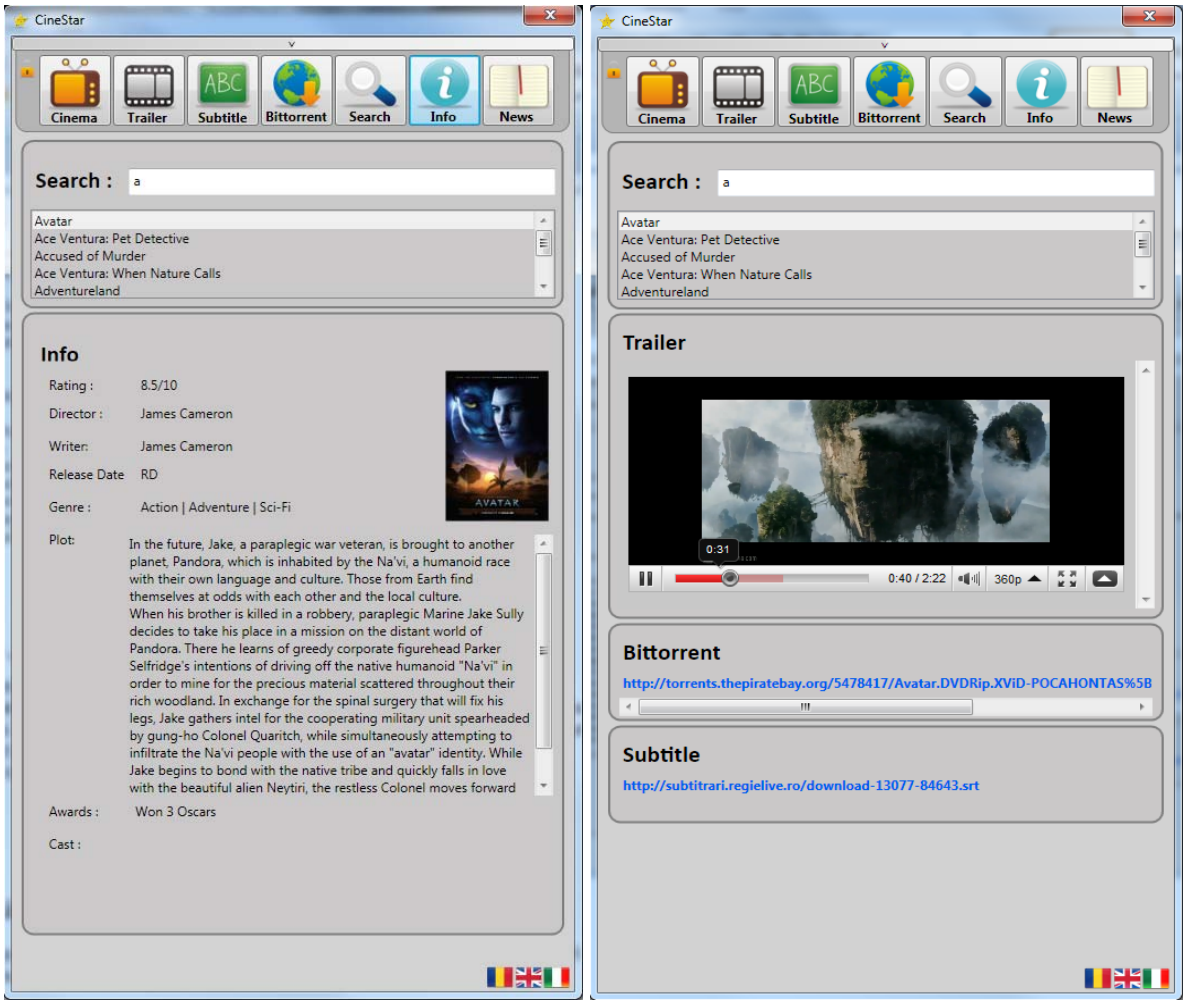


Fig.2 si Fig.3

2.1.3 Componenta „Cinema”

Aceasta componenta este una dintre cele mai importante din aplicatia de fata. „Cinema” are scopul de a ne furniza informatii in legatura cu locul unde se ruleaza filmul. In plus putem viziona si programul dupa care se ruleaza filmul in respectivul cinematograf. Presupun ca nu de putine ori v-ati intrebat unde anume este pozitionat acel cinematograf si cum puteti sa ajungeti pana acolo. Din aceasta cauza am integrat serviciile „Bing Maps” si „Google Maps”. Prin aceste doua componente vedem pozitia cinematografului dar si putem sa facem un traseu de acasa pana la destinatie. Componenta „Cinema” se poate vizualiza in figura 4, alaturi de care se afla si ferestrele care afiseaza continut „Bing Maps” si „Google Maps”.

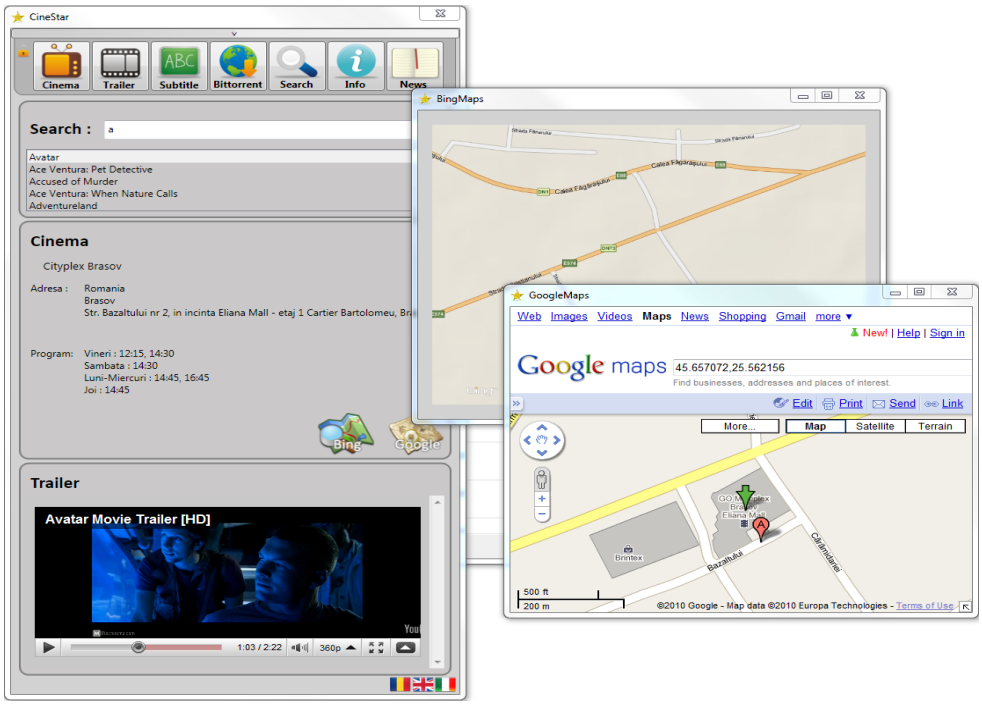


Fig. 4

2.2 Baza de date MySQL

In aceasta aplicatie, baza de date MySQL are rolul de a gazdui datele despre fiecare film. Prin aceasta metoda informatiile pot fi accesate foarte rapid si de oriunde, nefiind necesar pastrarea bazei de date ca resursa locala, aceasta poate rula pe un server MySql si poate fi accesata prin conexiunea la internet.

Stocarea in baza de date a informatiilor se face in urmatorul fel: exista o tabela denumita „movie”, in acesta tabela se inregistreaza denumirile filmelor alaturi de un id unic si o eventuala poza optionala. Pe langa aceasta tabela principala, exista tabelele „info”, „subtitle”, „trailer”, „bittorrent”, „cinema”. Pentru a face legatura dintre aceste tabele folosim id-ul unic din tabela movie. In acest fel atunci cand cautam un anumit film, ne folosim de id pentru a retrage informatiile.

In figura 5 se regaseste diagrama tabelelor, se pot vedea campurile fiecarui tabel alaturi de tipul de date folosit.

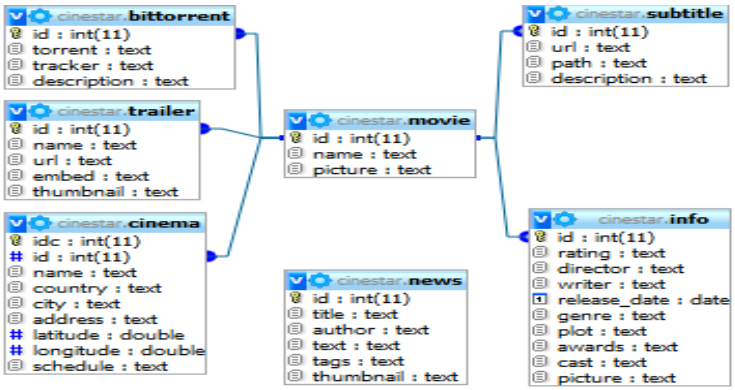


Fig. 5

3 Concluzii si posibile dezvoltari viitoare

Pornind de la necesitatea unei modalitati de a avea un pachet intreg de functii care sa ajute in cautarea de informatii despre un anumit film, am dezvoltat aplicatia „CineStar”. Aceasta inglobeaza toate necesitatile unui cinefil intr-un singur loc,scurtand timpul de cautare si furnizarea unor informatii corecte.

In dezvoltarile viitoare urmaresc sa extind functiile modulului „Cinema” intrucat acesta sa aiba o componenta prin care se poate cumpara sau rezerva un bilet la un cinematograf ales. De asemenea, doresc implementarea unui sistem pentru utilizator,prin acest sistem acesta va putea marca filmele favorite si se va putea loga in contul propriu. O alta componenta care va fi extinsa este cea de alegera a limbii,aici se vor implementa traducerii pentru zona EMEA.

Bibliografie

[1] Lucian Sasu, *Limbajul C#*
[2] http://msdn.microsoft.com/en-us/library/dd221354.aspx
[3] http://www.pantz.org/software/mysql/mysqlcommands.html
[4] http://blogs.msdn.com/devkeydet/archive/2008/06/24/wpf-and-virtual-earth-revisited.aspx
[5] http://www.vbdotnetheaven.com/UploadFile/scottlysle/VBQuickMap03182007085423AM/VBQuickMap.aspx
[6] Andrew Troelsen, *Pro C# with .NET 3.0*

POPA BOGDAN CONSTANTIN
Universitatea Transilvania,Facultatea de Matematica si Informatica
Specializarea Informatica Aplicata
Str. Iuliu Maniu, nr. 50, Cod postal: 500091, Brasov
ROMANIA
bogdi@ymail.com

Sesiunea Naţională de Comunicări Ştiinţifice a Studenţilor
"Imaginaţie, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Modelarea şi simularea fenomenelor care apar la nivelul Poligonului Willis

Cristina Elena Roman
Vlad Monescu

Abstract

The present paper presents an important current theme from the medical world and also whishes to facilitate the work in the field of researching the phenomena that appear inside the human brain. The specific part from the brain that is studied here is the Circle of Willis: An arterial circle at the base of the brain that is of critical importance. Based on this structure and on a mathematical model developed by a group of scientists, a software product was elaborated basically to simulate the blood flow inside the polygon and to determine the pressure and the flow capacity from the nodes and arteries. The software can obtain data based on the normal situations that appear inside the Circle of Willis but also can study the mutations of the structure on different subjects. The specific purpose of the software is to help the doctors to choose the safest medical procedure for a patient based on the obtained results and also to help the researchers to investigate and know better the behavior of the Circle. The crucial importance of the structure and the very few people that study this field gives proof of the paper importance and originality.

1 Introducere

Datorită continuei evoluţii în lumea medicala s-a făcut dorită prezenţa produselor soft care să îmbunataţească şi care să ajute la o dezvoltare mai rapidă şi în acest domeniu. În acest scop a fost realizat un produs soft care, pe baza datelor medicale introduse - date obţinute în urma colaborării cu Dr. Cristian Falup-Pecurariu de la Spitalul de Urgenţă Braşov, Secţia de Neurologie - rezultate din măsurători pe creiere umane ale unor pacienţi decedaţi, simulează diverse situaţii care apar în interiorul Poligonului Willis.

Poligonul lui Willis sau Cercul lui Willis, numit astfel după fizicianul englez Thomas Willis (1621-1673), reprezintă o structură anatomică de artere care irigă creierul. Un alt rol important al acestei configuraţii este şi cel de reglare a presiunii sângelui la nivel cerebral.

În colaborare cu Prof. Dr. Adrian Postelnicu de la Catedra de Termotehnică şi Mecanica Fluidelor, Universitatea Transilvania din Braşov, a fost implementat un model matematic care face trecerea de la sistemul real la ecuaţii matematice, acest model stă la baza produsului soft realizat.

Modul de calcul elaborat este derivat din diverse metode abordate de alţi autori care pe parcursul timpului au studiat şi ei hemodinamica Poligonului Willis şi au dezvoltat, la rândul lor, modele de rezolvare. Diferenţa între modelele vechi şi cel folosit aici o reprezină însă modul inedit în care acesta din urmă abordează subiectul, deoarece nu se orientează spre o rezolvare simbolică a problemei, ci mai degrabă spre una numerică. Datele astfel obţinute fiind extrem de utile şi de sugestive atât pentru medici, care pot fi ajutaţi în alegerea celei mai bune proceduri din punct de vedere al afectării circulaţiei sanguine pentru un anumit pacient, cât şi pentru cercetătorii care pot realiza comparaţii şi studii în domeniu.

Modelul matematic propus presupune implementarea unor ecuaţii pentru toate cele 16 segmente, aceste 16 segmente sunt arterele poligonului, din care este format poligonul. Pentru fiecare nod al poligonului se aplică legile lui Kirckhhoff (Gustav Robert Kirchhoff - n. 12 martie 1824, Königsberg, azi Kaliningrad - d. 17 octombrie 1887, Berlin - a fost un fizician german care a descoperit legile care îi poarta numele în domeniul circuitelor electrice legate de curentul, tensiunea şi rezistenţa electrică.), pentru fiecare arteră se aplică legea Hagen-Poiseuille (În dinamica fluidelor, legea Hagen–Poiseuille descrie curgeri cu viteze mici şi cu caracter vâscos printr-o secţiune circulară constantă.) şi, de asemenea, sunt introduse în ecuaţii constante precum presiunea venoasă, presiunea arterială medie şi vâscozitatea dinamică a sângelui. În urma rezolvării acestui sistem de ecuaţii se obţin presiunile din noduri şi debitele din arterele poligonului. Despre aceste date şi despre semnificaţia lor se va aprofunda în secţiunea destinată prezentării modelului matematic.
Datorită importanţei anatomice a Poligonului Willis şi a variaţiilor (Prin variaţii se înţelege lipsa uneia sau mai multor artere din structura poligonului, cazuri întâlnite des în realitate.) pe care le poate avea această structură am optat pentru realizarea unui produs soft care să implementeze modelul matematic descris mai sus şi care să ia în calcul majoritatea situaţiilor legate de funcţionalitatea poligonului întâlnite în practică.
Aplicaţia a fost realizată sub platforma .NET în limbajul Visual C#. Ca extindere au fost folosite librăria Mapack pentru facilitatea lucrului cu matrici, librăria itextsharp pentru exportarea documentelor în format .pdf şi Microsoft Chart Controls utilizat în realizarea graficelor.
Importanţa ("The Smithsonian / NASA Astrophysics Data System" este un centru NASA care are lucrări pe această temă precum: "Computational Hemodynamics Study of the Cerebral Arterial Circle of Willis" şi "Finite element modeling of the Circle of Willis from magnetic resonance data") deosebită a Poligonului lui Willis, precum şi numărul restrâns (În România există o singură echipă (cea cu care am colaborat) care realizează studii în domeniul hemodinamici Poligonului Willis.) de persoane care efectuează cercetări în acest domeniu demonstrează actualitatea şi originalitatea temei.
O altă dovadă a actualităţii şi importanţei subiectului abordat o reprezintă faptul că numeroase institute de cercetare pun mare accent pe fenomenele care se petrec la nivelul acestui poligon al lui Willis. Spre exemplu pe site-ul (http://www.nas.nasa.gov/Resources/Applications/applications.html) National Aeronautics and Space Administration (NASA) la secţiunea aplicaţii este prezentat un soft numit INS3D care realizează o modelare 3D a Poligonului Willis ce simulează efectul gravitaţiei asupra circulaţiei sângelui în această structură şi prezintă efectele deplasării în spaţiu pe lungi perioade de timp.
Contribuţia personală la acest proiect o reprezintă implementarea aplicaţiei numite **Brain Flow Simulation (BFS)** care simulează funcţionarea poligonului atât în condiţii normale cât şi în condiţii extreme. Realizarea acestui produs soft permite utilizatorului să simuleze curgerea sângelui prin interiorul poligonului, acţiune care implică introducerea datelor iniţiale sau importarea acestora din fişier şi simularea obstrucţionării diferitelor artere.
De asemenea, utilizatorul poate să modifice în partea de setări valorile maxime şi minime ale celor 16 lungimi şi diametre din aplicaţie şi să exporte rezultatele obţinute în urma simulării într-un fişier specific softului. Mai există posibilitatea de vizualizare a graficelor care, pe baza presiunilor din nodurile poligonului, prezintă evoluţia diferitelor debite şi rezistenţe ale arterelor. Valorile acestor presiuni sunt obţinute în urma simulărilor efectuate.
Tot în cadrul aplicaţiei se mai pot defini şi modifica valori critice ale presiunilor şi debitelor şi nota observaţii realizate pe baza fiecărei simulări. Programul oferă pe lângă facilităţile enumerate mai sus şi un help care are rolul de a familiariza utilizatorul cu softul.
Lucrarea de faţă va fi strucuturată în trei părţi. Prima parte sa va axa pe evidenţierea importanţei structurii Poligonului lui Willis, a doua parte se va baza pe descrierea succintă a modelului matematic abordat, iar ultima parte va prezenta produsul software dezvoltat.

2 Importanţa Poligonului Willis

2.1 Circulaţia cerebrală

Aproximativ 15% din cantitatea de sânge pompată de inimă pe minut ajunge la creier. Acestă alimentare cu sânge a creierului este vitală. Întreruperea ei chiar şi pentru câteva secunde duce la pierderea cunoştinţei, iar întreruperi de durată mai mare cauzează traume ireversibile.
În general circulaţia cerebrală rămâne relativ constantă, totuşi uneori apar schimbări la nivel local ca răspuns la schimbările activităţii neuronale. Spre exemplu, o lumină strălucitore îndreptată spre zona ochiilor are ca efect o circulaţie a sângelui mai ridicată în regiunea creierului responsabilă cu vederea.
În poziţie verticală a corpului, ca urmare a efectului gravitaţiei, presiunea sângelui în creier are o valoare mai mică decăt oriunde altundeva în corp, asta face creierul predispus la presiune sanguină scăzută. Cu toate astea creierul nu prezintă fenomenul de "autoreglare", astfel că şi circulaţia cerebrală este stabilă pe o mare gama de presiuni ale sângelui.
Chiar dacă această circulaţie cerebrală este influenţată într-o oarecare măsură de controlul nervos al diametrului vaselor de sânge, este determinată cu precădere de factori chimici, în mod particular de nivelul de dioxid de carbon care, atunci când creşte, are ca efect dilatarea vaselor cerebrale de sânge şi o circulaţie mărită la nivelul creierului.
Opusă circulaţiei ridicate este scăderea circulaţiei cerebrale cauzată de hiperventilaţie care se manifestă prin ventilaţie pulmonară excesivă ce micşorează nivelul de dioxid de carbon din corp şi poate cauza numeroase boli.
În lucrarea "Circulaţia şi metabolismul cerebral" scrisă de Ana Maria Zăgrean (UMF Carol Davila Bucureşti), autoarea afirmă că, în condiţiile în care fluxul sanguin în interiorul creierului este de 750-1000 ml/min, aportul sanguin cerebral se face prin intermediul:

- arterelor carotidiene
- sistemului vertebrobaziliar.

Cele două mari perechi de vase originare din arterele carotide sunt:

- arterele cerebrale anterioare
- arterele cerebrale medii, care transportă aproximativ 80% din sângele care ajunge la emisferele cerebrale.

Arterele cerebrale anterioare aprovizionează cu sânge lobii frontali, parţi ale creierului care se ocupă cu gândirea logică, personalitatea şi mişcarea voluntară, în special picioarele.
Arterele cerebrale medii transportă sânge unei porţiuni a lobilor frontali şi suprafeţei laterale a lobilor temporal şi parietal, incluzând zone senzoriale ale feţei, gâtului, mâinilor şi braţelor, iar în emisfera dominantă, zonei responsabile cu vorbirea. Arterele cerebrale medii sunt cele mai adesea afectate de blocaje ale sângelui. Ruta posterioară ale acestor artere deserveşte lobul occipital şi diencefalul.
În perioada de formare a arterelor cerebrale, arterele comunicante posterioare şi cerebrale posterioare au acelaşi calibru. În luna a patra de viaţă a fătului între artere apare o deodebire de calibru, cel mai adesea diferenţa apărută este în beneficiul arterei cerebrale posterioare. Această neasemănare continuă de obicei şi în perioada copilăriei.
Există două circuite care alimentează creierului cu sânge: circuitul anterior reprezentat de sistemul carotidian care este responsabil de circulaţia anterioară a creierului şi circuitul posterior reprezentat de artera baziliară care asigură circulaţia în porţiunea posterioară a creierului.
Cele două circuite, cel anterior şi cel posterior, nu sunt total independente, ci sunt interconectate prin intermediul arterelor comunicante, care formează Cercul lui Willis la baza creierului, astfel sunt furnizate căi de ocolire potenţiale între circulaţia laterală şi cea antero-posterioară.

2.2 Poligonul lui Willis

Cercul lui Willis sau Poligonul lui Willis reprezintă una dintre cele mai importante zone din creier şi una dintre cele mai eficient concepute sisteme din corpul uman.
Chiar dacă acest poligon al lui Willis nu se ocupă cu acţiuni precum gândirea sau reacţia este indispensabil tuturor celorlalte părţi ale creierului deoarece întreaga cantitate de sânge care ajunge la nivel cranian este eficient asigurată de arterele poligonului.

A fost demonstrat medical că, atât timp cât Cercul lui Willis poate menţine presiunea sângelui la 50% din valoarea normală, nu se produce un infarct sau moartea ţesutului unde are loc blocajul. Dacă circulaţia din zonele vecine cele afectate este bună, un blocaj nu are urmări grave sau permanete, şi asta numai datorită capacităţii poligonului de a regla presiunea sângelui în interiorul creierului.
În concluzie această structură anatomică unică este importantă

- pe de-o parte, datorită rolului său unic de reglare şi distribuţie sanguină la nivelul creierului
- apoi, la nivel clinic, datorită riscului de stroke - un "stroke" se produce atunci când circulaţia cerebrală este temporar întreruptă - care se poate produce în arterele acestei structuri.

Termenul de *stroke* este unul recent introdus în literatura medicală românească pentru a desemna varietatea de fenotipuri şi subfenotipuri reunite sub denumirea de **accident vascular cerebral**, boală cerebrovasculară, infarct cerebral sau hemoragie carebrală.
Prof. dr. D. Ştefănescu afirmă că 4 milioane de locuitori ai planetei mor anual din cauza *stroke*-ului. Cea mai frecventă formă de *stroke* (peste 85%) este cea ischemică. Stroke-ul ischemic rezultată în urma ocluziei unei artere printr-un cheag de sânge fie produs la nivel local cunoscut sub numele de **trombus**, fie mobilizat dintr-un alt teritoriu şi numit **embolus**.
Conform unui studiu realizat de "The University Hospital" din New Jersey-SUA, cercetare efectuată în colaborare cu alte instituţii - American Cancer Society, National Institute of Neurological Disorders and Stroke (NINDS), National Stroke Association, American Diabetes Association, American Society of Interventional and Therapeutic Neuroradiology, American Stroke Association, Brain Attack Coalition, Centers for Disease Control, Hazel K. Goddess Fund for Stroke Research in Women interesate de această temă, pe glob *stroke*-ul este a doua cauză de mortalitate. Din datele obţinute în urma cercetării reiese că: 10% din persoanele care au suferit un *stroke* se refac aproape complet, 25% se refac cu unele urme minore, pentru 50% din victime este necesară internare de urgenţă şi apar urmări grave, iar 15% mor la scurt timp după apariţia unui *stroke*.

2.3 Variaţiile Poligonului Willis

În urma unui studiu realizat de Alpers - Anatomical studies of the circle of Willis in normal brain, aproximativ 50% din populaţia globului are un cerc (poligon) Willis complet. În urma unei alte activităţi de cercetare efectuată aproape trei decenii mai târziu de către Lippert şi Fabst(1985), s-a stabilit existenţa unui acelaşi procent de creiere cu un PW complet, deci tot cca 50% cu un cerc arterial incomplet, situaţie care se mainfestă prin absenţă sau hipoplastie a cel puţin unei artere.
Prin *hipoplastie* a unei artere se înţelege că aceea arteră este foarte mică în calibru sau incomplet dezvoltată. În condiţii normale indivizii care se află în această situaţie nu suferă de probleme de sănătate din această cauză, dar totuşi, ei prezintă un risc mult mai ridicat de apariţie a unui accident cerebral. Acest factor de risc ridicat apare datorită greutăţii de reglare a presiunii sângelui în interiorul poligonului incomplet.
Omul de ştiinţă francez G. Lazorthes şi colaboratorii săi au avut o importantă contribuţie în cercetarea anatomiei funcţionale a PW şi a variantelor sale prin publicarea unei serii de lucrări Lazorthes G., Gouaze A., Santini JJ, Laffont J., *Le modelage du polygone de Willis*, Neurochirurgie, 1974.;Lazorthes G., Gouaze A.,*Conception fonctionnelle du Polligon de Willis*,C.R. Ass. Anat., 1970 care dezbat această temă. Interesul principal al lor a constat în studiul hemodinamicii poligonului şi studiul rolului acestuia în compensarea căilor arteriale de aport.
În funcţie de subiecţii studiaţi şi numărul de creiere pe care se efecutează studiul se pot obţine diferite rezultate pe baza cercetării variaţiilor poligonului. În urma unei analize - Bergman RA, Afifi AK, Miyauchi R, Circle of Willis. Illustrated Encyclopedia of Human Anatomic Variation - efectuate pe 1413 creiere, anatomia clasică a poligonului Willis a fost întâlnită în numai 34,5% din cazuri.
Pot exista şi alte tipuri de variante ale poligonului cum ar fi dublarea unei artere sau variaţii care ţin de apariţia unei ramuri neobişnuite în componenţa acestuia.

3 Modelarea hemodinamicii Poligonului Willis

3.1 Modelarea matematică a fenomenelor anatomice

Modelarea este procesul prin intermediul căruia un sistem real este reprezentat într-un limbaj simbolic. Dacă acest limbaj simbolic este unul matematic atunci modelarea se numeşte sugestiv *modelare matematică*.
Pe parcursul modelării, atunci când se face trecerea de la sistemul real la modelul fizico-matematic, este foarte important să se ţină cont de datele care se pot pierde sau se pot altera. Calităţi ale modelarii precum completitudinea sau adecvanţa depind în mare masură de scopul realizării modelului, de pregătirea persoanei care îl elaborează şi de nivelul de aşteptare al celui care beneficiază de pe urma lui.
De-a lungul timpului au existat oameni de ştiinţă care au încercat să automatizeze acest proces al modelării matematice. Dintre aceştia pot fi amintiţi Dhurjati şi Radhakrishnan, care în anul 2008 au enunţat 8 paşi necesar de parcurs în procesul de modelare matematică a unui fenomen biologic, în particular anatomic:

- Se definesc scopurile şi aşteptările studiului.
- Se definesc ipotezele de lucru.
- Se definesc variabilele, se selectează entităţile relevante pentru scopurile modelării.
- Se definesc variabilele interconectate (ceea ce dă conectivitatea sistemului).
- Se definesc relaţiile de legătură funcţionale şi ecuaţiile de lucru.
- Identificarea valorilor pentru parametrii constataţi din ecuaţiile modelului, cunatificarea relaţiilor de legătură.
- Rularea simulărilor şi sistematizarea cazurilor, pentru a pune în evidenţă tendinţele, vizualizările.
- Validarea şi confirmarea modelului, din punct de vedere calitativ şi cantitativ.

3.2 Evoluţia modelelor Poligonului Willis

Datele din literatura de specialitate arată că reglarea fluxului sanguin cerebral (FSC) este un mecanism complex care are două funcţii principale:

- menţinerea permanenă a aportului adecvat de oxigen şi glucoză;
- eliminarea eficientă a bioxidului de carbon şi a altor produşi finali ai metabolismului cerebral.

Poligonul Willis este un sistem anastomotic care reduce efectele scăderii fluxului sanguin cerebral în una sau mai multe vase aferente prin redistribuirea fluxului sanguin disponibil. Fluxul sanguin cerebral scade dacă tensiunea arterială TA ajunge la 60 mmHg sau creşte dramatic dacă TA medie este peste 150 mmHg. Limita inferioară a autoreglării este extrem de importantă şi reprezintă măsura capacităţii funcţionale a arterelor cerebrale.
Modelele Poligonului Willis descrise în literatură au evoluat de la forme simple la metode complexe care necesită rezolvarea pe calculator (motiv pentru care acestea din urmă sunt numite uneori şi modele computerizate).
Hemodinamica Poligonului Willis a fost studiată de mai mulţi autori prin diverse metode şi modele, fie de tip fizic, fie matematic şi numeric. Câţiva dintre cercetătorii care au formulat modele importante de-a lungul timpului sunt Rogers(1946); Avman şi Bering(1961); Murray(1964); Clark şi colab.(1965); Clark şi Kufahl(1978); Himwich şi colab.(1965); Chao şi Hwang(1971),(1972); Duros şi Navornik(1977); Hillen(1986),(1988); Pascu(1988),(1989); Ursino(2000),(2001),(2003).
Problema principală adusă în discuţie este cea a felului în care aceste modele ajută la cunoaşterea circulaţiei cerebrale. Din punct de vedere medical prin asta se înţelege determinarea modului în care metabolismul creierului depinde de cantitatea şi calitatea sângelui care ajunge în creier. De aceea, scopul modelelor este de a oferi predicţii corecte asupra fluxurilor sanguine pentru stabilirea necesarului perfuziei cerebrale în situaţii normale şi patologice.
Rezultatele obţinute de autori care au studiat modele animal arată modificările în flux cu ocluzii seriate a arterelor carotide sub presiuni normale şi crescute. La animal fluxul mediu bazilar cauzat de ocluzia unei carotide comune a fost de 18 ml/min, în timp ce ocluzia ambelor artere carotide rezultă într-un flux de 35 ml/min. Ocluzia ambelor carotide la câine şi cu o presiune crescută bazilară poate asigura 35.1 ml până la 37.5 ml necesari creierului.

Posibilitatea unei determinări corecte a autoreglării FSC prin model computerizat al PW şi similitudinea rezultatelor obţinute "in vitro" la oameni şi la animale pe care s-au realizat experimente, atestă valabilitatea modelului şi permite simularea unor situaţii variate în circulaţia Poligonului Willis.
Lipsa unanimităţii în opiniile acestor autori duce la concluzia generală că, datorită variabilităţii sale morfologice, nu se poate face nici o predicţie în privinţa funcţionării acestui poligon după ocluzia unuia din vasele aferente.

3.3 Hemodinamica Poligonului Willis în regim staţionar

În acesta secţiune este descris modelul matematic original formulat care simulează funcţionarea unei reţele constituite din mai multe vase sanguine, în acest caz Poligonul lui Willis, şi felul în care acesta a fost abordat în program (ecuaţiile care au fost implementate). În continuare este prezentat modului de calculare a debitelor şi a presiunilor în Poligonul Willis ideal (adică PW cu o morfologie simetrică) şi sunt descrise şi câteva cazuri morfologice particulare.
Funcţioarea reţelei se face prin presiuni şi fluxuri. Astfel, arterele carotide interne şi artera baziliară (ca artere aferente) comunică prin arterele comunicante anterioară şi posterioară prin care se poate realiza schimbul necesar redistribuirii sângelui la nivelul creierului. Acest lucru devine efectiv în cazul unei obstrucţii, a unui vas aferent. În acest mod se asigură o presiune de perfuzie adecvată în creier prin redistribuirea fluxului cerebral.
În simulăriile care vor fi realizate cu ajutorul softului va fi demonstrat că, în funcţie de fluxul din arterele aferente (carotida internă şi baziliară), fluxul în artera comunicantă posterioară poate fi negativ sau pozitiv, fiind cunoscut în literatură rolul critic al acestei artere.
Modelul matematic al hemodinamicii poligonului Willis a fost elaborat în două etape. Prima etapă a constat în scrierea şi validarea tuturor ecuaţiilor, iar a doua în aplicarea acestor ecuaţii pe unele cazuri concrete şi studierea rezultatelor obţinute.
Primul pas a fost modelarea hemodinamicii ca o curgere laminară a unui fluid incompresibil Newtonian (sângele) într-o reţea complexă de tuburi (conducte). În plus, s-au considerat condiţii de curgere de tip staţionar, ceea ce constituie un pas preliminar de abordare, în concordanţă, de altfel, cu metodologia întâlnită în literatura de specialitate. În aceste condiţii, s-a trecut la scrierea ecuaţiilor modelului, care constau în ecuaţiile de continuitate în nodurile modelului geometric şi a celor de echilibru dinamic în curgerea laminară pe fiecare din arterele reţelei hidrodinamice. Necunoscutele sunt debitele şi presiunile, iar numărul de necunoscute este riguros egal cu cel al ecuaţiilor, cu alte cuvinte problema este perfect determinată din punct de vedere algebric.
Conform legii I a lui Kirchhoff, despre care am mai vorbit, făcând analogia între intensitatea curentului electric şi debitul volumeric de sânge, suma debitelor care intra într-un punct (nod) este egala cu suma debitelor care ies din acel nod. Conform schemei reprezentate în figura 1, se poate scrie această lege în fiecare din cele 7 noduri ale poligonului, notate de la A. până la G.
Poligonul este alcătuit din 16 artere, acestea sunt: Artera carotidă internă stânga

- Artera cerebrală medie stânga
- Artera comunicantă posterioară
- Artera cerebrală posterioară I stg. (P1 stânga)
- Artera cerebrală posterioară II stânga (P2 stânga)
- Artera cerebrală anterioară II stânga (A2 stânga)
- Artera comunicantă anterioară
- Artera cerebrală anterioară I dreaptă (A1 dreaptă)
- Artera cerebrală anterioară II dreaptă (A2 dreaptă)
- Artera carotidă internă dreaptă
- Artera cerebrală medie dreaptă
- Artera comunicantă posterioară dreaptă
- Artera cerebrală posterioară I dreaptă (P1 dreaptă)
- Artera cerebrală posterioară II dreaptă (P2 dreaptă)
- Artera bazilară.

În toate nodurile Poligonului Willis se aplică ecuaţiile de continuitate, ecuaţiile sunt următoarele:

$$Q_{16} = Q_4 + Q_{14} \tag{1}$$

$$Q_5 = Q_3 + Q_4 \tag{2}$$

$$Q_1 = Q_2 + Q_3 + Q_6 \tag{3}$$

$$Q_6 = Q_7 + Q_8 \tag{4}$$

$$Q_9 = Q_8 + Q_{10} \tag{5}$$

$$Q_{11} = Q_{10} + Q_{12} + Q_{13} \tag{6}$$

$$Q_{15} = Q_{13} + Q_{14} \tag{7}$$

Pe fiecare din cele 16 segmente, se aplică legea Hagen-Poiseuille:

$$(\Delta p)_i = \frac{8\mu l_i}{\pi r_i^4} Q_1 = R_i Q_i \tag{8}$$

unde R_i este rezistenţa hidrodinamică (în curgerea laminară).
Acum, aplicăm legea (3.8) pentru toate segmentele modelului:

$$p_m - p_A = R_{16} Q_{16} \tag{9}$$

$$p_B - p_v = R_5 Q_5 \tag{10}$$

$$p_m - p_C = R_1 Q_1 \tag{11}$$

$$p_C - p_v = R_2 Q_2 \tag{12}$$

$$p_D - p_v = R_7 Q_7 \tag{13}$$

$$p_E - p_v = R_9 Q_9 \tag{14}$$

$$p_m - p_F = R_{11} Q_{11} \tag{15}$$

$$p_G - p_v = R_{15} Q_{15} \tag{16}$$

$$p_A - p_B = R_4 Q_4 \tag{17}$$

$$p_C - p_B = R_3 Q_3 \tag{18}$$

$$p_C - p_D = R_6 Q_6 \tag{19}$$

$$p_D - p_E = R_8 Q_8 \tag{20}$$

$$p_F - p_E = R_{10}Q_{10} \tag{21}$$

$$p_F - p_E = R_{13}Q_{13} \tag{22}$$

$$p_A - p_G = R_{14}Q_{14} \tag{23}$$

unde pm este presiunea arterială medie şi pv presiunea venoasă la limitele reţelei.

Strategia de rezolvare a ecuaţiilor este de a elimina cele 7 presiuni din nodurile poligonului, pA , pB, pC, pD, pE, pF, pG, între ecuaţiile (3.9)-(3.32) pentru a obţine un sistem algebric liniar în necunoscutele Q1, ..., Q16.

Sistemul de ecuaţii rezultat din (3.1)-(3.8), (3.9)-(3.32) a fost rezolvat numeric cu ajutorul softului realizat. Datele de intrare necesare în rezolvarea modelului sunt:

- Rezistenţele vasculare, conform Hillen şi colab. (1988).
- Diametrele şi lungimile diferitelor artere aferente, eferente şi aparţinând poligonului Willis. În aceste cazuri, rezistenţele vasculare au fost calculate utilizând o valoarea Poise a vâscozităţii dinamice a sângelui egală cu 0.04, considerat ca fluid Newtonian, ca şi în exemplul lui Pascu (1988).
- Presiunea arterială medie şi cea venoasă este conform Hillen şi colab. (1988).

Astfel se obţin valori pentru debit şi presiuni care pot fi studiate mai departe de cercetători.

3.4 Discuţii şi concluzii

S-a elaborat un model matematic liniar unidimensional al Poligonului Willis, care a fost rezolvat numeric. Etapele de lucru au fost:

- validarea modelului,
- modelul cu artera ocluzionată în care au fost prezentate 3 cazuri particulare - ocluzia de arteră carotidă stângă, de arteră bazilară şi respectiv arteră carotidă stangă şi arteră bazilară;
- blocare de arteră intra-Poligon Willis - arteră cerebrală posterioară, arteră comunicantă posterioară şi respectiv arteră cerebrală anterioară;
- patru modele cu variaţii ale lungimii şi respectiv ale diametrului.

Ca urmare a tuturor rezultatelor şi concluziilor prezentate în acestă secţiune se poate deduce că:

- Modelul elaborat dă predicţii corecte în privinţa modificărilor debitului şi presiunilor în cazul Polgonului Willis "ideal" simetric.
- Cazurile particulare de ocluzie la nivelul ACI stângă, arteră bazilară şi respectiv arteră carotidă stângă şi arteră bazilară au dat predicţii cu aplicabilitate practică.
- Artera comunicantă în cazul Poligonului Willis ideal simetric nu este funcţională (debit zero) ceea ce în realitate nu se poate întâmpla, ducând la ideea că Poligonul Willis "ideal" se poate referi la morfologia, nu şi la funcţionalitatea sa.
- Poligonul Willis pare a avea un rol potenţial funcţional numai în situaţii particulare.

4 Softul - BFS (Brain Flow Simulation)

4.1 Prezentarea generală a softului

Aplicaţia Brain Flow Simulation (În traducere liberă: Simularea Curgerii Sanguine la Nivelui Creierului) după cum ne spune şi numele, are ca scop realizarea în mod automat de simulării ale funcţionării Poligonului Willis. Softul a fost creat sub platforma Microsoft .NET, în limbajul de programare Microsoft Visual C# 2008. Justificarea alegerii acestui limbaj ţine de modernitatea şi simplitatea lui şi în plus de suportul semnificativ pe care îl oferă utilizatorului în realizarea aplicaţiilor bazate pe ferestre.

BFS se adresează în special comunităţii de medici şi de cercetători care efectuează studii în domeniul curgerii sanguine cerebrale şi echilibrarea presiunilor la nivelul creierului. Având în vedere scopul ei, aplicaţia oferă rezultate pentru numeroase cazuri întâlnite în practică şi, de asemenea, pot fi efectuate teste

pentru cazuri nedescoperite, dar cu potenţial de apariţie. Se poate concluziona că numărul simulărilor care se pot efectua prin intermediul softului este practic infinit, asta datorită multitudinii şi variaţiei de combinaţii a datelor de intrare care pot fi introduse.

Pe lângă suportul ştiinţific care stă în spatele aplicaţiei, se remarcă uşurinţa folosirii acesteia şi extindrea ei pe o arie mult mai largă decât obţinerea de rezultate în urma rezolvării unor ecuaţii. Cu toate că aceste date obţinute la finalul simulării virtuale sunt partea semnificativă a programului, softul încearcă să pună accent şi pe confortul utilizatorului.

Acesta poate naviga în cadrul unui meniu care îi oferă posibilitatea să efectueze simulări, să modifice setările programului, să definească valori între care rezultatele obţinute se pot încadra, să vizualizeze grafic evoluţia datelor în urma simulărilor, să noteze observaţii cu privire la datele pe care programul le-a oferit, toate acestea încadrate într-o interfaţa prietenoasă şi uşor de folosit. Totuşi, dacă utilizatorul întâmpină probleme pe parcursul utilizării softului poate apela la meniul de ajutor oferit de aplicaţie.

Pentru uşurinţa introducerii datelor în interiorul aplicaţiei acestea se pot importa din fişiere de tip .txt şi de asemenea rezultatele obţinute se exportă într-un fişier .wls (fişier cu extensie proprie aplicaţiei, interpretat de sistemul de operare ca fişier .txt). Tot cu rol de a facilita introducerea de valori, în părţile aplicaţiei unde s-a considerat necesar a fost inserată o imagine a Poligonului lui Willis pe care se marchează artera curentă pentru care se adaugă aceste valori.

La partea de simulare a curgerii sângelui prin Poligonul lui Willis, aplicaţia foloseşte modelul descris în capitolul anterior. Datele de intrare sunt reprezentate de lungimile şi diametrele celor 16 artere ale poligonului, iar datele de ieşire din fişierul .wls sunt debitele sângelui din artere şi presiunile în cele 7 noduri formate de aceste artere.

Rezultatele obţinute ca urmare a rulării unei simulări în cazurile de referinţă sunt şi ele conform tabelelor prezentate în capitolul precedent. Validarea valorilor din aceste tabele a fost făcută de colectivul implicat în realizarea modelului matematic al poligonului şi verificată de echipa de medici cu care s-a lucrat pentru obţinerea datelor de intrare. Ca urmare a tuturor acestor testări ale acurateţii rezultatelor se consideră valide toate datele obţinute de softul BFS.

4.2 Ghid de utilizare

Aplicatia este alcatuită din şapte taburi:

- Pornirea aplicaţiei - Tabul Home

O dată instalată aplicaţia şi imediat după prima rulare utilizatorul e întâmpinat de o fereastră cu conţinut introductiv unde este afişat numele aplicaţiei.
În partea de sus a ferestrei care s-a deschis se pot observa taburile prin care poate naviga utilizatorul. Numele fiecărui tab este însoţit de o iconiţă sugestivă a conţinutului acestuia.

- Tabul Simulare

Partea de meniu a tabului aflată în zona din stânga este alcătuită din arterele poligonului pentru care se pot introduce date. Fiecare arteră este reprezentată în cadrul meniului printr-o prescurtare a denumirii acesteia şi numărul ei acordat convenţional.
Pentru fiecare arteră există două zone unde se pot introduce date, zona lungimilor, respectiv zona diametrelor. Este declarată o valoare maximă şi una minimă între care poate lua valori o lungime şi un diametru.
Apăsarea butonului **Calculează** are ca efect execuţia simulării, adică pe baza datelor din căsuţele aflate în dreptul arterelor se calculează valori pentru debite şi presiuni (debitele în artere şi presiunile în noduri). Acest calcul se face aproape instant şi imediat rezultatele sunt exportate într-un fişier .wls creat în directorul curent, fişier care este şi deschis pentru vizualizare.

- Tabul Grafice Se remarcă două zone principale, zona de afişare a graficelor şi zona de butoane.

Partea de afişare este alcătuită la rândul ei din alte 7 taburi aliniate în partea de jos a imaginii. În cele 7 taburi apare prezentat succesiv câte un grafic pentru o presiune dintr-un nod al poligonului.
Fiecare presiune din noduri este caracterizată, pe lângă valorile constante de care depinde (presiunea venoasă, respectiv presiunea arterială medie), de două valori variabile: debitul dintr-o anumită arteră şi rezistenţa arterială respectivă. Se poate observa că aceste două valori au fost reprezentate pe grafic.

- Tabul Setări Acesta este locul unde sunt efectuate setările pentru realizarea simulărilor. Controlul

(vezi figura 7) conţine o zonă numită **Modificări**, zonă care are aceeaşi structură cu partea de **Meniu** din tabul **Simulare**. În acest loc pot fi redefinite valorile minime şi maxime pentru lungimi şi diametre. Căsuţele din **Modificări** conţin implicit capetele actuale de interval şi pot fi modificate fie de la tastatură fie prin apăsarea săgeţilor din dreapta lor. Aceleaşi reguli în ceea ce priveşte introducerea datelor ca şi la **Simulare** se aplică şi aici.

- Tabul Valori Critice

Pentru fiecare căsuţă este definită o valoare maximă şi una minimă şi de asemenea un pas de incrementare. Deoarece în această secţiune datele nu sunt introduse atât de des nu s-a considerat necesar redefinirea acestui pas sau a valorilor care alcătuiesc capetele de intervale.

- Tabul Ajutor

Acest tab oferă utilizatorului informaţii cu privire la modul în care softul poate fi folosit. Tot aici este locul unde se explică funcţionalitatea fiecărei părţi a aplicaţiei.

- Tabul Despre

Tabul **Despre** a fost implementat şi integrat în cadrul aplicaţiei cu scopul de a enumera persoanele responsabile cu realizarea acestui soft.

- Tabul Notiţe

Utilizatorul poate edita zona din centrul paginii, iar apoi, pentru salvare se apasă butonul **Adaugă notiţă** sau **Salvează pagină** din meniu. Dacă nu a fost inserat text nou butonul **Adaugă notiţă** nu este activ. La fel şi butonele **Pagină Înainte** şi **Pagină Înapoi** sunt inactive dacă nu există pagini deja scrise. Acţiunea de inserare separator se poate face şi acţionând butonul cu acelaşi nume din dreapta tabului.

4.3 Realizarea softului

4.3.1 Baza aplicaţiei

La fel cum am spus şi în introducerea acestui capitol softul a fost realizat în Visulal C# 2008, sub platforma .NET. Aplicaţia este alcătuită dintr-o formă care conţine un control de tip **Tab Control** unde sunt dispuse pe rând, în funcţie de **Tab Page**-ul selectat, paginile aplicaţiei.

Pentru realizarea aplicaţiei BFS au fost folosite şi două biblioteci, una pentru a facilita modul de lucru cu matrici şi alta pentru a ajuta la exportarea datelor în format .pdf. Biblioteca Mapack este cea care oferă metode deja implementate pentru realizarea operaţiilor cu matrici utile în rezolvarea ecuaţiilor prezentate la partea de modelare. Cealaltă bibliotecă folosită este itextsharp care ajută la exportarea automată de text din interiorul aplicaţiei în format .pdf mult mai elegant pentru utilizator.

Alte facilităţi folosite sunt şi Microsoft Chart Controls, Add-on pus la dispoziţie de Microsoft pentru lucrul cu grafice şi CHM Maker unde au fost înglobate toate paginile web care alcătuiesc help-ul în format .chm.

Ultimul lucru demn de menţionat înainte de a trece la prezentarea părţii de modelare, este că pozele cu schiţa poligonului au fost realizate manual în **Paint**.

4.3.2 Integrarea modelului matematic în program

Lansarea în execuţie a unei simulări se face prin click pe butonul **Calculează** din tabul **Simulare** al controlului. În urma apăsării acestui click se execută evenimentul *OnClick* implementat butonului **Calculează**.

Primul pas îl constituie stabilirea existenţei arterelor ocluzionate. Această verificare se face prin analizarea tuturor **checkBox**-urilor din tabul de Simulare. În funcţie de valoarea de adevăr a proprietăţii **NumecheckBox.Checked** este apelată procedura corespunzătoare cazului de ocluzionare sau procedura specifică neocluzionării nici unei artere.

Procedurile cu nume de tip Calculează+*cifra arterelor ocluzionate* au în esenţă aceeaşi structură. Diferenţa dintre ele constă în modul de formare al matricii de calcul şi în linia respectiv coloana care se elimină din aceasta. După pornirea execuţiei aceastei proceduri de calculare se apelează şi **Exportă_date**, o metodă care are rol de scriere a rezultatelor în fişier.

Procedura Calculează

Procedura are doi parametri, vectorul lungimilor şi vectorul diametrelor, două structuri de tip *double*, care au fost create în evenimentul butonului **Calculează**. Într-un ciclu de tip **for** sunt iniţializate valorile lungimilor şi diametrelor cu datele din controalele **numericUpDown** conţinute de *Form*, împarţite la 1000, controale ale căror valori au fost setate de utilizator. Prin intermediul comenzii *(double)numeNumericUpDown.Value* este obţinută în aplicaţie valoarea, care este apoi transformată în tipul *double* a controlului.

În acelaşi ciclu **for** este implementat şi vectorul rezistenţelor vasculare ce primeşte valoarea **8*miu*lungimi[i]/ (pi * Math.Pow((0.5 * diametre[i]), 4))**, formula extrasă din modelul matematic. **miu** este vâscozitatea sângelui pentru 50% hematocrit, valoare considerată constantă egală cu 0.004, pi este constanta matematică π, iar *Math.Pow* este o funcţie care returnează primul parametru la puterea celui de-al doilea.

Sunt calculate apoi valorile pentru *pm* şi *pv* (presiunea arterială medie şi presiunea venoasă la limitele reţelei) care sunt egale cu *0.092 * 13596 * 9.81* şi *0.010 * 13596 * 9.81* şi din aceste două valori se obţine *deltap* prin scaderea lui pv din pm. Valoarea *deltap* se foloseşte în continare la calculul debitelor.

Pasul următor constă în crearea unei noi matrici **a** alcătuită din 16 linii şi 16 coloane, care primeşte ca valori un şir de 1, -1, respectiv datele conţinute în rezistenţele vasculare calculate. Felul în care aceste valori trec în matrice depinde de tipul de ocluzionare care a fost sau nu efecutat. Imediat după implementarea acestei matrici se mai creează o matrice **b** cu un rând şi 16 coloane care conţine valori de 0 şi 1, dispunerea lor depinzând tot de modul de ocluzionare.

Cu ajutorul acestor două matrici nou create se calculează debitele în cele 16 artere ale poligonului. O nouă matrice Qb este iniţializată, ea primeşte ca valoare **Matrix.Multiply(a.Inverse, b.Transpose())**. *Matrix.Multiply*, *Inverse* şi *Transpose* sunt metode puse la dispoziţie de biblioteca Mapack, în ordine, acestea fac înmulţire, inversare respectiv transpunere de matrici. Vectorul **Q** al debitelor va primi valoarea acestei matrici *Qb* obţinută înmulţită cu *deltap* calculat mai sus.

Procedura se încheie cu calculul acestui vector Q, urmând ca în procedura de **Exportă_date** să fie calculate şi presiunile.

Procedura Exportă_date

Această procedură se apeleaza imediat după cea de calculare, în acelaşi eveniment *OnClick* al butonului **Calculează**. Mai întâi se setează numele fişierului în care urmează a fi exportate date prin comenzile:

```
DateTime currentDateTime = DateTime.Now;
String dateStr =
 currentDateTime.ToString("yyyy-MM-dd HH_mm_ss");
TextWriter tw = new StreamWriter(dateStr + ".wls");
```

În prima linie este preluată în variabila *DateTime* data curentă apoi această data este transformată în *string* cu structura an, luna, zi, ora, minut, secudă, urmând ca un fişier să fie creat cu acest nume. Prin această comandă se evită suprascrierea fişierelor, fiecare având astfel nume unic.

Primele linii din fişier conţin valorile lungimilor şi ale diametrelor formatate cu comanda *String.Format* pentru o aşezare în pagină mai eficientă. Pe următoarele sunt afişate debitele obţinute prin înmulţirea valorilor din vectorul *Q* de mai sus cu *1e+6* adică 10 la puterea 6.

Mai departe sunt calculate valorile tuturor presiunilor din noduri prin scăderea sau adunarea la *pm* ori *pv* debitele şi rezistenţele unor anumite artere, la rezultatul obţinut aici se mai înmulţeşte o constantă numită *fac* egală cu *100/(13596*9.81)*.

Datele sunt salvate în fişier, iar acesta este deschis în aplicaţie prin comanda:

```
System.Diagnostics.Process.Start
("notepad.exe", dateStr + ".wls");
```

unde *dateStr* este numele fişierului în care se salvează datele.

5 Rezultate si concluzii

Rezultatele obţinute şi importanţa acestora ţin în primul rând de aplicabilitatea lor în sfera medicală. Pe lângă concluzile care pot fi trase pe baza rezolvării modelului matematic, softul mai pune la dispoziţie câteva metode de analiză a datelor.

După fiecare simulare utilizatorul poate copia în notiţe datele generate de program care ţin de simularea în discuţie. Pe baza simulărilor sunt generate grafice sugestive care ajută la analiza în profunzime a rezultatelor. Pe fiecare grafic se pot observa diferenţe de rezistenţă la un debit aproape constant şi difenreţe de debite la o rezistenţă relativ constantă. Din această constatare se poate trage concluzia că în cadrul poligonului se întâlnesc aceleaşi rezistenţe pe o arteră şi la debite mari, dar şi la debite mici. Rezistenţa deci nu este influenţată de debit. Cu creşterea numărului de simulări realizate creşte şi posibilitatea de cercetare.

Partea de **Setări** a aplicaţiei are rolul de a extinde zona de efectuare a simulărilor dincolo de datele cunoscute şi întâlnite în practică. Este de remarcat fapul că aplicaţia, şi prin intermediul acestui tab, oferă o mult mai mare gamă de valori pe care se pot realiza apoi cercetări medicale.

Datele obţinute în urma rulării simulărilor ajută echipa de medici a Spitalului Judeţean în cercetarea medicală. În plus softul BFS asigură o platformă experimentală virtuală care poate fi folosită pentru training în acest domeniu. Pe viitor, atunci când datele despre lungimile şi diametrele arterelor acestui poligon vor fi preluate prin intermediul rezonanţei magnetice nucleare (Rezonanţa Magnetică Nucleară, RMN, este o tehnică spectroscopică foarte des folosită în chimie, chimie fizică, medicină, biofizică şi inginerie nucleară pentru determinarea structurii diverşilor compuşi chimici, în biochimie pentru determinarea structurii proteinelor fiind singura tehnică destinată determinării structurii proteinelor în soluţie (condiţii mult mai apropiate de cele native) sau în imagini, diagnostice medicale sau radiologice pentru determinarea caracteristicilor fizico-anatomice a unor organe sau ţesuturi.) sau Doppler-sonografia (Doppler-sonografia este o investigaţie noninvazivă care masoară fluxul sangvin în anumite regiuni anatomice. Doppler-sonografia vaselor sanguine – o metodă de analiză cu ultrasunet pentru constatarea tulburărilor în refluxul venos la nivelul gâtului, capului sau membrelor, ca urmare a mai multor cauze.) softul poate asigura indicaţii specifice pentru realizarea intervenţiilor chirurgicale. Investigaţii numerice care se pot realiza prin intermediul aplicaţiei pot ghida chirurgul în alegerea celei mai bune proceduri pentru un anumit pacient, din punct de vedere al afectării circulaţiei sanguine.

Bibliografie

[1] Adrian Postelnicu, *Modelarea numerică a hemodinamicii Poligonului Willis*, Editura Universităţii Transilvania din Braşov, 2008

[2] Falup Pecurariu C., Postelnicu A., Pascu A., *Studiul debitelor şi presiunilor în Poligonul Willis în modelul cu ocluzie aferentă*, Revista Română de Stroke, vol. IX, nr.1, 2006

[3] Falup Pecurariu C., Postelnicu A., *Modelarea computerizată a hemodinamicii Poligonului Willis*, J Med Brasov, 2007

[4] J J Van Overbeeke, B Hillen, and C A Tulleken, *A comparative study of the circle of Willis in fetal and adult life. The configuration of the posterior bifurcation of the posterior communicating artery*, 1991 June, 176: 45–54.

[5] A. Quarteroni, *Modeling the Cardiovascular System – A Mathematical Adventure*, SIAM News 34 (5), 2001 (Partea I) şi SIAM News 34 (6), 2001 (Partea II).

Roman Cristina Elena
Universitatea Transilvania
Braşov
Informatică aplicată
Adresa Str. Iuliu Maniu, nr. 50,
Cod poştal: 500091, Brasov
România
E-mail: *cri_roman@yahoo.com*

Sesiunea Naţională de Comunicări Ştiinţifice a Studenţilor
"Imaginaţie, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Sistem de Management al Informatiei

RUSU Andrei, VOLOSINCU Mihai Bogdan
Coordonator: Prof. univ. dr. SIMIAN Dana

Abstract

The aim of this paper is to present an original data-centric application, that has realized an optimization of information management and distribution in companies or educational institutions. Our application has a desktop component, build in Java and a WEB component, built in PHP. This System of data management provides many advantages compared with other existing free systems like: adaptability, friendly interface, many functions and security.

1 Introducere

Mai mult ca oricând, datorită importanţei pe care a ajuns să o deţină in orice domeniu de activitate, informaţia trebuie prelucrată si distribuită cât mai rapid, corect şi eficient.

În ceea ce priveşte distribuţia informaţiilor în România multe societaţi şi instituţii la acest capitol nu stau bine deoarece nu îşi permit achiziţionarea şi întreţinerea unui sistem care să ofere posibilitatea de a manipula informaţiile pe plan intern, pe partea de client sau public larg.

Lucrarea prezintă un nou sistem de distribuţie a informaţiilor din cadrul instituţiilor si societaţilor comerciale, care elimină o parte din problemele specifice entitaţilor precizate mai sus. Pe piaţă există mai multe aplicaţii destinate distribuţiei informaţiei, dar prezintă unele dezavantaje pe care incercăm să le inlăturăm prin prezenta aplicaţie. Unul dintre principalele dezavantaje il reprezintă costul unui astfel de sistem. Programele existente pe piaţă au o complexitate mare necesitând personal calificat in utilizarea lor ceea ce duce la costuri suplimentare. Alegerea unei aplicaţii existente, care este free, are ca principal dezavantaj faptul că modelul oferit de manipulare si distribuire a informaţiei nu se potriveşte cu modelul dorit.

Există un număr mare de exemple de aplicaţii pentru organizarea si distribuirea informaţiei, accesibile pe internet, care se ocupă de gestionarea informaţiilor dar acestea nu au ca scop eliminarea problemelor recent amintite Acestea sunt create pentru comunităţi web sau aplicaţii desktop pentru utilizare la nivel personal si nu pentru organizaţii.

Pintre acestea precizam :
Database Master (http://www.nucleonsoftware.com/?page=databasemaster)
MySQL Workbench (http://www.mysql.com/products/workbench/)
Aceste două programe necesită cunoştinţe solide in baze de date.
PHP-Fusion (http://www.phpfusion.ro/news.php)
PHPnuke (http://phpnuke.org/modules.php?name=Release)

Obiectivul principal al sistemelor existente este să centralizeze informaţia din cadrul unei societăţi sau instituţii şi nu să şi asigure o distribuţie corectă şi optimă a acesteia.

2 Organizarea Sistemului

Sistemul prezentat este compus dintr-o componentă desktop construită in Java şi o componentă Web construită in PHP care are ca obiect central o bază de date, cele două componente oferind următoarele avantaje:

- Simplitate in privinţa lucrului cu informaţia, deoarece aplicaţia va fi folosită de toţi angajaţii;
- Adaptabilitate la nivelul utilizatorului ;
- Poate fi adaptată la structura ierarhică de organizare a informaţiei proprie instituţiei care o foloseşte
- Deşi se caracterizează prin simplitate de utilizare şi rezolvă problemele curente de distribuire a informaţiei, inglobează şi aspecte mai complexe, oferind suport pentru funcţii mai complexe

Am ales Java si PHP deoarece: sunt free, oferă un suport complex pentru lucrul cu baze de date, există multe materiale destinate documentarii, sunt două limbaje foarte puternice in lucrul cu cantitaţi mari de date .

Aplicaţiile se folosesc de un sistem complex de autorizare a utilizatorilor, care permit sau restricţionează accesul la un set specific de informaţii şi acţiuni pentru a asigura distribuirea corectă a datelor. Un punct foarte important la capilolul securitate este ca la crearea unui utilizator i se atribuie un nivel de securitate.

Sistemul se foloseşte de o diagramă ierarhică a firmei sau instituţiei care ajută sistemul să determine un set iniţial de privilegii in mod special pentru conducerea organizaţiei. Utilizatorul poate modifica ierarhia care o oferă sistemul pentru a reflecta cât mai bine structura societăţii sau instituţiei. Diagrama poate suferi modificări fără a influenţa intr-un fel circulaţia fluxurilor informaţionale.

Sistemul se foloseşte de fluxuri informaţionale pentru a trimite informaţia de la un emiţător la un receptor.Am folosit modelul fluxului informational prezentat in cartea notate cu A.1. in bibliografie.

La cerere un emiţător poate crea un flux informaţional cu următoarele caracteristici de comunicare:

a) Un emiţător la mai mulţi receptori:

Proprietăţi:

- O singură direcţie, spre receptor;
- Se transmite acelaşi mesaj;

Exemplu: o directivă nouă trimisă de la conducere către toţi angajaţii sau doar la un număr select de angajaţi.

b) Mai mulţi emiţători la mai mulţi receptori:

Proprietăţi:

- Fiecare poate fi emiţător si receptor;
- Comunicarea se face in toate direcţiile;

Exemplu: membri unei echipe colaborând pentru ducerea la bun sfârşit a unui proiect.

c) Un emiţător-receptor la un alt emiţător-receptor:

Proprietăţi:

- Comunicarea se face in două sensuri;
- Mesaj personalizat;

Exemplu: schimb de informaţii între doi colegi ai aceluiaşi departament (aceeaşi echipă).

d) Pe segment ţintă:

Proprietăţi:

- Un emiţător si mai mulţi receptori;
- O singură direcţie a fluxului, spre receptor;

- Se transmit mesaje diferite pentru fiecare receptor;

Exemplu: liderul unei echipe deschide un flux cu colegii săi şi împarte responsabilităţile fiecăruia pentru un anume proiect.

Fluxul este caracterizat de următoarele:

- Conţinut;
- Formă;
- Sens;
- Frecvenţă;
- Volum;
- Lungime.

Informaţiile pot fi în mai multe forme, fluxul permiţând transferul de fişiere (imagini, documente, secvenţe audio/video) şi nu doar informaţiile scrise.

Fluxul informaţional poate funcţiona pe toate direcţiile sau sensurile dintr-o diagramă ierarhică:

- Orizontal, intre compartimentele de pe acelaşi nivel;
- Vertical, intre compartimentele de pe nivele diferite;
- Ascendent, de la un compartiment de pe un nivel inferior la un nivel de conducere;
- Descendent, de la un nivel de conducere la un compartiment inferior.

Aplicaţia răspunde unor serii de cerinţe pentru a asigura un flux informaţional mai stabil, sigur şi productiv:

- Serverul Web, serverul bazei de date şi modului php pot fi achiziţionate la un cost foarte redus, toate cele trei componente necesare implementării aplicaţiei sunt gratuite, costurile fiind doar legate de intreţinere pentru serverele proprii ale organizaţiei sau ca plată pentru un plan de web hosting al unei firme de specialitate;

- Sistemul permite logarea fiecărui flux şi modificări a bazei de date pentru a fi mai uşoară şi sigură rezolvarea unei probleme in cazul unei erori umane;

- Aplicaţia poate coda informaţiile pe server pentru mai multă siguranţă;

- Nu are limită pentru cantitatea de informaţii transmise pe un flux informaţional fără a încetini sistemul in cazul prelucrării, stocării sau vizualizării informaţiei (cea de-a 3- a poate fi încetinită, dar doar de conexiunea de internet a utilizatorului);

3 Modulul Desktop

Aplicaţia desktop este construită in limbajul Java, limbaj care pune la dispoziţia programatorilor un set complex de tehnologii pentru lucrul cu date, mai exact pentru această aplicaţie, aceste tehnologii constau in API's pentru lucrul cu bazele de date;

Accesul la aplicaţia desktop se face după următorul model :

1. Utilizatorul, inainte de logare trebuie să se înregistreze cu numele şi parola create de root sau de un alt utilizator cu un nivel de securitate ridicat şi care are dreptul de a crea utilizatori.
2. Înregistrarea este obligatorie deoarece aplicaţia poate fi folosită pentru accesarea bazei de date a unei firme dar şi a unei baze de date personale.
3. Nivelul de securitate al utilizatorului este citit imediat ce aplicaţia a reuşit să se conecteze la baza de date.
4. Opţiuni cum ar fi datele personale, nivelul de securitate, o scurtă istorie a sa în cadrul programului şi posibilitatea de a modifica aceste date, sunt afişate intr – o secţiune specială.
5. Accesul la informaţie se face pe baza unei structuri ierarhizate, construită după un model specific fiecărei firme sau instituţii.

3.1 Baza de date

Acest capitol este foarte important deoarece aici am creat un model mai diferit şi cu elemente de originalitate, în ceea ce priveşte lucrul cu informaţia unei baze de date.

- Afişarea tabelelor, rapoartelor sau a anumitor coloane din tabele este foarte simplă şi foarte rapidă
- Modificarea datelor este foarte simplă necesitând doar schimbarea informaţiei din celula selectată
- Pentru persoanele care au cunoştinţe solide în ceea ce priveşte administrarea bazelor de date această secţiune dispune de un command line ce ofera posibilitatea de a folosi baza de date la capacitate maximă
- Un element original aplicaţiei este dat de posibilitatea de a introduce un volum mare de informaţii în tabele cum ar fi liste sortate sau fişiere
- Opţiunea de căutare a unui cuvânt în conţinutul tabelelor
- În funcţie de nivelul de securitate a utilizatorului acesta are posibilitatea de a crea utilizatori noi, modifica tabele, crea tabele noi sau şterge

3.2 Informaţia în cadrul aplicaţiei

Cea de-a doua secţiune din cadrul aplicaţiei se ocupă de lucrul cu fişiere şi a fost creată pentru ca utilizatorii să-şi poată prelucra informaţia in cadrul aplicaţiei .

1. Astfel utilizatorul are posiblitatea de sortare după diferite criterii. Ex: listele de personal pot fi sortate după nume, prenume, vârsta sau alt criteriu;
2. Creare de secţiuni speciale care se ocupă cu statistici, grafice;
3. Creare tabele si mutarea lor în baza de date;
4. Căutare în fişiere a unui cuvânt cheie;
5. Editor de texte;

Secţiunea de lucru cu fişierele este foarte importantă deoarece oferă posibilitatea de a prelucra informaţia şi în cazul în care utilizatorul se află într-un loc unde nu există o conexiune la internet sau conexiunea s-a întrerupt.

Pentru prelucrarea informaţiei am creat un set de unelte originale, diferite, cu o legătură foarte strânsă intre ele.

4 Modulul Web

Partea web a sistemului a fost facută pentru a ajuta comunicarea între departamente, personal şi conducere, în cazul colaborării sau informării pe anumite subiecte ce pot ajuta la rezolvarea sau informarea în cadrul unui proiect sau rezolvării unei probleme.

Aplicaţia web are un rol dublu în cadrul sistemului. Poate fi accesat din afară de către oricine indiferent dacă este sau nu parte din organizaţia ce deţine sistemul, deoarece este format din doua module diferite. Primul este o aplicaţie de prezentare care serveşte atât ca punct de informare pentru vizitatori cât şi ca usă de intrare pentru cei autorizaţi în cel de-al doilea modul. Toate informaţiile din modulul de prezentare pot fi modificate de orice persoană care are autorizaţia necesară pentru a face modificări în toate paginile sau doar selectiv.

Exemplu: Personalul care se ocupă cu comunicatele de presă poate avea dreptul să modifice doar prima pagină care este pentru noutăţi şi comunicate, în timp ce un angajat de la resurse umane poate modifica doar pagina legată de oferte de lucru. Conducerea, desigur, poate modifica totul.

1)Logarea pe site se face in două moduri:
Cu aplicaţia desktop deschisă, utilizatorul accesează site-ul folosind butonul dedicat din aplicaţie sau pur şi simplu apelând browserul. în oricare din aceste situaţii, atâta timp cât aplicaţia Java este pornită, utilizatorul va fi logat automat.

2)Cu formularul de logare de pe partea de prezentare a site-ului.
Observatie 1: Nici site-ul, nici aplicaţia desktop nu au opţiune pentru logare permanentă din motive de securitate, deoarece dacă logarea este permanentă sau pe o perioadă de timp (exemplu: 2 saptămâni) atunci oricine accesează calculatorul respectiv poate folosi intregul sistem cu contul altei persoane.
Observatie 2:Aplicaţia php nu poate impiedica site-ul să reţină parola şi numele utilizatorului, folosirea acestei opţiuni rămâne la discreţia utilizatorului, noi recomandăm să nu se folosească această opţiune.
În urma logării, aplicaţia redirecţionează utilizatorul pe pagina sa principală unde apar toate informaţiile relevante utilizatorului. Cum ar fi:
- Directive/decizii adresate echipei/grupului/departamentului sau întregului personal;
- Mesaje private noi;
- Conferinţe in care a fost chemat;
- Lista de obiective (to-do list);
- Program de activităţi;
- Notiţe relevante accesării în funcţie de zi şi/sau relevanţei setate de utilizator;

Aplicaţia se bazează pe lucrul sau nevoile de zi cu zi ale utilizatorului în cadrul organizaţiei, comunicarea cu alţi colegi, schimb de informaţii irelevante conducerii sau organizaţiei, sistemului contabil/juridic şi/sau cel informaţional dar importante pentru ducerea la bun sfârşit a unui proiect.
Exemplu 1: O echipă care lucrează la un proiect, dar membrii săi nu sunt in acelaşi loc.
Exemplu 2: Apare o urgenţă iar persoana care o poate rezolva este in concediu sau acasă, în această situaţie aplicaţia dă voie utilizatorului să aibă acces la informaţii din baza de date necesare rezolvării problemei.

5 Relaţie Desktop-Web

Un rol foarte important în cadrul sistemului îl are comunicarea dintre aplicaţiile desktop şi site. Aceasta se realizează prin:
1. Accesul la baza de date comună, care reprezintă sursa principală de informaţii;
2. Cookiuri care ajută la logarea automată in site;
3. Aplicaţia poate genera dinamic un URL prin apelarea căruia aplicaţia desktop dă o comandă sau o cerere aplicaţiei web;
4. Aplicaţia oferă posibilitatea utilizatorilor de a lucra direct cu informaţia ce este disponibilă pentru ceilalţi angajaţi cât şi pentru accesul public oferă un plus de originalitate, reducând costul intreţinerii;

Câteva din caracteristicile oferite de sistem sunt:
a)Trimitere de mesaje private – între doi sau mai mulţi utilizatori. Mesajele private sunt structurate pe subiecte şi răspunsuri. Când selectezi un mesaj apar toate răspunsurile aferente acelui mesaj.
b)Acces de la distanţă la informaţii din baza de date (depinde de autorizaţia pe care o are utilizatorul in acest caz). Accesul la distanţă a bazei de date ajută la rezolvarea problemelor sau chiar şi lucrul când utilizatorul nu are acces la calculatorul de la servici.
c)Conferinţele au scopul de a ajuta la comunicarea intre mai mulţi emiţători-receptori care încearcă să rezolve o problemă. Conversaţiile sunt logate pentru referinţe ulterioare.
d)Anunţuri şi directive care sunt persistente pentru o perioadă de timp. Anunţurile şi directivele sunt afişate doar pe prima pagină dar doar cele relevante utilizatorului sau grupului din care face parte utilizatorul. Acestea sunt persistente fie până sunt vizualizate o dată fie până se termină o perioadă de timp setată de emiţător.

e)Logarea conversaţiilor din conferinţe şi schimbările din baza de date. Logarea conversaţiilor şi modificărilor in baza de date sunt necesare pentru a ajuta în cazul unei erori umane dar din motive de securitate (logarea nu ajută intotdeauna la recuperarea tuturor informaţiilor).
f)Lista de obiective (to-do list);
g)Program de lucru (agenda de lucru şi întâlniri – inclusiv conferinţe programate);
h)Notiţe.
Punctele (f, g, h) formează o agendă virtuală care ajută utilizatorul in diferite aspecte ale serviciului amintindu-i intalniri, conferinţe programate sau orice tip de informaţie.

Observaţie 1:Din motive de securitate recomandăm ca aplicaţia desktop să fie instalată doar pe calculatoarele din organizaţie, altfel apare riscul ca anumite informaţii confidenţiale să fie accesate de cineva din afară.

Observaţie 2:Aplicaţia web permite accesul doar la informaţiile neconfidenţiale dar cu posibilitatea ca in cazul unei urgenţe cineva mai sus pe scara ierarhică să poată da voie unui subordonat să acceseze anumite informaţii confidenţiale la un moment dat pentru o perioadă de timp.

6 Concluzii

Lucrarea prezintă un sistem de manipulare şi distribuţie a informaţiei care este construit pe baza unei idei originale, care oferă o mai mare importanţă informaţiei în cadrul entităţilor mai mici şi care nu au un buget mare pentru aşi permite firme specializate sau personal calificat pentru un sistem complex de gestiune a informatiei.

Organizarea logică a sistemului precum şi implementarea acesteia sunt originale. Sistemul are o sferă largă de utilizare atât în instituţii ,firme mici, instituţii educaţionale dar poate fi folosită şi în scop personal.Accentul este pus pe utilizator, oferindu-i o interfată usor de utilizat, putere de accesare şi distribuire rapidă a datelor.O caracteristică deosebit de utilă a sistemului este aceea de flexibilitate, sistemul putând fi adaptat uşor la ierarhia oricărei instituţii.

Bibliografie

A. Tratate, monografii, cursuri universitare şi alte lucrări de specialitate

[1] NICOLESCU, Ovidiu, „Management”, Editura Didactică si Pedagogică, Bucuresti, 1992
[2] THOMAS, M. Todd, „Java Data Access – JDBC, JNDI and JAXP”, M&T Books, New York, 2002
[3] FRASINARU, Cristian, „Curs practic de Java”
[4] KEOGH, Jim, „Javascript fara mistere”, Rosetti Educational, Bucuresti, 2006
[5] ULLMAN, Larry, „PHP si MySQL pentru site-uri web dinamice”, Teora, Bucuresti,2006

B. Surse Internet

[1] 1.http://dev.mysql.com/doc/
[2] 2.www.bcu-iasi.ro/biblos/biblos13/sistinf.pdf
[3] 3.http://en.wikipedia.org/wiki/Business_intelligence
[4] 4.http://en.wikipedia.org/wiki/Business_model

RUSU Andrei
Universitatea Lucian Blaga, Sibiu
Facultatea de Ştiinţe, Sibiu
Str. Dr. Ion Ratiu, 5-7
ROMANIA
rusu.andy1@gmail.com

VOLOSINCU Mihai Bogdan
Universitatea Lucian Blaga, Sibiu
Facultatea de Ştiinţe, Sibiu
Str. Dr. Ion Ratiu, 5-7
ROMANIA

Sesiunea Naţională de Comunicări Ştiinţifice a Studenţilor
"Imaginaţie, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Agent SQL Express

Mihai Stancu
Coordonator: Prof. Univ. Dr. Dana Simian

Abstract

Această lucrare prezintă designul unui Agent SQL pentru Microsoft SQL Server 2005/2008 Express. Agentul poate planifica rularea unei secvenţe de comenzi T-SQL sau a unei proceduri stocate la anumite intervale stabilite de utilizator. Agentul foloseşte tabele pentru a salva informaţiile despre operaţii şi nu apelează la servicii Windows sau la Windows Task Scheduler pentru planificarea operaţiilor. Scopul acestei lucrări este de a propune şi implementa o modalitate nouă de a planifica operaţii pe baze de date folosind o tehnologie Microsoft disponibilă gratuit, numită Service Broker. Rezultatul este o aplicaţie robustă, flexibilă, extrem de uşor de folosit şi cu un potenţial imens pentru creşterea productivităţii şi sporirea securităţii bazelor de date.

1 Introducere

Microsoft SQL Server 2005 Express este versiunea gratuită a serverului SQL dezvoltat de Microsoft. Deşi este compatibilă cu versiunile comerciale, varianta express a serverului SQL prezintă totuşi anumite limitări, dintre care menţionăm absenţa serviciului SQL Server Agent. Din acest motiv planificarea joburilor nu este posibilă fără a instala produse soft terţă parte, care sunt adesea foarte costisitoare. Anumite firme nici nu permit instalarea acestor aplicaţii din cauza politicii lor de securitate, ceea ce reprezintă o problemă majoră atât pentru administratori cât şi pentru utilizatori.

Acesta este motivul pentru care am ales să realizăm o aplicaţie care să permită întreţinerea bazelor de date folosind numai serviciile oferite de SQL Server 2005/2008 Express. Soluţia dezvoltată este formată din două părţi importante. Prima parte este reprezentată de scriptul SQL folosit pentru crearea tabelelor, a procedurilor stocate şi a obiectelor de planificare, iar cea de-a doua este reprezentată de interfaţa cu utilizatorul, creată în Visual Basic 2005, folosind platforma .NET Framework 2.0 şi Windows Forms.

2 Modul de planificare a operaţiilor

Pentru a realiza planificarea am folosit Microsoft Service Broker, tehnologie ce face parte din motorul de baze de date al serverului SQL. Acest serviciu oferă o platformă de comunicare bazată pe mesaje care permite unor componente independente ale aplicaţiei să funcţioneze ca un întreg. Este o adiţie importantă adusă serverului SQL dezvoltat de Microsoft, prezentând o mulţime de facilităţi noi precum cozi de aşteptare, proceduri de activare, mesaje, contracte, conversaţii, etc.

Obiectul Service Broker care a face posibilă planificarea operaţiilor în aplicaţia noastră este timerul de conversaţie. După setare, timerul aşteaptă un anumit număr de secunde, după care trimite un mesaj către coada locală, de asemenea un obiect Service Broker. De îndată ce mesajul ajunge în coadă, este lansată procedura de activare asociată, care primeşte mesajul din coada de aşteptare şi execută secvenţa de comenzi SQL sau procedura stocată stabilită de utilizator.

Pentru a implementa agentul, am creat următoarele obiecte:

1. Tabelele pentru păstrarea informaţiei despre fiecare operaţie
2. Procedurile stocate pentru crearea, rularea şi ştergerea operaţiilor.
3. Contractul, coada de aşteptare şi serviciul de planificare, bazat pe timerul de conversaţie

2.1 Tabelele

Două tabele păstrează informaţiile despre operaţiile planificate şi despre posibilele erori care pot apărea în momentul rulării. Pentru a descrie în detaliu tipul informaţiilor reţinute în aceste tabele, prezentăm mai jos instrucţiunile SQL folosite pentru crearea acestora:

```
CREATE TABLE tblOperatiiPlanificate
(
	ID INT IDENTITY(1,1),
	JobName nvarchar(100) NOT NULL,
	JobDescription nvarchar(max) NOT NULL,
	ScheduledSql nvarchar(max) NOT NULL,
	ScheduleType INT NOT NULL,
	FirstRunOn datetime NOT NULL,
	LastRunOn datetime,
	LastRunOK BIT NOT NULL DEFAULT (0),
	IsRepeatable BIT NOT NULL DEFAULT (0),
	IsEnabled BIT NOT NULL DEFAULT (0),
	ConversationHandle uniqueidentifier NULL
)

CREATE TABLE tblEroriOperatiiPlanificate
(
	Id BIGINT IDENTITY(1, 1) PRIMARY KEY,
	ErrorLine INT,
	ErrorNumber INT,
	ErrorMessage NVARCHAR(MAX),
	ErrorSeverity INT,
	ErrorState INT,
	ScheduledJobId INT,
	ErrorDate DATETIME NOT NULL
	DEFAULT GETUTCDATE()
)
```

2.2 Procedurile stocate

Aplicaţia foloseşte trei proceduri stocate pentru crearea, ştergerea, respectiv rularea operaţiilor pe serverul SQL.

spAdaugaOperaţiePlanificata adaugă o înregistrare pentru o operaţie nouă în tabelul tblOperaţiiPlanificate, crează o nouă conversaţie şi setează un timer pentru aceasta. Adăugarea conversaţiei se face într-o tranzacţie deoarece dorim ca această operaţie să fie atomică.

spStergeOperaţiePlanificata realizază eliminarea unei operaţii, pe baza id-ului acesteia. Sfârşeşte conversaţia asociată şi şterge înregistrarea corespunzătoare din tabelul tblOperaţiiPlanificate. Toate aceste operaţii se realizează de asemenea în cadrul unei tranzacţii.

spRuleazaOperatiePlanificata este procedura stocată de activare asociată coadei de aşteptare. Aceasta rulează operaţie şi actualizează tabelul tblOperaţiiPlanificate. Nu este tranzacţională deoarece erorile sunt înregistrate în tabelul tblEroriOperaţiiPlanificate, şi nu dorim ca mesajul DialogTimer să fie returnat în coadă, ceea ce ar duce la un ciclu infinit.

2.3 Contractul, coada şi serviciul

Enumerăm mai jos obiectele Service Broker folosite, şi modul în care au fost create:

[//ScheduledJobContract] este contractul care permite trimiterea mesajelor DialogTimer:

```
CREATE CONTRACT [//ScheduledJobContract]
([http://schemas.microsoft.com/SQL/ServiceBroker/DialogTimer] SENT BY INITIATOR)
```

[//ScheduledJobQueue] este coada folosită pentru a posta mesajele DialogTimer şi pentru a rula procedura de activare usp_RunScheduledJob pentru execuţia unei operaţii. Este creată astfel:

```
CREATE QUEUE ScheduledJobQueue WITH STATUS=ON, ACTIVATION
(PROCEDURE_NAME = usp_RunScheduledJob, MAX_QUEUE_READERS=20, EXECUTE AS 'dbo')
```

[//ScheduledJobService] este un serviciu peste coada ScheduledJobQueue, controlat de contractul [//ScheduledJobContract]. Instrucţiunea TSQL folosită pentru crearea lui este:

```
CREATE SERVICE [//ScheduledJobService] AUTHORIZATION dbo
ON QUEUE ScheduledJobQueue ([//ScheduledJobContract])
```

3 Interfaţa

În această secţiune descriem interfaţa planificatorului de operaţii al agentului.

Forma principală conţine un control de tip DataGridView care afişează coloanele importante ale tabelului tblOperaţiiPlanificate. Operaţiile care pot fi executate asupra joburilor pot fi accesate prin intermediul meniului principal sau prin meniul contextual. În cazul în care se alege Adaugă, se deschide cea de-a doua formă, care oferă utilizatorului posibilitatea de a crea şi planifica operaţia potrivit unui orar zilnic, săptămânal sau lunar. Pentru scrierea comenzilor SQL, utilizatorii au la dispoziţie un editor integrat asemănător aplicaţiei Notepad, accesibil prin intermediul butonului Deschide Editor. Dupa adăugarea unei operaţii noi, forma principală este redimensionată corespunzător. Acelaşi lucru se întâmplă şi la ştergerea unei operaţii. Editarea foloseşte aceeaşi formă ca cea folosită pentru adăugare, însă logica este uşor modificată.

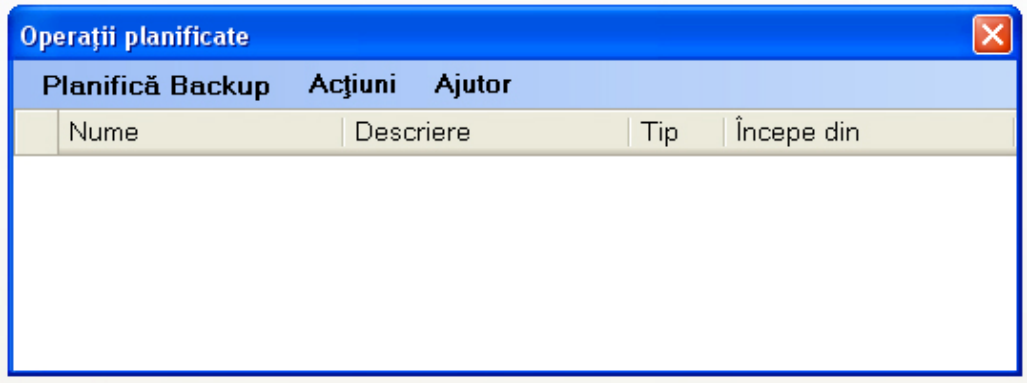


Fig. 1 Fereastra principală

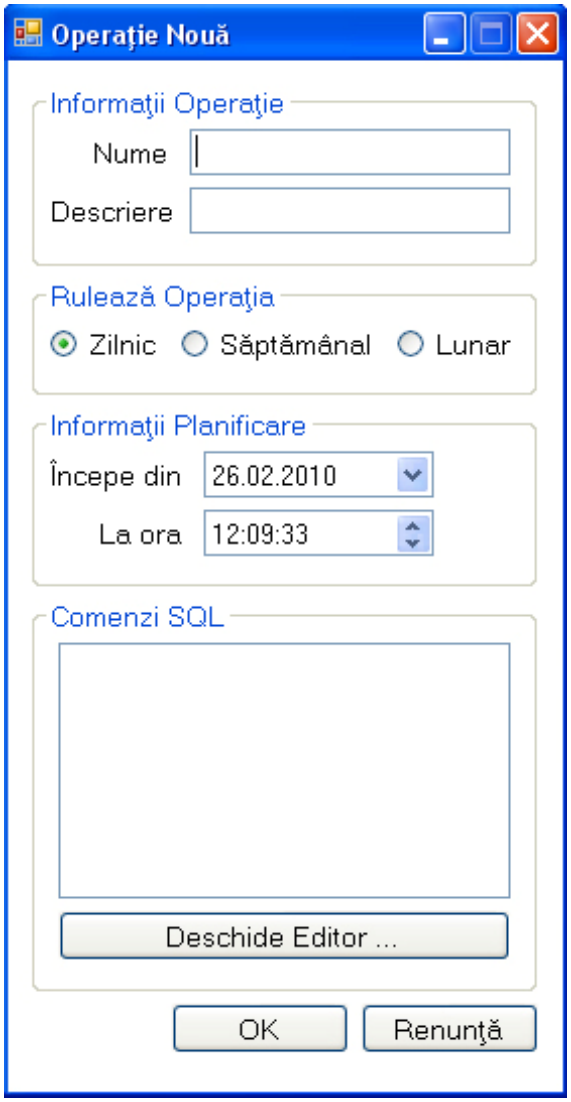


Fig. 2 Fereastra de adăugare operaţie

Fig. 3 Fereastra de planificare backup

Interfaţa a fost realizată având ca scop principal transparenţa acţiunilor şi uşurinţa de executare a acestora. Pentru a seta valorile diferitelor controale se pot folosi fie numai tastatura, fie numai mouseul, sau o combinaţie a acestora. Astfel un utilizator poate crea sau edita o operaţie în numai câteva secunde, reducând mult timpul necesar întreţinerii bazelor de date.

4 Concluzii

În această lucrare am prezentat designul şi implementarea unui agent SQL care permite crearea şi planificarea unor operaţii pe un server Microsoft SQL 2005 sau 2008 Express. Aplicaţia creată este extrem de uşor de folosit, flexibilă şi robustă. Spre deosebire de celelalte produse soft asemănătoare, sistemul nostru de planificare foloseşte Service Broker, un serviciu gratuit, integrat chiar în motorul de baze de date al serverului SQL de la Microsoft. Acest fapt sporeşte siguranţa aplicaţiei şi elimină majoritatea erorilor care ar putea apărea în timpul rulării.

5 Planuri de dezvoltare viitoare

Am arătat că acest tip de aplicaţie aduce o mulţime de beneficii, şi au fost depuse numeroase eforturi pentru sporirea utilizabilităţii, însă există o serie de îmbunătăţiri ce pot fi aduse, şi care sunt planificate pentru versiunile următoare. Dintre acestea menţionăm:

- folosirea unui sistem de planificare mai avansat care să permită utilizatorului să ruleze o operaţie în anumite zile ale săptămânii sau ale lunii (de exemplu: luni, miercuri şi vineri la fiecare două săptămâni, ultima zi de vineri în fiecare lună, etc.)

- atribuirea mai multor paşi unui singur job, fiecare având o anumită sarcină, lucru folositor în cazul în care, de exemplu, utilizatorul doreşte ca la anumite intervale de timp, să realizeze un backup la o bază de date, să arhiveze şi să încripteze acel backup, să copieze arhiva încriptată la o anumită locaţie partajată în reţea, iar apoi să şteargă backupurile care sunt mai vechi decât o anumită dată specificată.

- rularea unui job de un număr specificat de ori în cazul apariţiei unor erori la rulare (în prezent dacă un job sfârşeşte cu erori, este efectuat un rollback, se înregistrează eroarea în tabelul de erori şi se stabileşte data următoare la care va rula jobul)

- realizarea unui sistem de notificări pentru utilizatorii care doresc, de exemplu, să primească un email în cazul rulării cu succes sau dimpotrivă, în cazul în care apare o eroare în timpul rulării unui job.

- includerea unui asistent, care să permită unui utilizator începător să planifice mai uşor anumite operaţii des întâlnite, precum crearea unui backup, reindexarea unei baze de date, actualizarea unor statistici, etc.

Bibliografie

[1] Klaus Aschenbrenner, *Pro SQL Server 2005 Service Broker*, Apress, New York, 2007
[2] Roger Wolter, *Rational Guide to SQL Server 2005 Service Broker*, Rational Press, Rollinsford, 2006
[3] SQL Server 2005 Books Online, Service Broker
[4] Roger Wolter, *Manage Your Tasks With Service Broker*, TechNet Magazine http://207.46.16.252/en-us/magazine/2005.05.servicebroker.aspx, 24 februarie 2010
[5] Mladen Prajdic, *Scheduling Jobs in SQL Server Express*, SQL Team http://www.sqlteam.com/article/scheduling-jobs-in-sql-server-express, 24 februarie 2010

STANCU MIHAI
Universitatea „Lucian Blaga” din Sibiu
Facultatea de Ştiinţe
Specializarea Matematica-Informatică
Bd-ul. Victoriei, Nr. 10, Sibiu, 550024
ROMANIA
E-mail: stancu_meehigh@yahoo.com

Sesiunea Națională de Comunicări Științifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

AI in 3D Gaming - AICon

Lucian Stoica
Coordonator: Prof. dr. Dana Simian

Abstract

The paper presents Artificial Intelligence (also known as AI) that is used in 3D gaming, and describes how an application named AICon (a 3D game) was developed using different tools and algorithms. Some of these tools are: Irrlicht for graphics, IrrKlang for sound and Newton Dynamics for physics. Movements and decisions taken by soldiers in AICon game represent "the core" of this application. AICon has a built-in 3D map editor that can be useful with others tools for creating 3D applications. This editor can work with 3D models objects that was created with specialized tools like Maya, 3D Studio Max, Collada, .X, Milkshape, etc.

1 Prezentarea generală a aplicatiei AICon

AICon este un joc cu grafica tridimensionala destinat tuturor iubitorilor de jocuri de tip First person Shooter (FPS) care isi are originea intr-unul dintre cele mai „iubite" si jucate jocuri din toate timpurile: Counter-Strike. Oare cine nu s-a jucat macar o data acest fel de joc? Fie ca s-a numit Doom, Quake sau Counter-Strike. Jocul este structurat pe misiuni cu un grad de dificultate ascendent dupa cum urmeaza:

- in prima misiune trebuie ca jucatorul sa ajunga la o anumita locatie de pe harta folosind pentru deplasare tastele WASD si mouse-ul,
- in cea de-a doua misiune trebuie sa-l lichideze pe seful teroristilor, fara a omori insa vreun alt dusman,
- in cea dea treia misiune trebuie sa planteze o bomba in baza teroristilor si sa o detoneze, dupa care trebuie sa ajunga la locul de extractie pentru a fi ridicat de un elicopter.

Jucatorul trebuie sa duca la bun sfarsit fiecare misiune cu orice pret.
Bineinteles ca ordinea misiunilor sau chiar editarea hartilor si a misiunilor este posibila, deoarece AICon poseda propriul editor de harti tridimensionale, care este specializat si modelat intocmai pentru acest lucru. Jocul mai ofera utilizatorului posibilitatea de a-si alege driver-ul folosit la randare (OpenGL, DirectX, sau Software), sa-si configureze tastatura si mouse-ul, sunetul, imaginea, etc.

Sunetul pentru un astfel de joc tridimensional, nu putea fi decat unul tridimensional, si asta se datoreaza bibliotecii IrrKlang care ne ofera o multitudine de functii specializate in lucrul cu sunete.

De departe, una dintre cele mai frumoasa parti ale acestei aplicatii o reprezinta Inteligenta Artificiala, pentru ca asa cum spuneam la inceputul acestei prezentari, inteligenta artificiala este „miezul si sufletul" aplicatiei, si la urma urmei, ce-ar fi un joc pe calculator fara inteligenta

artificiala? Inteligenta bot-ilor, in acest joc, se poate seta pe o scara de la unu la zece, in sensul ca valoarea unu reprezinta un grad de inteligenta scazut pentru un bot (un soldat „controlat” de calculator), adica aceasta valoare se poate folosi de catre cei care nu s-au mai jucat pana acum un astfel de joc, iar o valoare ridicata cum ar fi opt, face din bot-i niste adevarati luptatori capabili sa elimine orice dusman, si se poate folosi de cei mai experimentati jucatori.

2 Prezentarea editorului de harti tridimensionale

Dupa cum am mai spus AICon poseda propriul editor de harti tridimensionale, editor care poate incarca si folosi modele 3D create cu alte unelte de modelare 3D cum ar fi: Maya, 3D Studio Max, Collada, Milkshape, etc. S-a optat pentru posibilitatea folosirii mai multor tipuri de modele tridimensionale, deoarece unii utilizatori se descurca mai bine in editoare de genul 3D Studio Max si sunt capabili sa-si creeze propriile modele tridimensionale, dar majoritatea utilizatorilor nu stiu sa foloseasca un asemenea editor si din aceasta cauza pot descarca gratuit modele de pe internet pentru ca in prezent informatia este cum nu se poate mai ieftina (traim intr-o „era open-source”).

In figura 1 este prezentat editorul de harti, si se poate observa cu usurina ca cel care foloseste acest editor nu trebuie sa posede cunostinte speciale despre lumea tridimensionala.

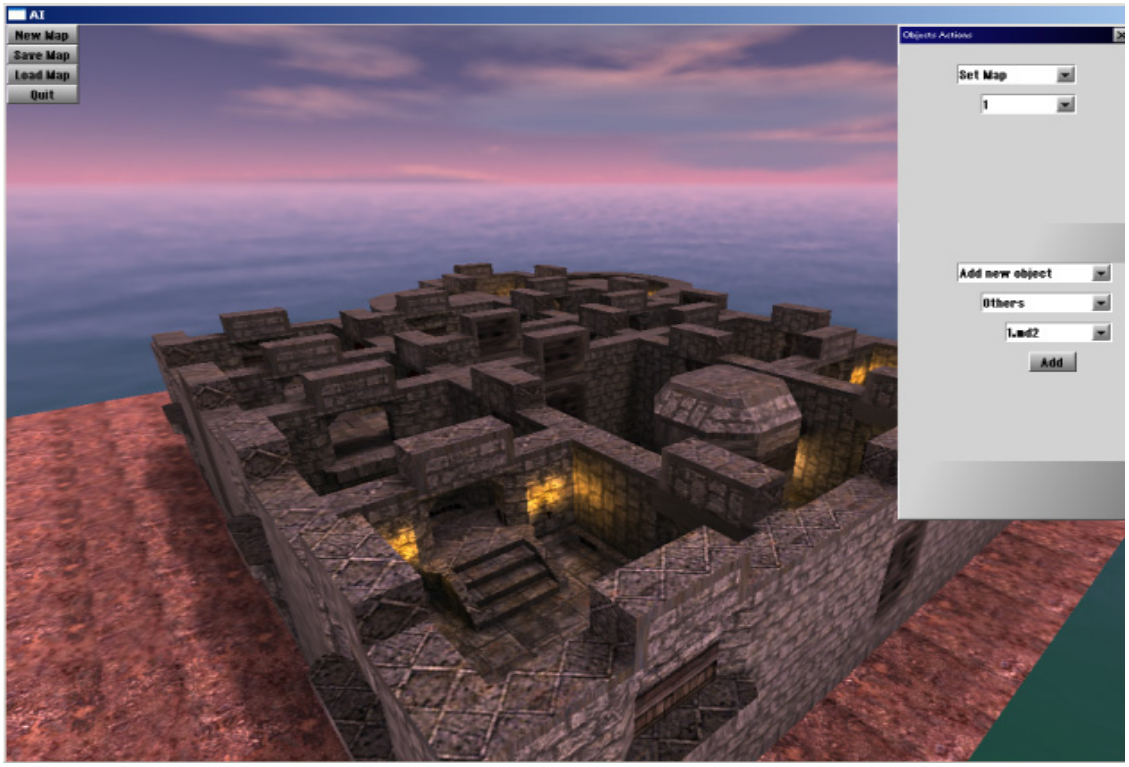


Fig. 1: Editorul AICon

In primul combobox din coltul dreapta-sus al editorului avem urmatoarele optiuni la alegere:

- **Set Map** permite incarcarea si folosirea unei harti (model tridimensional) create anterior cu unul dintre utilitarele mai sus mentionate (Maya, 3D Studio Max, etc.), alegerea unei astfel de harti facandu-se cu combobox-ul situat putin mai jos.
- **Set Sky Box** permite folosirea unui „tablou inconjurator” pentru harta curenta care de cele mai multe ori reprezinta un peisaj din natura sau din mediul urban. Un skybox reprezinta „un cub”

in care toata harta este „inchisa”, iar apropierea de acest „cub” este practic imposibila, deoarece acesta este decorativ.

- **Save My Position** ofera utilizatorului posibilitatea de a salva pozitia pe harta curenta, pozitie din care se va pleca in indeplinirea misiunii (pozitia de start a soldatului).
- **Finish Condition** reprezinta conditia de terminare a unei misiuni, iar din bombobox-ul de mai jos se poate alege Find Location, adica jucatorul castiga daca a ajuns la locatia definita de cele trei puncte (XYZ), Kill All, inseamna ca trebuie ucisi toti teroristii pentru a indeplini cu succes misiunea, Plant Bomb, reprezinta plantarea bombei in punctul de coordonate XYZ.
- **Add New Object** pentru a putea adauga obiecte pe o harta existenta. Va aparea un nou combobox din care se poate alege tipul obiectului (planta, soldat, vehicul,...) si obiectul propriu-zis (de exemplu: Add New Object->Vehicles->Hummer)
- **Deselect** pentru a deselecta obiectul curent selectat, deoarece la adaugarea unui nou obiect acesta devine selectat si toate actiunile care au loc, se intampla pe acel obiect.
- **Set Texture** pentru a putea selecta o noua textura care dorim s-o aplicam pe un obiect. In figura 2 este prezentata o masina de teren cu o textura de tip camuflaj, iar in figura 3 este reprezentata aceeasi masina (de oras) dar cu alta textura. In acest mod se poate crea un singur obiect tridimensional si mai multe texturi, avand astfel o multitudine de obiecte care difere doar prin textura.

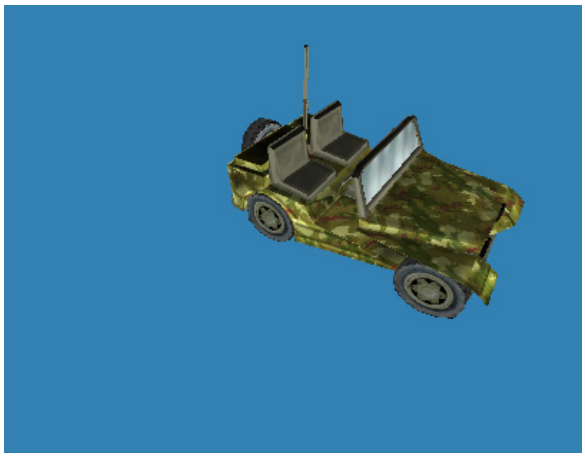


Fig. 2: Masina „camuflaj”

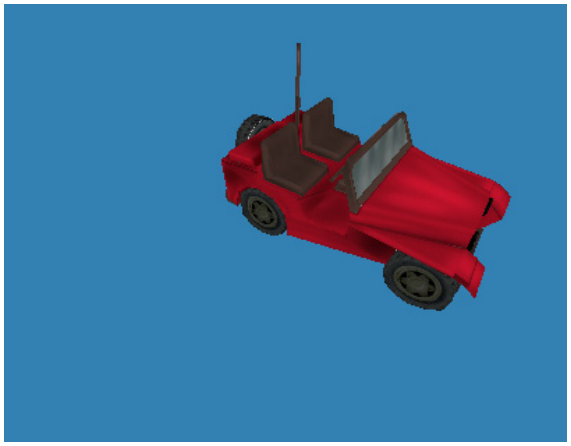


Fig. 3: Masina „de oras”

- **Scale** ne da posibilitatea scalarii (maririi/micsorarii) obiectelor pe toate cele trei axe (XYZ).
- **Rotate** pentru rotirea obiectelor pe cele trei axe.
- **Move** pentru a putea muta in spatiu un obiect selectat.
- **Delete** ofera posibilitatea stergerii unui obiect (obiectul curent selectat).
- **Delete All** folosit pentru stergerea tuturor obiectelor adaugate.

Butoanele din stanga ferestrei aplicatiei reprezinta:
- **New Map** crearea unei noi harti folosind toate functiile si uneltelt mai sus mentionate,
- **Save Map** salvarea hartii curente intr-un fisier pe hard-disk care mai apoi se poata incarca si folosi ca o posibila misiune in joc,
- **Load Map** incarca o harta care a fost creata anterior si pe care se pot adauga noi elemente pentru a creste complexitatea unei misiuni, de exemplu,
- **Quit** permite iesirea din aplicatie.

Unul dintre atuurile acestui editor este faptul ca utilizatorul poate defini un poligon (un dreptunghi) care sa-i dea proprietatea de „apa”, iar acel poligon se va comporta ca o suprafata lichida, de apa, adica va avea niste valuri cu o inaltime definita de utilizator, sau se pot defini si flacari, adica se poate incarca un model tridimensional care sa reprezinte o torta, iar pe aceasta torta se va defini flacara cu intensitate, culoare, directia pe care sa emita lumina, etc.

In urmatoarea versiune a acestui joc, editorul va fi capabil sa incarce harti Counter-Strike, pe care se vor adauga elemente dinamice (apa, foc, obiecte care interactioneaza cu mediul, etc.).

3 Prezentarea jocului AICon

AICon este un joc de tip First Person Shooter (FPS), este impartit pe misiuni care difera ca grad de complexitate in functie de nivelul atins de jucator. Jucatorul este un soldat care trebuie sa duca la bun sfarsit mai multe misiuni cu orice pret. Daca o misiune esueaza, atunci nu va putea trece la urmatoarea misiune. Deplasarea in lumea tridimensionala se face folosind tastele WASD si mouse-ul.

Pentru a sari peste diferite obstacole se foloseste tasta Space, iar pentru schimbarea armelor se folosesc tastele 1 – 9. La inceputul jocului, player-ul are o suma mica de bani cu care poate cumpara doar un pistol, dar pe masura ce termina mai multe misiuni primeste mai multi bani si isi poate cumpara diferite arme cum ar fi: AK47, M16, AWP, ShotGun, Grenade, Smoke Grenade, FlashBang, etc. Bineinteles ca daca va incalca anumite reguli va fi penalizat si i se vor retrage bani. Fiecare misiune trebuie dusa la capat intr-un anumit interval de timp, altfel jucatorul va fi penalizat si va trebui sa inceapa din nou acea misiune.

In figura 4 se pot vedea urmatoarele:
- **Health** sau sanatatea soldatului controlat de utilizator care in cazul nostru este 80 (coltul stanga-jos) si este reprezentata de o cruce rosie,
- **Kevlar & Helmet** sau armura soldatului, care este de 50 si este reprezentata de un scut albastru
- **Time Remaining** sau timpul ramas pentru a indeplini misiunea curenta (9 minute si 39 secunde), si este reprezentat de un ceas,
- **Weapon Clips** ne ofera informatii despre numarul de cartuse existent in incarcatorul armei (care este 10 in cazul nostru) si numarul de cartuse disponibile ramase (30),
- **Crosshair** sau tinta, situata in centrul imaginii si este de culoare albastra

Fig. 4: Informatii utile jucatorului

Soldatii condusi de catre calculator (sau bot-i cum mai sunt numiti), au o inteligenta artificiala care difera de la o misiune la alta, sunt capabili sa navigheze in lumea tridimensionala, sa ridice obiecte de jos (arme de exemplu), sa il „vada” pe soldatul condus de catre utilizator, sa traga in acesta, sa se ascunda de el (dupa o lada, un stalp sau un perete), sau chiar sa fuga de el.

4 Inteligenta artificiala in AICon

De departe, una dintre cele mai frumoase parti ale acestei aplicatii, atat ca carte de implementare, cat si ca grad de dificultate, deoarece cu cat un proiect este mai dificil de implementat cu atat este rezultatul obtinut mai frumos si apreciat. AICon detine elemente de inteligenta artificiala in sensul ca soldatii stiu sa navigheze prin spatiul tridimensional fara sa se ciocneasca de peretii hartii sau alte obstacole, stiu sa „vada” obiectele existente intr-o harta (arme, alti soldati, lazi, etc.), sunt capabili sa schimbe o arma mai slaba cu una mai performanta, sa pazeasca un perimetru, sa traga dupa soldatul jucatorului, sa fuga de acesta, etc.

Pentru a „invata” un soldat sa navigheze prin lumea tridimensionala s-a apelat la un motor fizic, si anume Newton Dynamics, motor care ne ofera informatii despre coliziunile dintre obiecte (locul de intersectie a doua modele tridimensionale), sau rezolva, de exemplu aruncarea unei grenade (ciocnire, deviere, acceleratie gravitationala). Pentru ca acest lucru sa fie posibil a trebuit ca fiecare mesh al fiecarui model tridimensional sa fie desfacut in triunghiuri si sa fie adaugat intr-o lume virtuala numita „lumea newton”. Deplasarea soldatilor pe harta se face random in sensul ca acestia isi aleg niste puncte in care doresc sa ajunga si se deplaseaza spre acele puncte, utilizatorul nefiind astfel capabil sa prevada ceea ce vrea sa faca „calculatorul”. Aceste puncte care se aleg la intamplare, trebuie sa nu intersecteze harta, adica sa nu treaca brusc prin pereti sau alte obstacole, altfel „calculatorul” ar intra prin pereti si ar aparea un efect ciudat. In acest sens,

cand „calculatorul” isi alege un punct spre care sa navigheze foloseste un „sonar”, adica duce niste linii imaginare inspre punctul ales si daca acele linii intersecteaza vreun plan, atunci punctul de destinatie pe care l-a ales initial se modifica pana cand devine un punct valid. Cand se intalnesc doi soldati, atunci cel din stanga are prioritate fata de cel din dreapta, ceea ce inseamna ca cel din dreapta trebuie sa astepte pana cand cel din stanga a plecat de langa el, dupa care isi poate continua drumul. In figura 5 sunt 3 soldati care se deplaseaza pe o portiune din harta, pe o raza de 15m fara sa se ciocneasca intre ei folosind „sonarul”. Se poate vedea ca doi dintre acesti soldati au o linie verde in fata lor ceea ce inseamna ca se pot deplasa pe traiectoria acelei linii deoarece aceasta nu intersecteaza niciun plan, pe cand soldatul din dreapta are o linie rosie, adica el se poate deplasa pe o distanta mai scurta, deoarece linia sa virtuala intersecteaza cladirea.

Fig. 5: Deplasarea soldatilor folosind „sonarul”

Bibliografie

[1] BACIU, Rodica, *Programarea aplicatiilor grafice 3D cu OpenGL*, Editura Albastra, Cluj-Napoca, 2005.
[2] HARBOUR, Jonathan S., *Programarea jocurilor in Visual Basic*, Editura Rosetii Educational, Bucureşti, 2006.
[3] http://irrlicht.sourceforge.net/
[4] http://www.ambiera.com/irrklang/
[5] http://nehe.gamedev.net/
[6] http://www.opengl.org/

LUCIAN STOICA
Universitatea „Lucian Blaga”, Sibiu
Master Tehnologia Informatiei
Str. Dr. I. Ratiu, Nr. 5-7
ROMANIA
E-mail: luci15ian@gmail.com

Sesiunea Naţională de Comunicări Ştiinţifice a Studenţilor
"Imaginaţie, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Aplicaţie Android pentru Localizarea Persoanelor cu Ajutorul Google Maps

Monica STOICESCU
Coordonator: Asist. univ. Ing. Alexandru RADOVICI

Abstract

This paper presents an efficient way of keeping track of your friends using locating services and Google Maps on Android mobile devices.

1. Introducere

În prezent cu toţii resimtim nevoia de comunicare cu cele mai apropiate persoane indiferent unde ne-am afla, iar ştiinta vine in sprijinul nostru cu cele mai inovatoare tehnologii pentru a realiza acest lucru. Platforma Android pentru telefoane mobile, aflată în continuă dezvoltare in ultimii ani, pune la dispoziţia utilizatorilor aceste competenţe cu ajutorul claselor din pachetele *android.location* si *com.google.android.maps* din biblioteca externă Google Maps.
O modalitate rapidă şi eficientă de a afla în orice moment unde se află o anumită persoană poate fi extreme de utilă şi comodă întrucât se economisesc atât timp cât şi resurse financiare şi chiar stresul apelurilor telefonice.
Din ce în ce mai mult, oamenii vor accesa servicii bazate pe internet folosind mijloace asa-numite "netradiţionale" cum ar fi dispozitivele mobile.
Android este platforma pentru dispozitive mobile dezvoltată de Google. Sistemul de operare este Linux.
Programarea Android se realizează folosind sintaxa Java, plus o bibliotecă de clase care formează o submulţtime a bibliotecii Java SE, plus extensii specific Android. Codul Java compilat - împreună cu orice alte date sau resurse necesare aplicaţiei - constituie un pachet, o arhivă cu extensia .apk. Sub această formă este distribuită şi instalată pe telefoane.
Componentele unei aplicaţii:

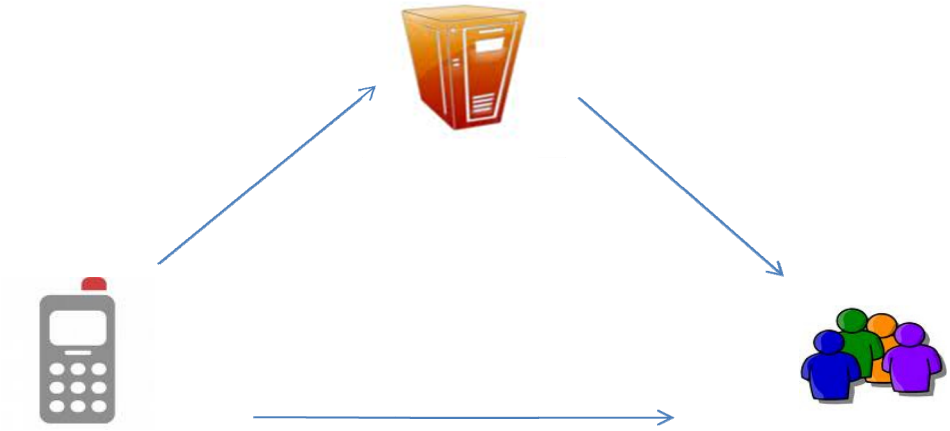
- *activităţi* – interfeţe similare ferestrelor. O aplicaţie poate conţine una sau mai multe activităţi independente, fiecare reprezentând o subclasă a clasei Activity. Conţinutul vizual este reprezentat de obiecte ale clasei View grupate în containere;
- *servicii* – nu au o interfaţă ci rulează în background pentru o perioadă nedeterminată de timp;
- *receptori* – primesc şi reacţionează la anunţuri; extind clasa BroadcastReceiver;
- *baze de conţinut* – creează un set de date valabile altor aplicaţii;
- *intenţii* – obiecte ale clasei Intent.

Android vine cu un număr de caracteristici care ajută la dezvoltarea aplicaţiilor:

- stocare
- reţea
- multimedia
- servicii de localizare
- servicii de telefonie

2. Mod de funcţionare

Aplicaţia propusă creează o listă a persoanelor care interesează pe utilizator, iar localizarea lor se realizează în timp real. Datele fiecărui utilizator sunt preluate dintr-un server.
Lista este creată chiar de la pornirea aplicaţiei, afişând persoanele aflate în acel moment într-o anumită rază (de exemplu toţi cei care se află la maxim 100 km depărtare), dispuse în ordinea crescatoare a distanţei. Alături de nume şi distanţă, un element al listei mai conţine o poză şi un status pentru a primi informaţii mai exacte cu privire la locul în care se află şi activitatea pe care o desfaşoară persoana respectivă. Această activitate poartă numele de FriendsList şi extinde clasa ListActivity. Din aceasta, printr-o opţiune a meniului se poate trece la vizualizarea unei hărţii propriu-zise (MapActivity), în care se poate vedea exact poziţia fiecărei persoane.

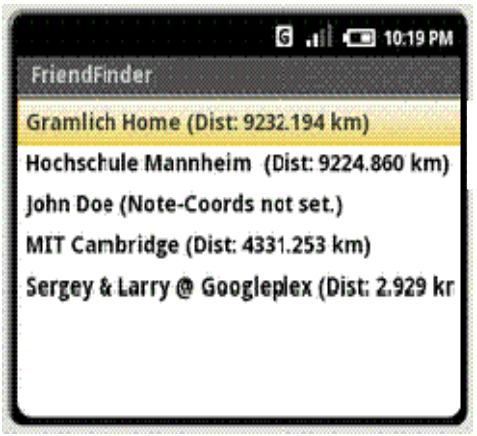


2.1 ListActivity

ListActivity este o activitate care afişează o listă de elemente prin conectarea la o sursă de date. Componenta sa principală este ListView, iar interfaţa este creată într-un fişier XML.
ListView este una dintre cele mai importante componente ale unei aplicaţii, pur şi simplu pentru că este foarte des utilizată.
În cazul de faţă *FriendsList* este o subclasă a ListActivity, iar fiecare element al listei reprezintă un obiect de tip *Friend.* Metode utilizate sunt *refreshFriendsList()*, apelată la pornirea aplicaţiei;

updateList(), actualizează distanţele la un anumit interval de timp sau la o anumită distanţă parcursă şi *setupForGPSAutoRefreshing()*.
Unele dintre cele mai populare caracteristici ale dispozitivelor mobile în ziua de astăzi sunt serviciile de localizare. Dispozitivele Android pot avea acces la mai multe mijloace de determinare a poziţiei, dintre care unele pot fi mai precise decât altele. Unele pot fi accesate gratuit, în timp ce altele vor avea un cost asociat; unele pot transmite mai mult decât poziţia curentă (ex. altitudinea sau viteza de deplasare). Astfel toate aceste servicii sunt grupate într-un set de obiecte de tip *LocationProvider*.
Componenta centrală a structurii de localizare este serviciul *LocationManager* care furnizează un API (Application Programminc Interface) pentru determinarea poziției. Prin apelarea *getSystemService(Context.LOCATION SERVICE)* se returnează calea spre o nouă instanţă LocationManager. Odată realizat acest lucru, aplicația va fi capabilă de urmatoarele lucruri:

- Interogare pentru lista tuturor instanţelor de tip LocationProvider cunoscute de LocationManager la ultima sa localizare;
- Înregistrare pentru update-uri periodice ale poziţiei curente de la un LocationProvider (specificat prin criteriu sau nume);
- Înregistrare pentru lansarea unei intenții în cazul in care dispozitivul se află într-o anumită rază dată (în metri).



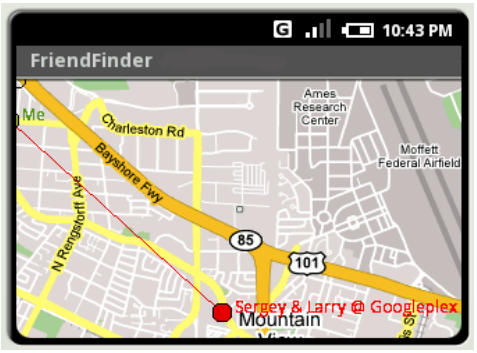
(model)

2.2 MapActivity

Un alt serviciu extrem de des utilizat este Google Maps, integrat si de Android. Pentru implementarea unei hărti este nevoie de o activitate de tip MapActivity şi de o componentă de tip MapView (subclasă a clasei ViewGroup, care afisează o harta cu date obținute prin serviciul Google Maps).
Google furnizează biblioteca externă Maps ce contine pachetul *com.google.android.maps*, oferind o varietate de optiuni de afisare si control. Clasele principale din acest pachet sunt:

- *MapView*;
- *MapController* - oferă toate elementele de interfată necesare pentru controlul utilizatorului (in special zoom-ul);
- *Overlay* – elementele grafice suprapuse peste hartă.

Metodele sunt similare cu cele din FriendsList, întrucât diferă doar modul de transmitere a datelor către utilizator.



3. Concluzii şi dezvoltări ulterioare

Având în vedere creşterea continuă a numărului de utilizatori Android, am gândit această aplicaţie pentru a uşura modul de a lua legătura cu prieteni ce folosesc aceeaşi tehnologie. Avantajele acestui soft:

- Usor de folosit
- Comod
- Eficient
- Furnizează date în timp real
- Grad mare de securitate
- Open-source

Aplicaţia constă într-o listă în care fiecare element reprezintă o instanţă a clasei Friend (nume, distanţă, status, imagine) şi o hartă.

Lista este creată şi afişată de la pornirea aplicaţiei şi realizează un update la un anumit interval de timp scurs sau la o anumită distanţă parcursă de catre una dintre persoanele din listă.

Harta arată poziţia utilizatorului, precum şi cea a prietenilor aflaţi in apropiere.

Trecerea dintre cele doua activităţi se face printr-o comandă din meniu.

Datele persoanelor din listă sunt preluate dintr-un server. De asemenea fiecare are posibilitatea de a alege să nu fie localizat, neîncalcându-se astfel dreptul la intimitate (se rezolvă problemele de securitate).

Cautăm încă cel mai eficient mod de preluare a datelor din server.

Bibliografie

[1] Mark MURPHY, *Beginning Android,* Apress, 2009
[2] Android Dveloper's Guide
http://developer.android.com/guide/index.html
[3] Android Reference
http://developer.android.com/reference/packages.html
[4] Cursul de Programarea Dispozitivelor Mobile din cadrul FILS, UPB

MONICA STOICESCU
Universitatea Politehnica Bucuresti
Facultatea de Inginerie in Limbi Straine
Splaiul Independenţei 313 Bucuresti
ROMANIA
monica.stoicescu00@gmail.com

Sesiunea Națională de Comunicări Științifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Maya number converter

Zoltan Tamasi, Zsigmond-Attila Szasz
Coordonator: Lector Dr. Anca Vasilescu

Abstract

The Maya Converter is a Java application that transforms the decimal mathematical system into an vigesimal system (each placeholder has a possible twenty digits [0 - 19]), system used by the Maya civilisation to perform calculations. The application also calculates the sum of two numbers, explaining in detail the operation. The numbers can be entered from the keyboard or from the buttons on the applications interface trough event listeners, and the detailed operations and the result are displayed on the panels adequate for each operation. This way, the application explain how the Maya civilization developed their culture.

Keywords: Maya, Java, converter, mathematical system, GUI

1 Introducere

1.1 Scurt istoric al matematicii si al metodelor de calcul

Istoria matematicii nu are un inceput clar definit, insa aparitia acesteia este strans legata de evolutia omului. Este posibil ca oamenii sa-si fi dezvoltat anumite abilitati matematice inca inainte de aparitia scrierii.

Din momentul in care omul a fost capabil sa foloseasca si sa inteleaga notiuni abstracte, dar si datorita dezvoltarii relatiilor interumane si intertribale si, nu in ultimul rand, a primelor sisteme de scris (insemnarile facute pe peretii pesterilor sub forma unor imagini care exprimau, atat trairi in taramul real, dar si in cel oniric si, din ce in ce mai mult, pe taramul ideilor), a aparut nevoia de „numar”.

Numarul este una dintre cele mai simple notiuni abstracte; este abstracta deoarece un numar nu poate fi relevat de un obiect material; exista numai semne conventionale care il exprima. Relatiile comerciale s-au dezvoltat odata cu evolutia spiritului uman; in acelasi timp, numarul a inceput sa fie din ce in ce mai prezent in viata oamenilor si, in cele din urma, indispensabil unei existente umane asa cum am inceput s-o constientizam ca omenire in urma cu 5.000 de ani, de cand dateaza urmele primelor state care au aparut in lume.

De asemenea, au aparut operatiile: adunarea, scaderea, inmultirea si, in cele din urma, impartirea, care a pus probleme oamenilor invatati pana in timpul Renasterii, cand s-a dezvoltat metoda moderna de impartire, numita metoda sahului, deoarece a fost inspirata de unele miscari pe tabla de sah. Unele din primele descoperiri matematice tin de extragerea radacinii patrate, a radacinii cubice, rezolvarea unor ecuatii polinomiale, trigonometrie, fractii, artimentica numerelor naturale, etc. Acestea au aparut

in cadrul civilizatiilor akkadiene, babyloniene, egiptene, chineze, maiase si civilizatiile de pe valea Indului.

1.2 Civilizatia Maya

Trimis din Teotihuacan, marele oras de la vest, in anul 378 d.Hr., comandantul de trupe S-a Nascut Foc a fondat noi dinastii, care au adus o splendoare fara pereche lumii maya.

Maya au fost intotdeauna o enigma. Cu cateva decenii in urma, gloria oraselor lor ruinate, precum si scrierea lor frumoasa, dar nedescifrata facusera ca multi cercetatori sa-si imagineze o societate pasnica de scribi si preoti. Cand epigrafistii au reusit, in cele din urma, sa descifreze basoreliefurile maya, a reiesit o imagine mai intunecata, a unor dinastii razboinice, a rivalitatilor intre curtile regale, a unor palate arse din temelii. Istoria maya a devenit o fresca cu date precise si personaje puternic conturate.

Dar au ramas mistere profunde, printre care ce anume a determinat saltul final al civilizatiei maya catre maretie? Cam in perioada in care renumele lui S-a Nascut Foc lua proportii, un val de schimbari a traversat lumea maya. Ceea ce fusese o adunatura de orase izolation.

Similar cu alte civilizatii mesoamericane, maiasii au folosit sistemul de numerotare in baza 20 (vigesimal) si in baza 5. De asemenea, maiasii preclasici si vecinii lor au dezvoltat in mod independent conceptul de numar zero undeva in jurul anului 36 i.Hr..

Inscriptiile arata ca maiasii lucrau cu numere de pana la sute de milioane si date atat de mari incat era nevoie de mai multe linii doar pentru a le reprezenta. Ei au facut observatii astronomice extrem de precise; calculele lor privind miscarile Lunii si planetelor sunt egale sau superioare oricarei altei civilizatii care a folosit metoda de observare cu ochiul liber.

1.3 Scopul lucrarii

Aceasta lucrare se doreste a fi un instrument didactic prin ilustrarea unei metode de calcul folosita in epoca veche.

Programul ne poate ajuta sa intelegem modul de gandire a maiasilor, dar si sa comparam operatia de adunare folosita de maiasi si adunarea obisnuita in baza 10.

2 Realizarea programului

2.1Limbajul de programare

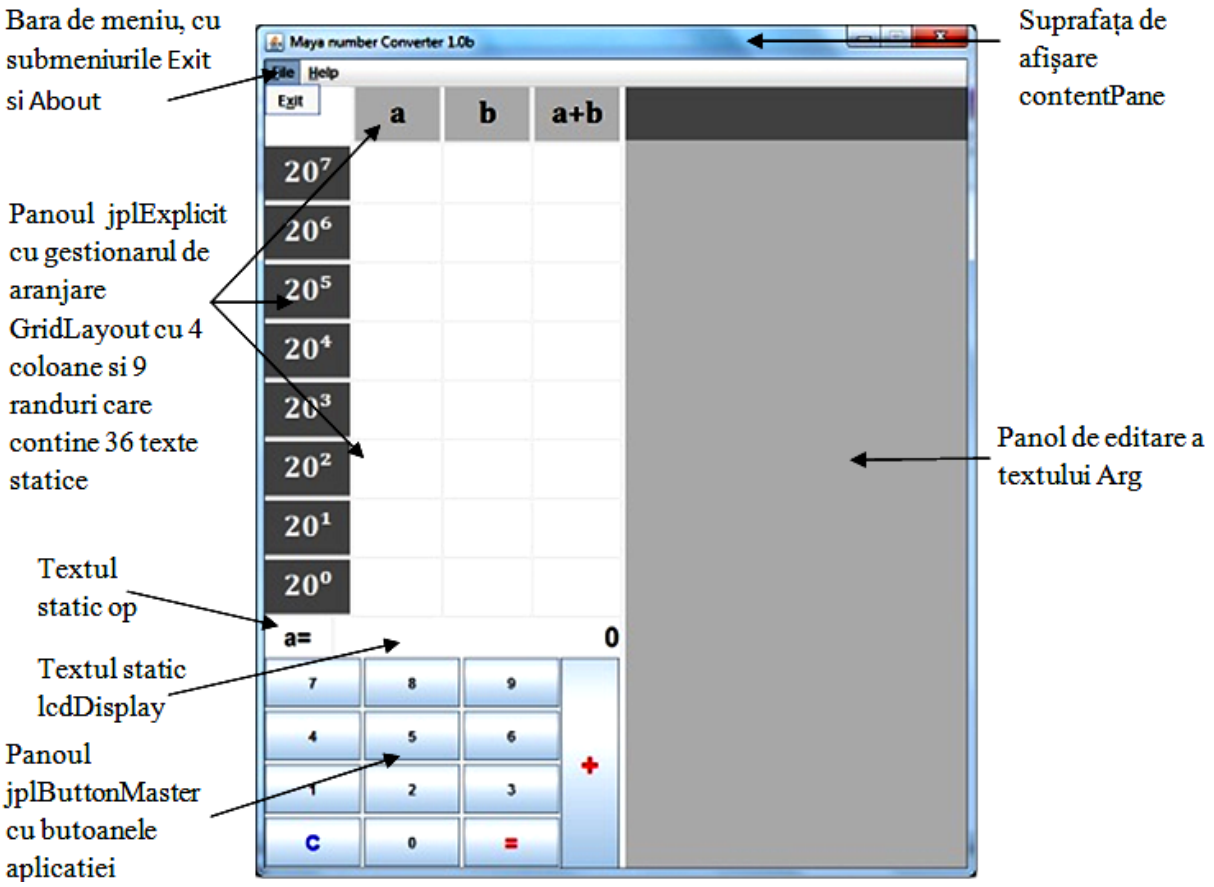
Pentru realizarea programului am folosit limbajul de programare Java, deoarece este o tehnologie care se aseamana cu limbajele C, C++ si Pascal, insa ofera posibilitatea crearii unor aplicatii mai complexe. Java este in acelasi timp un limbaj obiect orientat si unul care ofera suport pentru interfetele grafice. Totodata este un limbaj modern, gratuit, si open source, folosit des pentru crearea aplicatii desktop.

Ca si editor Java am ales Eclipse, pentru ca este usor de folosit chiar si de catre un programator fara experienta in programarea Java, ajutand mult la interpretarea erorilor si la folosirea corecta a functiilor si claselor Java.

2.2 Elementele vizuale

Pentru crearea interfatei vizuale am folosit componentele grafice Java apartinand bibliotecilor de clase *AWT* si *Swing.* Suprafata de afisare (containerul), meniul, panourile, butoanele si textele statice

(labelurile) al aplicatiei sunt din biblioteca Swing iar gestionarele de pozitionare (layout manager) sunt din biblioteca AWT. Figura de mai jos ilustreaza concret componentele folosite:



2.3 Algoritmul programului

Algoritmul programului realizeaza conversia unui numar zecimal (baza 10) intr-un numar vigesimal (baza 20), atasarea numarului rezultat simboluri folosite de maiasi, respectiv afisarea simbolurilor pe suprafete coraspunzatoare.

Astfel functiile fiecarei sectiuni ai algoritmului arata in felul urmator:

Conversia:

```
void numar(int x,int i,int k)
      {
       if((k==0)&&(x==0))
                    num[i][0]=0;
       else if(x!=0)
                   {
                    num[i][k]=x%20;
                    numar(x/20,i,k+1);
                   }
      }
```

Adunarea:

```
      int[] transport=new int[10];
```

```
void tr0()
{
 int i;
 for(i=0;i<10;i++)
        transport[i]=0;
}

void adunare(int k)
{
 if(k==0)
        tr0();
 if((num[0][k]>=0)||(num[1][k]>=0))
        {if((num[0][k]>=0)&&(num[1][k]>=0))
                {
                 num[2][k]=(num[0][k]+num[1][k]+transport[k])%20;
                 transport[k+1]=(num[0][k]+num[1][k]+transport[k])/20;
                 adunare(k+1);
                }
        else
                {
                 num[2][k]=num[0][k]+num[1][k]+transport[k]+1;
                 adunare(k+1);
                }}
 else
         if(transport[k]>0)
        {
         num[2][k]=num[0][k]+num[1][k]+transport[k]+2;
         adunare(k+1);
        }
}
```

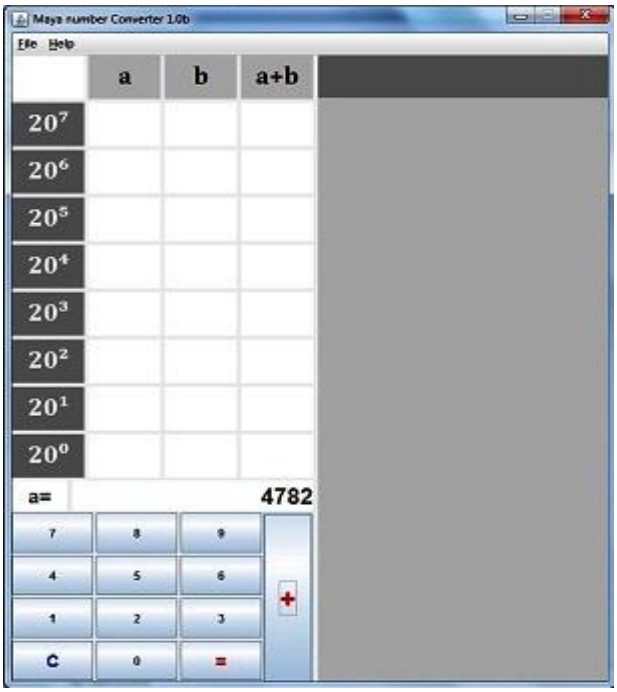
Afisarea:

```
ImageIcon desen(int i)
        {ImageIcon q;
         switch(i)
           {case 0: q=zero; break;
                case 1: q=unu; break;
                case 2: q=doi; break;
                case 3: q=trei; break;
                case 4: q=patru; break;
                case 5: q=cinci; break;
                case 6: q=sase; break;
                case 7: q=sapte; break;
                case 8: q=opt; break;
                case 9: q=noua; break;
                case 10: q=zece; break;
                case 11: q=unspe; break;
                case 12: q=doispe; break;
                case 13: q=treispe; break;
                case 14: q=paispe; break;
                case 15: q=cinspe; break;
                case 16: q=saispe; break;
                case 17: q=saptispe; break;
                case 18: q=optispe; break;
                case 19: q=nouaspe; break;
                default: q=zero;
                }
         return(q);
        }
```

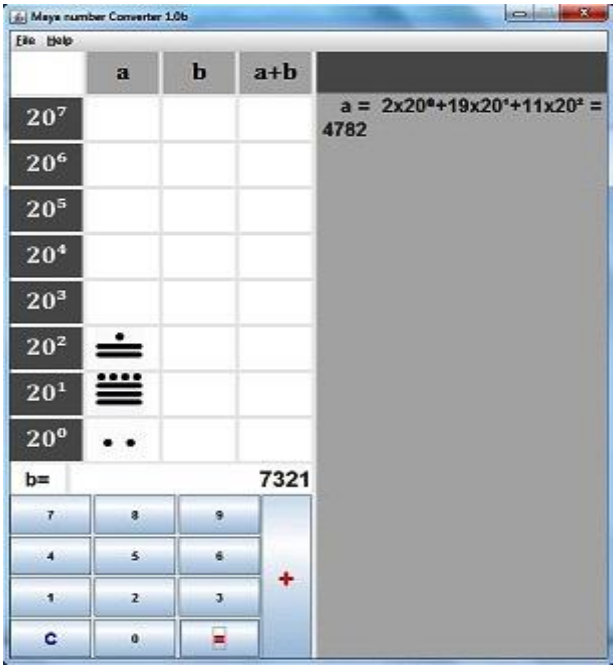
```
JLabel celula(int i,int j)
	{
	 JLabel q=new JLabel(" ", JLabel.CENTER);
	 if(i==0)
		switch(j)
			{
			 case 0: q=niv01; break;
			 case 1: q=niv11; break;
			 case 2: q=niv21; break;
			 case 3: q=niv31; break;
			 case 4: q=niv41; break;
			 case 5: q=niv51; break;
			 case 6: q=niv61; break;
			 case 7: q=niv71; break;

			}
	 if(i==1)
		switch(j)
			{
			 case 0: q=niv02; break;
			 case 1: q=niv12; break;
			 case 2: q=niv22; break;
			 case 3: q=niv32; break;
			 case 4: q=niv42; break;
			 case 5: q=niv52; break;
			 case 6: q=niv62; break;
			 case 7: q=niv72; break;

			}
	 if(i==2)
		switch(j)
			{
			 case 0: q=niv03; break;
			 case 1: q=niv13; break;
			 case 2: q=niv23; break;
			 case 3: q=niv33; break;
			 case 4: q=niv43; break;
			 case 5: q=niv53; break;
			 case 6: q=niv63; break;
			 case 7: q=niv73; break;

			}
	 return(q);
	}


void afis(int i)
	{
	int j;
	for(j=0;j<8;j++)
		if(num[i][j]>=0)
			celula(i,j).setIcon(desen(num[i][j]));
	}
```

2.4 Utilizarea aplicatiei

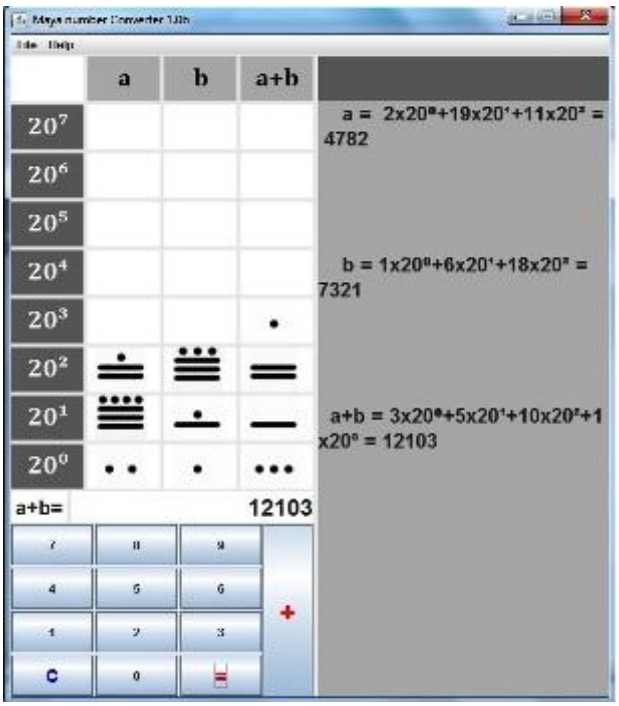
Prima data introducem numarul intreg a:



Apasam butonul „+”, numarul a este convertit in numar maya si afisat pe panoul din stanga, iar explicit pe panoul din dreapta:



Apasam ENTER sau butonul „=”, iar rezultatul adunarii impreuna cu numarul b apar si pe panoul de explicatie (dreapta), si pe panoul maya (stanga):



3 Concluzie

Acest program se poate dezvolta sa ilustreze nu numai simbolurile si modul de calcul al civilizatiei maiase ci si ai altor civilizatii in acelasi timp, ajutand astfel sa comparam mai multe civilizatii din punct de vedere al utilizarii notiunilor „numar” respectiv „adunare”.

4 Bibliografie

[1] E. Ciurcea si C. Luca *Algoritmi si Programare Java.* Editura Albastra, 2008
[2] Roger Cadenhead *Teach Yourself Java 2 in 24 hours.* Sams Publishing, Editia a doua, 2001
[3] http://java.sun.com/docs/books/tutorial/
[4] http://java.sun.com/
[5] http://www.eclipse.org/
[6] http://www.wikipedia.com/

TAMASI ZOLTAN
Universitatea Transilvania Brasov
Facultatea de Matematica-Informatica
Specializarea: Informatica
Iuliu Maniu 50, Brasov
ROMANIA
zoltan.tamasi@xu.unitbv.ro

SZASZ ZSIGMOND-ATTILA
Universitatea Transilvania Brasov
Facultatea de Matematica-Informatica
Specializarea: Informatica
Iuliu Maniu 50, Brasov
ROMANIA
zsigmond.szasz@xu.unitbv.ro

Sesiunea Naţională de Comunicări Ştiinţifice a Studenţilor
"Imaginaţie, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Sistem de recomandări bazat pe extragerea regulilor de asociere

Claudia Vîrnă
Îndrumător Lect. Dr. Lucian Sasu

Abstract

Thanks to their ability in improving sales, recommender systems have become very popular nowadays. As a result many efforts are made to improve the performance of existing algorithms. One of the most used techniques in this area is the collaborative filtering method, which is based on the idea that a user receives personalized recommendations according to his or hers previously expressed preferences and to the preferences of similar users. In this article, we propose a new approach to an association rule mining algorithm, which belongs to the class of collaborative filtering techniques: the popular Apriori algorithm. The main drawback of the original Apriori algorithm is the fact that the user must specify two critical values that have a major impact on the evolution of the algorithm: the minimum support and the minimum confidence. Consequently, in this paper we provide a solution to this problem. Moreover, we suggest a new method that extends the initial usage of the algorithm, by making predictions of the ratings a user may give. One of the benefits of this technique is the improvement of the algorithm's evaluation. The experimental results show that the new algorithm gives better results than similar approaches.

Cuvinte cheie: sisteme de recomandări, reguli de asociere, suport minim, rating

1 Introducere

Într-o epocă a vitezei, precum cea în care trăim, este imposibil să nu fim zilnic copleşiţi de numărul mare de decizii ce trebuiesc luate. Astfel, şi deciziile cele mai banale devin dificile dacă avem în vedere cantitatea mare de date ce ne bombardează permanent. Prin urmare, orice mic ajutor este bine venit. Aici intervin sistemele de recomandări, care ne fac viaţa mult mai uşoară atunci când vine vorba despre alegerea unei opţiuni dintre mii de astfel de variante. Un exemplu în acest sens ar fi chiar cumpărarea unei cărţi de pe un site specializat ca şi amazon.com, unde există nenumărate posibilităţi de selecţie. Interesul larg arătat domeniului recomandărilor este dovedit şi de relativ recenta competiţie Netflix Prize [1], care a avut ca scop determinarea celui mai potrivit algoritm pentru a prezice rating-uri (note) de filme, dându–se rating-uri exprimate de către alţi clienţi. Scopul a fost obţinerea unui algoritm care să întreacă cu 10% algoritmul Cinematch al companiei Netflix. Premiul acordat reflectă fără dubiu gradul de interes economic: 1 milion de dolari.

Datorită procesului rapid de dezvoltare a tehnologiei, aceste sisteme de recomandări au cunoscut şi ele o evoluţie semnificativă de-a lungul ultimilor ani, extinzându-se trei tipuri majore de astfel de sisteme[2]:

1. Sisteme bazate pe conţinut (content-based systems)
2. Sisteme bazate pe cunoştinţe (knowledge-based systems)
3. Sisteme bazate pe inteligenţa colectivă / filtrare colaborativă (collaborative filtering systems)

Sistemele de recomandări bazate pe conţinut stabilesc un profil al utilizatorului analizând anumite caracteristici ale articolelor preferate de acesta. În mod similar, sistemele bazate pe cunoştinţe iau în considerare date referitoare atât la utilizator, cât şi la produsele achiziţionate de acesta, oferind, pe baza unei analize, anumite recomandări.

Spre deosebire de cele două situaţii prezentate anterior, sistemele bazate pe inteligenţa colectivă folosesc doar lista preferinţelor utilizatorilor. Prin urmare, au ca şi principal avantaj faptul că nu necesită informaţii suplimentare, ce reprezintă un aspect critic pentru primele două abordări. Astfel, prin necesitatea de a obţine numeroase informaţii, primele două tipuri de sisteme de recomandări cer utilizatorului completarea anumitor date, a căror corectitudine este imposibil de verificat. În plus, ignorând informaţiile referitoare la caracteristicile articolelor, recomandările furnizate ţin cont mai mult de gusturile utilizatorului, decât de dependenţele dintre obiecte. Acest lucru înseamnă că folosind astfel de sisteme se pot recomanda articole care să nu fie similare, dar care totuşi să fie pe placul utilizatorului, ceea ce nu se intâmplă în cazul sistemelor precizate anterior [3].

În această lucrare atenţia cade asupra sistemelor de recomandări bazate pe inteligenţa colectivă. În continuare vom face o scurtă prezentare a celor mai folosite tehnici din acest domeniu. Apoi va fi descris un algoritmul care a fost implementat într-o aplicaţie web ce permite unui utilizator care a dat anterior nişte rating-uri să vizualizeze alte filme, în ordinea descrescătoare a rating-urilor estimate de către algoritm.

1.1 Rezultate similare

1.1.1 k-Nearest Neighbor (k-NN)

Această metodă este des întâlnită în sistemele de recomandări de tipul CF, constând din trei paşi de baza: calculul similarităţilor dintre utilizatori, selecţia "vecinilor" şi calculul predicţiei.

Determinarea similarităţilor necesită parcurgerea întregii baze de date şi compararea utilizatorului pentru care se doreşte a se oferi recomandări cu toţi ceilalţi utilizatori. Similarităţile dintre aceştia se obţin din rating-urile pe care utilizatorii le dau pentru articolele comune, folosind metrici precum corelaţia Pearson sau distanţa Euclidiană [4], [5].

În funcţie de tipul de problemă şi de cantitatea de date se preferă să se folosească filtrarea datelor după articolele favorite ale utilizatorilor în locul filtrării după utilizatori [4].

Deşi algoritmul este simplu din punct de vedere conceptual, există câteva dezavantaje, cel mai important fiind legat chiar de formula prin care se realizează predicţiile. Astfel, deşi doi utilizatori au preferinţe similare, acestea pot să nu fie corelate. În plus, se cunoaşte faptul că metodele locale suferă de "blestemul dimensionalităţii", adică necesarul de date pentru construirea unor estimatori creşte exponential cu numărul de dimensiuni [6], [7]. Problemele ce apar frecvent în momentul utilizării acestei metode sunt tratate pe larg in [8].

1.1.2 Extragerea regulilor de asociere (ARM - Association Rule Mining)

Extragerea regulilor de asociere este o abordare care are ca şi scop descoperirea unor relaţii interesante între articole, prin găsirea acelor articole care apar frecvent în aceleaşi tranzacţii. Aplicaţi sistemelor de recomandări, algoritmii de tip ARM pot genera recomandări după ce aceste reguli între articole sau între utilizatori au fost descoperite. În literatura de specialitate există numeroşi algoritmi de extragere a regulilor de asociere, însă numai o parte pot fi utilizaţi în cadrul sistemelor de recomandări. Printre aceştia se află şi algoritmul Apriori şi numeroasele sale versiuni, cum ar fi MAR [9], ASARM [10] şi FARAMS [11].

În anul 2003, Kim a propus o nouă variantă a algoritmului Apriori, şi anume algoritmul MAR (Multi-level Asociation Rules) [9], care aplică metoda de extragere a regulilor de asociere, atât asupra articolelor, cât şi asupra categoriilor din care acestea fac parte.

Algoritmul prezentat de Alvarez si Lin, ASARM (Adaptive Support Association Rule Mining) [10], găseşte pe rând regulile pentru fiecare articol şi ajustează suportul minim pe baza unui număr de reguli specificat de utilizator.

O altă versiune a lui Apriori este algoritmul FARAMS [11], care combină cele două metode prezentate anterior, folosind în plus şi logica fuzzy pentru determinarea regulilor.

1.2 Elemente de originalitate

Scopul acestei lucrări este de a prezenta o versiune îmbunătăţită a algoritmului Apriori, algoritm construit pornind de la paradigma regulilor de asociere, introdusă de Agrawal [12].

Principalele caracteristici ale metodei propuse sunt următoarele:

1. Modelul de extragere a regulilor de asociere are la bază două valori prag esenţiale pentru evaluarea asocierilor exprimate prin intermediul regulilor: *gradul de încredere*, care măsoară nivelul de corelare dintre articolele prezente în cadrul regulii şi *suportul*, care determină importanţa unei astfel de corelări. Găsirea regulilor de asociere se reduce la găsirea regulilor care au suportul şi gradul de încredere mai mari decât două constante: *minsup* şi respectiv *minconf*. Majoritatea algoritmilor ce implementează extragerea regulilor de asociere lasă utilizatorului sarcina de a fixa valorile pentru *minconf* şi *minsup*. Însă acest lucru este greu de realizat deoarece prin specificarea unei valori mari pentru suportul minim se evită explozia numărului de reguli obţinute, dar se pot pierde reguli interesante care au suportul mai mic. În cele ce urmează se va prezenta o metodă prin care suportul minim va fi specificat automat, fără intervenţia utilizatorului.

2. Algoritmul Apriori, de la care s-a plecat, nu permite predicţia unui rating pentru articolele recomandate utilizatorilor, fapt pentru care nu se poate face o comparare corectă a rezultatelor obţinute. Din această cauză, am considerat necesară introducerea unei astfel de predicţii, care este folosită şi pentru micşorarea numărului de reguli rezultate. Metoda sugerată are la bază determinarea regulilor pentru care antecedentul se găseşte în lista rating-urilor utilizatorului. Pornind de la aceste reguli se obţin valori pentru rating-urile articolelor recomandate folosind regresia liniară cu metoda celor mai mici pătrate şi rating-urile date de utilizator pentru articolele din antecedentul regulilor.

3. Algoritmul Apriori, in varianta originală, este prohibitiv din punct de vedere computaţional, deoarece poate ajunge sa genereze reguli cu un număr mare de elemente, reguli care în cazul sistemelor de recomandări nu îşi au rostul. De aceea prin limitarea numărului de elemente ale unei reguli se reduce complexitatea algoritmului, rezultatele obţinute fiind suficiente pentru a recomanda utilizatorilor articolele potrivite.

1.3 Structura lucrării

Lucrarea este împărţită astfel: în secţiunea 2 este introdusă problema extragerii regulilor de asociere, fiind prezentate noţiunile referitoare la această problemă şi paşii urmaţi în cadrul algoritmului Apriori clasic, apoi in secţiunea 3 este prezentat algoritmul modificat — rezultatul principal al acestei lucrări — cu precizările de rigoare. În secţiunea 4 sunt prezentate rezultatele experimentale obţinute prin compararea algoritmului introdus cu alţii similari, iar in secţiunea 5 sunt prezentate concluziile.

2 Extragerea regulilor de asociere

2.1 Noţiuni specifice

În această secţiune sunt prezentate noţiunile care stau la baza construcţiei algoritmului Apriori. O prezentare detaliată poate fi găsită în [12], [13].

Fie I o mulţime de articole şi D o mulţime de tranzacţii, în care fiecare tranzacţie t reprezintă o submulţime de articole $t \subseteq I$. O tranzacţie t conţine mulţimea de articole Xdin I dacă $X \subset t$. O regulă de asociere este o implicaţie de forma $X \Rightarrow Y$, unde $X \subset I$, $Y \subset I$ şi $X \cap Y = \emptyset$. În acest caz X se numeşte antecedentul, iar Y este consecventul regulii.

Suportul unei reguli de această formă reprezintă procentul de tranzacţii din D care conţin atât X cât şi Y, în timp ce gradul de încredere este definit ca şi procentul de tranzacţii ce îl conţin pe Y, dintre tranzacţiile care îl conţin pe X.

Problema tradiţională de extragere a regulilor de asociere este definită astfel: fiind dată o mulţime de tranzacţii şi valori pentru suportul minim şi gradul de încredere minim, să se găsească toate regulile de asociere pentru care suportul şi gradul de încredere depăşesc valorile specificate de utilizator.

Pornind de la această definiţie algoritmul Apriori parcurge următorii paşi:

1. Se găsesc toate combinaţiile de articole pentru care suportul depăşeşte valoarea specificată de utilizator, folosindu-se proprietatea conform căreia dacă o mulţime de articole Y este frecventă, atunci şi toate submulţimile sale sunt tot frecvente.

2. Se generează reguli pe baza mulţimilor frecvente obţinute la pasul anterior şi se păstrează regulile pentru care constrângerea referitoare la gradul de încredere este respectată.

Cu toate că modul de definire al problemei este potrivit pentru analiza coşului de cumpărături [14], în cazul sistemelor de recomandări nu se poate spune acelaşi lucru.

2.2 Eliminarea pragului minsup

Una dintre problemele majore ale algoritmului Apriori este, după cum specifică Lin şi Alvarez în [10], faptul că utilizatorul trebuie să precizeze anterior valori pentru *minsup* şi *minconf*, valori care depind exclusiv de setul de date avut la dispoziţie. Astfel, se pot obţine fie prea multe, fie prea puţine reguli, singura cale de a ajunge la rezultate potrivite fiind cea empirică, bazată pe încercări repetate cu diferite valori ale acestor parametri.

O posibilă soluţie a acestei probleme a fost oferită de către Leung, Chan şi Chung în [11], în care se sugerează ajustarea dinamică a valorii suportului minim, plecând de la o altă constantă furnizată de utilizator, şi anume numărul de reguli rezultate.

Însă şi de data aceasta rămâne la latitudinea utilizatorului să stabilească valoarea potrivită pentru această constantă, ceea ce nu este foarte uşor de realizat, mai ales în cazul sistemelor de recomandări.

O altă variantă este cea prezentată de Wen-Yang Lin în [15], în care extragerea regulilor se face fără a fi nevoie ca utilizatorul să specifice valoarea pentru *minsup*. Pentru a realiza acest lucru se fixează un alt prag, care se obţine direct din setul de date şi care porneşte de la noţiunea de suport minim per articol. Vom prezenta această abordare întrucât algoritmul pe care îl vom introduce foloseşte noţiunile ilustrate în [15]. Trebuie totuşi punctat faptul că algoritmul original din [15] nu oferă rezultate suficient de bune în cazul sistemelor de recomandări.

Pentru a elimina pragul *minsup*, în locul căutării mulţimilor de articole care satisfac constângerea specifică suportului, se introduce o nouă constrângere referitoare la *diferenţa de nivel*, despre care vom vorbi în continuare. Prin urmare, apar următoarele definiţii şi notaţii preluate din [15]:

Definitie 1 *Fie $ms(a)$ suportul minim al unui articol a din I. O mulţime de articole $A = \{a_1, \ldots, a_k\}$, cu $a_i \in I$, este frecventă dacă suportul lui A verifică urmatoarea condiţie:*

$$sup(A) \geq \min_{a_i \in A} sup(a_i)$$

Definitie 2 *O regulă de asociere de forma $A \Rightarrow B$ este puternică dacă*

$$sup(A \Rightarrow B) \geq \min_{a_i \in A \cup B} sup(a_i)$$

si

$$conf(A \Rightarrow B) \geq minconf$$

După cum am anticipat se utilizează în definirea noii constrângeri noţiunea de diferenţa de nivel, prezentă în următoarea definiţie:

Definitie 3 *O regulă de asociere $A \Rightarrow B$ este interesantă dacă*

$$lift(A \Rightarrow B) \geq 1$$

unde

$$lift(A \Rightarrow B) = \frac{sup(A \cup B)}{sup(A)sup(B)}$$

reprezintă diferenţa de nivel.

Ideea principală a acestei abordări este de a utiliza gradul de încredere şi diferenţa de nivel în cadrul constrângerii referitoare la suportul minim pentru a elimina mulţimile de articole frecvente, care nu generează reguli interesante. În acest fel precizarea pragului *minsup* nu mai este necesară.

Pornind de la aceste definiţii, în [15] se demonstrează următoarele rezultate, care au ca şi scop găsirea unei metode de specificare a noii constrângeri:

Lema 1 *Fie $A \cup B$ o mulţime frecventă astfel încât $A \cap B = \emptyset$. Se consideră, fără a se pierde din generalitate, că $a \in A$ este articolul pentru care are loc următoarea relaţie:*

$$sup(a) = \min_{a_i \in A \cup B} sup(a_i)$$

În aceste condiţii, regula de asociere $A \Rightarrow B$ este puternică dacă $ms(a) = sup(a) \times minconf$.

Lema 2 *Fie I o mulţime de articole. Dacă suportul minim al fiecărui articol este definit astfel:*

$$ms(a_i) = sup(a_i) \times \max_{a_j \in I - \{a_i\}} sup(a_j)$$

atunci orice regulă de asociere puternică $A \Rightarrow B$, cu $A, B \subset I$ şi $A \cap B = \emptyset$ este interesantă, adică verifică în plus şi condiţia:

$$\frac{sup(A \cup B)}{sup(A)sup(B)} \geq 1$$

Aşadar, în prima lemă se enunţă o modalitate de alegere a suportului minim per articol, modalitate ce asigură eliminarea mulţimilor frecvente ce nu generează reguli puternice. Mai departe, cea de-a doua lemă ilustrează o metodă prin care se găsesc regulile interesante. Pentru a implementa rezultatele anterioare mulţimea iniţială de articole trebuie să fie sortată crescător după suport.

Prin urmare, constrângerea de tip CLS (confidence-lift support constraint) se defineşte astfel:

$$ms(a_i) = \begin{cases} sup(a_i) \times \max\{minconf, sup(a_{i+1})\}, & \text{dacă } 1 \leq i \leq n-1 \\ sup(a_i), & \text{dacă } i = n \end{cases} \quad (1)$$

unde n reprezintă numărul de articole.

Conform celor menţionate mai sus procesul de extragere a regulilor de asociere se modifică după cum urmează [15]:

1. Se parcurge setul de tranzacţii D pentru a obţine suportul fiecărui articol şi se calculează suportul minim conform ecuaţiei 1. Se creează lista $L1$ reprezentând lista tuturor articolelor sortate crescător în funcţie de suportul minim.

2. Se generează candidaţii de ordin 2 conform metodei apriori-gen specifică algoritmului Apriori original, iar pentru candidaţii de ordin mai mare se procedează astfel:

   (a) se aplică metoda apriori-gen

   (b) se elimină candidaţii ce conţin submulţimi pentru care articolul cu suportul minim coincide sau are acelaşi suport minim cu cel al candidatului, însă nu se află în lista anterioară de mulţimi frecvente

   Trebuie specificat faptul că fiecare candidat reprezintă o listă de articole sortată crescător în funcţie de suportul minim.

3. Dintre candidaţii obţinuti se selectează doar cei care verifică următoarea relaţie:

$$sup(A) \geq ms(A[1])$$

unde A reprezintă canditatul, iar $A[1]$ este primul articol al candidatului A.

Prin urmare, se selectează doar candidaţii al căror suport este mai mare decât suportul minim al primului articol al candidatului. Acest lucru rezultă din definiţia suportului minim per articol şi din lemele prezentate anterior.

4. După obţinerea celor mai frecvente mulţimi de articole se generează regulile de asociere care respectă constrângerea de încredere.

3 O nouă abordare a algoritmului Apriori

3.1 Modificarea algoritmului Apriori pentru generarea de rating-uri

În ceea ce urmează vom prezenta o nouă modalitate de a furniza recomandări, pornind de la algoritmul Apriori.

În primul rând, pentru a putea face acest lucru este important să se fixeze modul de alegere a tranzacţiilor, deoarece pe baza acestora se vor obţine şi regulile de asociere. În cazul recomandărilor, se presupune faptul că fiecare utilizator a asociat anumitor articole nişte rating-uri. În această situaţie există două modalităţi de a oferi recomandări utilizatorilor:

1. Se generează reguli ce evidenţiază asocierile dintre utilizatori
2. Se generează reguli pentru asocierile dintre articole

Însă, având în vedere faptul că probabilitatea evenimentului de apariţie a unui nou utilizator este mai mare decât cea a introducerii unui nou articol se preferă cea de-a doua variantă. În acest fel execuţia algoritmului nu trebuie repetată foarte des.

În general, o tranzacţie conţine lista de articole pentru care un utilizator a precizat rating-uri. Pentru a facilita metoda de calcul a suportului, vom considera, în abordarea propusă, că fiecare articol are ataşată lista de utilizatori care au precizat rating-uri pentru articolul respectiv. În acest fel, nu mai trebuie parcurs tot setul de date pentru a calcula suportul unei mulţimi, ci este suficient să se ia în considerare doar articolele, din care este formată mulţimea.

Având stabilit punctul de pornire al algoritmului, putem trece mai departe la paşii efectivi de obţinere a recomandărilor.

După cum am precizat anterior, algoritmul introdus în această lucrare se bazează pe modelul prezentat în [15]. Cu toate că versiunea algoritmului Apriori ilustrată de Wen-Yang Lin oferă posibilitatea de a reduce numărul de constante furnizate de utilizator, se obţin mult prea multe reguli, care utilizate in procesul de recomandare micşorează performanţa acestuia.

Din această cauză propunem înlocuirea pasului 3 cu următorul:

3'. Dintre candidaţii obţinuţi se selectează doar cei care au suportul mai mare decât media suporturilor minime ale articolelor din care sunt formaţi.

Se poate observa cu uşurinţă, faptul că mărind pragul de selecţie a celor mai frecvente mulţimi, numărul de reguli se micşorează, iar rezultatele precizate anterior îşi păstrează veridicitatea. Acest lucru se întâmplă datorită faptului că, prin creşterea valorii pragului, regulile obţinute rămân tot interesante.

În plus, vom considera doar regulile care au partea de consecvent formată dintr-un singur articol. În acest fel se reduce timpul de execuţie al algoritmului şi nu există confuzii în momentul realizării recomandărilor.

Un alt element de noutate prezentat în această lucrare este partea de predicţie a rating-urilor pentru articolele recomandate. Pentru a realiza acest lucru am folosit regresia liniară cu metoda celor mai mici pătrate astfel:

Pentru fiecare regulă rezultată în urma execuţiei algoritmului Apriori modificat, se parcurg tranzacţiile şi se selectează doar cele ce conţin articolele regulii, atât cele din antecedent, cât şi cele din consecvent. Pornind de la aceste tranzacţii se foloseşte regresia liniară cu metoda celor mai mici pătrate ([6]) pentru a se obţine o dreaptă de regresie corespunzătoare fiecărei reguli. Regulile pentru care nu există o astfel de dreaptă sunt eliminate. Apoi, rezultatele obţinute sunt utilizate pentru predicţia rating-ului articolului din consecventul regulii. În realizarea predicţiei se foloseşte rating-ul dat de utilizatorul curent pentru articolul din antecedent.

În continuare vom prezenta un exemplu prin care se ilustrează modul de predicţie a rating-urilor.

Exemplu 1 *Presupunem că în urma aplicării algoritmului, printre regulile rezultate se află şi următoarele:*

$$a \Rightarrow c \qquad d \Rightarrow e \tag{2}$$

şi dorim să furnizăm recomandări unui utilizator care a precizat rating-urile:

$$a \text{ cu rating-ul } 5, \qquad b \text{ cu rating-ul } 3, \qquad d \text{ cu rating-ul } 4 \tag{3}$$

După cum se poate observa, atât prima cât şi a doua regulă au antecedentul printre articolele pentru care utilizatorul a dat rating-uri. Prin urmare, rămâne să estimăm rating-urile ce ar fi date de utilizator pentru filmele c şi e, care se află în consecventul celor două reguli. Pentru a putea face acest lucru trebuie parcurse toate tranzacţiile din baza de date şi găsite acele tranzacţii în care sunt date rating-uri pentru articolele din cele două reguli. Acest lucru se poate face uşor prin memorarea în cadrul fiecărei reguli a unei liste a utilizatorilor care au precizat rating-uri pentru toate articolele regulii. Procedând astfel parcurgerea întregii baze de date nu mai este necesară. Presupunem că în tranzacţiile obţinute utilizatorii au dat următoarele rating-uri:

$$\begin{array}{llll} a \text{ cu rating-ul } 5 & c \text{ cu rating-ul } 4, & a \text{ cu rating-ul } 3 & c \text{ cu rating-ul } 5 \\ a \text{ cu rating-ul } 5 & c \text{ cu rating-ul } 3, & d \text{ cu rating-ul } 4 & e \text{ cu rating-ul } 4 \\ d \text{ cu rating-ul } 3 & e \text{ cu rating-ul } 4, & d \text{ cu rating-ul } 2 & e \text{ cu rating-ul } 5 \\ d \text{ cu rating-ul } 3 & e \text{ cu rating-ul } 3 & & \end{array} \tag{4}$$

Aplicând regresia liniară cu metoda celor mai mici pătrate pentru

$$X = (5,3,5)^t \quad şi \quad Y = (4,5,3)^t \tag{5}$$

se obţine dreapta de regresie ce va fi utilizată pentru predicţia rating-urilor în cazul primei reguli. Valorile obţinute pe baza dreptei de regresie sunt rotunjite la cel mai apropiat intreg

Similar se procedează şi pentru a doua regulă pentru:

$$X = (4,3,2,3)^t \quad şi \quad Y = (4,4,5,3)^t \tag{6}$$

În final se obţine rating-ul 3.5 pentru c şi 3.5 pentru e.

În cazul în care pentru acelaşi utilizator există reguli care deşi au antecedente diferite au acelaşi articol în consecvent, pentru estimarea rating-ului articolului din consecvent se calculează media ponderată dintre rating-urile obţinute per regulă şi gradul de încredere al regulilor, dedus prin algoritmul Apriori.

După ce s-a realizat şi acest pas se trece printr-o etapă de validare, descrisă în secţiunea 3.2.

În final, la pasul 7 se parcurge lista utilizatorilor pentru care se doreşte a se furniza recomandări şi se caută regulile pentru care articolele din antecedent se găsesc printre articolele pentru care utilizatorii au precizat rating-uri. Odată găsite regulile corespunzătoare fiecărui utilizator se recomandă articolele aflate în consecventul lor.

Întregul proces prezentat anterior este schiţat în pseudocodul următor:

```
procedure C[k]-gen(L[k-1])
begin
    C[k]=apriori-gen(L[k-1]);
    for each itemset A=<a[1],a[2],...,a[k]> in C[k] do
```

```
       for each (k-1)-subset A'=<a[1'],a[2'],...,a[(k-1)']> of A do
          if a[1]=a[1'] or ms(a[1])=ms(a[1']) then
             if A' not in L[k-1] then delete A from C[k];
end;

procedure aprioriCLS(D,minconf)
begin
    compute ms(a) for all a in I;/* compute the minimum support for all items in I */
    L[1]=sort(I);/*sort all items according to their minimum support*/
    k=2;
    while(L[k] not null)
       begin
          if k=2 then C[2]=apriori-gen(L[1]);
                 else C[k]=C[k]-gen(L[k-1]);
          for each transaction t in D do
             begin
                C[t]=subset(C[k],t);
                for each candidate A in C[t] do
                   increase the count of A;
             end
          L[k]={A in C[k]| sup(A) is greater or equal to the average of
                              the item's minimum support};
          k++;
       end;
    return L=the union of all L[k];
end;

procedure generateRules(L,minconf)
begin
    R is an empty set f rules;
    for each frequent itemset l in L do
       for each nonemptysubset s of l do
          if sup(l)/sup(s)>minconf then
             we add the rule 's=>l-s' to R;
    return R;
end;
```

Prin intermediul procedurilor prezentate anterior se obţin regulile folosite în partea de recomandare. Pseudocodul asociat procesului de recomandare este dat mai jos.

```
procedure calculateRuleRegressionLine(r)
begin
   find all transactions containing the rule;
   use liniar regresion with least squares to obtain the separation line d;
   return d;
end;

procedure getRecommendations(user,articles,R)
begin
for each rule r in R do
  for each rated article item in articles do
     if(r.antecedent=item)
        begin
```

```
            d=calculateRuleRegressionLine(r);
            recommend item with rating d[r.antecedentRating];
        end;
end;
```

Trebuie specificat faptul că în pseudocodul prezentat nu se face referire la partea în care se obţin valorile erorilor medii pătratice pentru fiecare regulă, valori necesare eliminării regulilor care nu oferă rezultate suficient de bune.

3.2 Alegerea valorilor parametrilor prin validare

În problemele ce presupun determinarea unor modele este nevoie de precizarea de valori adecvate pentru parametrii acestora. Principalul obiectiv este de ajunge la parametri care sa dea cele mai bune rezultate pentru date noi. Performanţa pe setul de antrenare nu este un indicator bun pentru puterea de predicţie a unui astfel de model, datorită posibilităţii de a ajunge la overfitting pe setul de date. Dacă datele sunt în cantitate mare, atunci o metodă este de a folosi o parte din ele pentru a face antrenarea, iar pe un set independent se face evaluarea performanţei modelului. Se oferă astfel un criteriu statistic prin care se aleg cele mai potrivite valori pentru parametri.

Se pot considera următorii 2 parametri: *minconf* şi numărul de articole din antecedentul regulii. Eliminarea parametrului *minsup* din algoritmul Apriori original este motivată de rezultatele prezentate în secţiunea 2.2. Pentru al doilea parametru, în experimentele efectuate numărul de articole din antecedentul regulii a fost considerat pe rând 1 si 2. Diferenţele dintre rezultatele obţinute pe cele două cazuri nu au fost majore, fapt pentru care vom detalia doar situaţia şi rezultatele pentru cazul regulilor cu un singur element în antecedent.

Totodată, setul de validare poate fi folosit pentru a decide subsetul de reguli care se foloseşte efectiv pentru predicţie, astfel: se calculează pentru setul de validare eroarea medie pătratică pentru fiecare regulă. Apoi, lista regulilor este sortată crescător după această eroare, iar jumătate din regulile cu eroare mare sunt eliminate. În acest fel, se face o triere a regulilor obţinute pentru a asigura calitatea recomandărilor furnizate. Eliminarea regulilor se face după etapa în care se determină valoarea optimă pentru pragul *minconf*, chiar înainte de etapa de testare.

Iniţial, setul de date este divizat în 20% date de testare, pe baza cărora se dau valorile raportate în secţiunea de rezultate experimentale. Restul de date constituie mulţimea pe care se face antrenare şi validare. Pentru a determina o valoare optimă pentru *minconf*, setul de date de antrenare şi validare este împărţit conform metodei k-fold cross validation cu $k = 10$. Pe rând, fiecare din cele 10 partiţii este folosită pentru validare iar cele 9 rămase pentru antrenare. La fiecare pas se calculează eroarea medie pătratică obţinută pe setul de validare. În final se realizează o medie a rezultatelor obţinute în cele 10 situaţii, în acest fel reducându-se variabilitatea estimării. Prin această metodă se determină valoarea optimă pentru pragul *minconf*, eliminându–se necesitatea de a "ghici" valoarea potrivită. Aceasta este unul din punctele nevralgice ale multor lucrări ce pornesc de la algoritmul Apriori, valorile pentru parametri fie dându–se de către autori fără a arăta cum au fost alese, fie raportându–se rezultatele pentru mai multe valori ale parametrilor, fără a mai avea un criteriu care să dea unicul model ce se testează.

4 Rezultate experimentale

În această secţiune sunt descrise rezultatele obţinute prin compararea performanţelor algoritmului prezentat anterior în raport un alţi algoritmi folosiţi în procesul de recomandare.

În experimentele efectuate am folosit un setul de date public MovieLens [16], care conţine 100.000 de rating-uri date de 943 de utilizatori pentru 1682 de filme. Rating-urile reprezintă numerele intregi de la 1 la 5.

După alegerea valorii celei mai bune pentru *minconf* prin 10-fold cross validation, se consideră seturile de antrenare si validare astfel: 30% din datele rămase după extragerea setului de testare constituie setul de antrenare, iar restul datelor formează setul de validare. Acest pas este utilizat pentru a putea reduce numărul de reguli, prin calcularea erorii medii pătratice per regulă pe setul de validare.

Una dintre cele mai populare metode de evaluare folosite în cadrul inteligenţei artificiale şi nu numai, este calculul erorii pătratice (RMSE) definită ca:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(x_{1,i} - x_{2,i})^2}{n}}$$

unde $x_{1,i}$ reprezintă rating-ul dat de utilizator şi $x_{2,i}$ reprezintă valoarea obţinută în urma procesului de predicţie, valoare care este rotunjită la cel mai apropiat întreg.

În urma procesului de validare aplicat algoritmului prezentat în această lucrare s-au obţinut următoarele rezultate pentru *minconf*:

Table 1: Rezultatele obţinute prin cross-validation pentru pragul *minconf*

minconf	average(RMSE)	average($\tau(\lambda)$)
0.7	0.93099	1.00585
0.75	0.92216	1.04964
0.8	0.90569	1.15677

În acest caz funcţia τ reprezintă suma dintre eroarea medie pătratică şi o funcţie de forma λ/n. n constituie numărul de reguli care au putut fi folosite pentru a da recomandări utilizatorilor din seturile de validare, iar λ a fost ales 10. Aşadar funcţia τ arată astfel:

$$\tau(\lambda) = RMSE + \frac{\lambda}{n} \tag{7}$$

Factorul λ/n controlează numărul de reguli pentru care se calculează eroarea pătratică medie; se favorizează astfel situaţiile în care numărul de reguli pe baza căruia se calculează eroarea este mare. Motivul este că un set mic de reguli poate să ducă la o eroare mică, dar puterea predictivă a setului este redusă la etapa de testare.

Se poate observa că cele mai bune valori ale măsurilor de testare se obţin pentru *minconf*=0.7. Prin urmare, în experimentele viitoare se va folosi această valoare.

Tot pe baza metodei cross validation s-au testat valorile parametrului k pentru algoritmul k-NN şi s-au obţinut următoarele valori:

Table 2: Rezultatele obţinute prin cross-validation pentru parametrul k

k	average(RMSE)	$average(\tau)$
50	1.42022	1.44972
100	1.42888	1.45332
200	1.32193	1.34334
300	1.15952	1.17841

Din tabelul 2 se poate vedea că cele mai bune rezultate se obţin pentru $k = 300$ de vecini.

Similar s-a procedat şi pentru algoritmul FARAMS ([11]) la care s-a adăugat partea de regresie liniară pentru a putea compara rezultatele. De această dată s-au luat în considerare valori atât pentru pragul *minconf* cât şi pentru *minsup*. Astfel, deşi algoritmul prezentat în [11], foloseşte în locul pragului *minsup*, un alt prag în care se specifică numarul de reguli ce trebuie obţinute, am preferat această abordare, deoarece diferenţele nu sunt semnificative. Acestea pot fi observate în tabelul 3.

În experimentele viitoare vom utiliza deci *minconf*=0.2 şi *minsup*=0.1 pentru algoritmul FARAMS cu regresie liniară.

În ceea ce urmează vom ilustra rezultatele comparative obţinute pe algoritmii k-NN, FARAMS şi algoritmul introdus în lucrarea de faţă, utilizând valorile parametrilor precizate anterior. Rezultatele sunt date în tabelul 4.

Table 3: Rezultatele obţinute prin cross-validation pentru *minconf* şi *minsup*, pentru FARAMS.

minconf	minsup	average(RMSE)	average($\tau(\lambda)$)
0.2	0.1	0.99457	1.03441
0.2	0.2	1.07865	1.17560
0.2	0.3	1.0052	1.31209
0.3	0.1	1.07689	1.16019
0.3	0.2	1.19308	1.08989
0.3	0.3	1.31383	1.00636
0.4	0.1	1.15917	1.05750
0.4	0.2	1.20020	1.06060
0.4	0.3	1.31455	1.00343

Table 4: Rezultatele comparative

algoritm	RMSE
k-NN	1.16190
FARAMS cu regresie liniară	0.97511
algoritmul nou introdus	0.93122

Prin urmare, se poate observa o îmbunătătire a rezultatelor obţinute cu noul algoritm atât faţa de algoritmul k-NN, cât şi faţă de algoritmul FARAMS cu regresie liniară.

Implementarea aplicaţiei este făcută în limbajul C#, folosind platforma .NET Framework 3.5 SP1; baza de date de filme, utilizatori şi rating-uri este accesată prin intermediul lui SQL Server 2005 Express Edition, iar interfaţa utilizator este realizată în ASP.NET 3.5. Sistemul permite unui utilizator care a dat anterior nişte rating-uri să vizualizeze alte filme, în ordinea descrescătoare a rating-urilor estimate de către aplicaţie.

5 Concluzii

Scopul acestei lucrări a fost de a propune un nou model, prin intermediul căruia să se ofere posibile soluţii problemelor apărute în implementarea algoritmului Apriori. De asemenea s-a dorit să se realizeze o extindere a modelului original, în direcţia evaluării de rating, necesară în momentul utilizării algoritmului pentru sistemele de recomandări.

Pornind de la articolele deja existente pe această temă, am observat câteva dintre problemele cu care algoritmii de recomandări se confruntă şi am încercat să oferim o soluţie viabilă în acest sens.

Spre deosebire de majoritatea algoritmilor de extragere a regulilor de asociere, în care utilizatorul este nevoit să precizeze valori pentru parametri, precum suportul minim, în varianta furnizată în această lucrare problema a fost evitată prin introducerea unui prag ce se calculează automat pe parcursul execuţiei algoritmului.

O altă îmbunătăţire adusă algoritmului constă în reducerea numărului de reguli obţinute, prin impunerea condiţiei ca fiecare regulă să aibă un singur element în consecvent şi maximum două elemente în antecedent. În plus, regulile sunt eliminate şi pe baza erorii medii pătratice obţinute pe setul de validare. Se mişorează astfel numărul de calcule necesare, fără a se sacrifica din performanţa algoritmului.

Ceea ce trebuie, de asemenea, specificat este faptul ca timpul de obtinere a suportului unei reguli poate fi imbunatatit prin utilizarea pentru fiecare articol a unei liste ce contine utilizatorii care si-au exprimat parerea referitoare la articolul in cauza.

Nu trebuie însă neglijată importanţa valorii confidenţei minime în obţinerea unor reguli optime. În cazul de faţă, această valoare a fost calculată pe baza unor teste pe seturi diferite de validare. Cu toate acestea o posibilă direcţie viitoare ar fi găsirea unei metode de înlocuire a acestei valori.

În concluzie, algoritmul prezentat oferă numeroase avantaje, comparativ cu alte versiuni ale algoritmilor de recomanări.

Bibliografie

[1] ***, *The Netflix Prize Rules.* http://www.netflixprize.com/rules

[2] Burke R, Knowledge-based Recommender Systems. *Encyclopedia of Library and Information Systems.*,Vol. 69, Supplement 32. New York: Marcel Dekker.

[3] Shardanand U., Maes P., *Social Information Filtering: Algorithms for Automating 'Word of Mouth'.*, Proceedings of ACM CHI'95 Conference on Human Factors in Com- puting Systems, May 7-11, Denver, Colorado

[4] Toby Segaran, *Programming Collective Intelligence* editura O'Reilly, 2007

[5] Haralambos Marmanis, Dmitry Babenko *Algorithms of the Intelligent Web*, Manning Publications Co., 2009

[6] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *Elements of Statistical Learning*, Stanford, California, 2008

[7] Richard O. Duda, Peter E. Hart, David G. Stork, *Pattern Classification*, Willey Publishing, 2000

[8] D. Billsus, M.J. Pazzani *Learning Collaborative Information Filters Proc of the fifteenth International Conference on Machine Learning* Wisconsin, Morgan Kaufmann Publishers

[9] C. Kim, J. Kim, *A Recommendation Algorithm Using Multi-level Association Rules.* Proceeding of the IEEE/WIC International Conference on Web Intelligence, October 13-17, Halifax, Canada

[10] W. Lin, S.A. Alvarez, C. Ruiz, *Efficient Adaptive-Support Association Rule Mining for Recommender Systems., Data Mining and Knowledge Discovery*

[11] Cane Wing-ki Leung, Stephen Chi-fai Chan, Fu-lai Chung, *A Collaborative Filtering Framework Based on Fuzzy Association Rules and Multiple-Level Similarity.* Knowledge and Information Systems

[12] R. Agrawal, R. Srikant, *Fast Algorithms for Mining Association Rules.* Proceedings of the 20th International Conference on Very Large Databases, September 1994, Santiago de Chile, Chile

[13] Xindong Wu, Vipin Kumar *The Top Ten Algorithms in Data Mining.* Data Mining and Knowledge Discovery Series. CRC Press

[14] Michael Berry, Gordon Linoff, *Data Mining Techniques For Marketing Sales And Customer Relationship Management* Wiley Publishing, 2004

[15] Wen-Yang Lin, Ming-Cheng Tseng *Automated support specification for efficient mining of interesting association rules*

[16] ***, *MovieLens Data Sets.* http://www.grouplens.org/node/73

VÎRNĂ Claudia
Universitatea "Transilvania"
Facultatea de Matematică şi Informatică
Bulevardul Iuliu Maniu nr. 50,
Braşov
ROMÂNIA
E-mail:
virna.claudia@gmail.com

PARTEA 2

Secţiunea dedicată elevilor

Sesiunea Naţională de Comunicări Ştiinţifice a Studenţilor
"Imaginaţie, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

Aplicaţii grafice în C++

Diana Nastase, Adina Gaja, Claudiu Bruda
Coordonator: prof. Monica Oancea

Abstract

This project aims to present more problems related to computer-intensive curriculum, XIth grade . We will present programming techniques Divide et impera, backtracking, and applications with simple linear linked lists and circular double linked lists. In the context of programming in Borland C++, advanced programming techniques require hard work, many lines of code and text mode display results do not reflect the amount of work.. To underline the results of the problem we resorted to posting the output in graphic mode using graphics.h library. With the help of colors, special fonts and effects, applications of the techniques presented above can be presented in a more attractive manner. We specify that work in graphics mode C++ is not included in the curriculum of high school, but under the guidance of our teacher we studied the graphics library's functions and tried to solve the proposed problems graphically also.

1 Introducere

Lucrarea isi propune sa prezinte mai multe probleme ce tin de programa de informatica-intensiv, clasa a XI-a. Se vor prezenta tehnicile de programare Divide-et-impera, Backtracking, si aplicatii cu liste liniare simplu înlănţuite şi circulare dublu inlantuite.

In contextul programarii in Borland C++, aplicarea tehnicilor de programare complexe presupune o munca laborioasa, multe linii de cod, iar modul text de afisare a rezultatelor nu reflecta indeajuns munca depusa.
Pentru a evidentia rezultatele problemelor am apelat la afisarea in modul grafic cu ajutorul librariei graphics.h.
Astfel cu ajutorul culorilor, fonturilor speciale se pot prezenta aplicatii la tehnicile prezentate mai sus intr-un mod atractiv.
Specificam ca lucrul in modul grafic C++ nu este inclus in programa de liceu, dar sub indrumarea d-nei profesoare am studiat functiile librariei graphics si am incercat sa rezolvam problemele propuse si in mod grafic.

2 Noţiuni teoretice

2.1. **Metoda Backtracking** este exemplificată prin:

- problema steagurilor;

Se dau N culori. Sa se genereze toate steagurile cu trei culori distincte avand la mijloc doar una din ultimele doua culori citite.

- problema cuburilor;

Fiind date n cuburi etichetate de la 1 la n, de laturi Li si culori ci, 1<=i<=n, sa se afiseze toate turnurile de h cuburi care se pot forma, astfel incat cuburile din turn sa aiba laturile in ordine descrescatoare, iar culorile cuburilor alaturate sa fie diferite.

-problema damelor;

*Fiind o tabla de sah cu dimensiunea n*n, se cer toate solutiile de aranjare a n dame, astfel incat sa nu se afle doua dame pe aceiasi linie, coloana sau diagonala (damele sa nu se atace reciproc).*

2.2 Metoda Divide-et-Impera

- problema Turnurilor din Hanoi;

Se dau trei tije simbolizate prin a,b,c. Pe tija a se gasesc discuri de diametre diferite, asezate in ordine descrescatoare a diametrelor privite de jos in sus. Se cere sa se mute discurile de pe tija a pe tija b utilizand ca tija intermediara tija c, respectand urmatoarele reguli:

- *la fiecare pas se muta un singur disc*
- *nu este permis sa se aseze un disc cu diametrul mai mare peste un disc cu diametrul mai mic*

- Fulgul lui Koch pentru un triunghi echilateral;

Se considera un triunghi echilateral. Fiecare latura a sa se transforma astfel incat sa se imparta in trei segmente congruente, se elimina segmentul din mijloc si se construieste deasupra un triunghi echilateral. Fiecare latura a acestui poligon se transforma din nou, dupa aceeasi regula. Se sa se vizualizeze figura obtinuta dupa ls pasi.

- Fractali –Arborele

*Se da un segment AB, cu ajutorul acestuia se construieste un arbore. Lungimea fiecarei ramuri este 1/3 din lungimea initiala a segmentului. Fiecare latura se transfoarma in mod asemanator. Se cere sa se vizualizeze figura astfel rezultata, dupa **ls** transformari.*

2.3 Liste liniare

- problema Joc de cărti-Război;

Fiecarui jucator i se vor distribui aleator 16 carti numerotate de la 1 la 8. Jucatorii vor intoarce cate o carte de deasupra mini-pachetului. Jucatorul cu cartea cea mai mare va lua cartile si le va pune sub cartile sale, acestea urmand a fi repuse in joc. In caz de egalitate numerica, jucatorii vor intoarce un numar de carti egal cu numerele cartilor initiale. Jucatorul cu cea mai carte dupa razboi ia toate cartile puse in joc. Pierde jucatorul care ramane fara carti.

- **problema Seifului-aplicatie la listă dublu înlănţuită circulară**

Se dau n numere dispuse intr-o lista circulara dublu inlantuita. Din fisierul de intrare se citeste o secventa codificata (5d 2s 4d 1s) ce reprezinta succesiunea de rotiri stanga/dreapta necesare pentru a deschide seiful. Sa se afle codul de acces al seifului.

3 Codul sursă a programului

```
#include <iostream.h>#include <math.h>#include <conio.h>#include <stdlib.h>#include <graphics.h>
#include <string.h>#include <dos.h>#include <time.h>
struct nod { int t;     int cl;       nod *urm;                 };
nod *c1, *c2, *c3, *ul1, *ul2, *ul3, d1, d2; int f[9],dly=1000, n1, n2;
void ad(nod *&c, nod car, nod *&ul)
{ nod*q;q=new nod;  q->t=car.t;   q->cl=car.cl;  q->urm=NULL;
  if(c==NULL)  {  c=q;  ul=q;}
  else  {ul->urm = q;    ul=q;}
  if (c==c1)   n1++;  if (c==c2)   n2++;}
void distribuire_carti(nod *&c1, nod *&c2)
{ int ct=0, semn=1;  nod x;  c1=NULL; c2=NULL;  time_t t;  srand((unsigned) time(&t));
  while(ct<32)
  { x.t=1+rand()%8;
    if(f[x.t]<4)   {  f[x.t]++;   ct++;x.cl=f[x.t]+2;if(semn>0) ad(c1,x, ul1);else ad(c2, x, ul2);semn=semn*(-1);}}
  }
nod st(nod*&c)
{ nod*q,x;  x.t=c->t;  x.cl=c->cl;  q=c; c=c->urm;  delete q;  if (c==c1)   n1--;  if (c==c2)   n2--;
  return x;
}
nod stj1()
{ nod x = st(c1);
 if (c1==NULL)
 { cleardevice();   settextstyle(TRIPLEX_FONT, 0, 6);setcolor(YELLOW);outtextxy(10,200,"Jucatorul 2 a
castigat");   delay(4000);   exit(0); } return x;
}
nod stj2()
{ nod x = st(c2);
 if (c2==NULL)
 { cleardevice(); settextstyle(TRIPLEX_FONT, 0, 6); setcolor(YELLOW);outtextxy(10,200,"Jucatorul 1 a
castigat");   delay(4000);   exit(0); } return x;
}
void parcurg(nod *prim)
{ cout<<"Lista este: \n";
  for( ;prim!=NULL; prim=prim->urm)    cout<<"n:"<<prim->t<<" c:"<<prim->cl<<"   ";  cout<<endl;
}
void egalitate(nod d1, nod d2)
{ int num=d1.t;   settextstyle(TRIPLEX_FONT, 0, 6);   setcolor(YELLOW);   outtextxy(200,20,"RAZBOI!");
cout<<'\x7';  delay(500);  ad(c3, d1, ul3);  ad(c3, d2, ul3);
  while(num - 1 > 0)
   {     nod carte1 = stj1();nod carte2 = stj2(); ad(c3, carte1, ul3); ad(c3, carte2, ul3);  num--; }
  d1=stj1(); d2=stj2();
 if(d1.t<d2.t) { while(c3!=NULL) ad(c2, st(c3), ul2); ad(c2, d1, ul2); ad(c2, d2, ul2);
settextstyle(TRIPLEX_FONT,   0,   4);   setcolor(YELLOW);   outtextxy(140,400,"Castiga   jucatorul   2");
delay(100);    }  else if(d1.t>d2.t) { while(c3!=NULL) ad(c1, st(c3), ul1);   ad(c1, d2, ul1);   ad(c1, d1, ul1);
          settextstyle(TRIPLEX_FONT, 0, 4);  setcolor(YELLOW);outtextxy(140,400,"Castiga jucatorul 1");
delay(1000); }  else egalitate(d1, d2);
}
void afisare(nod cr1,nod cr2)
{ char s1[2],s2[2], nr1[2], nr2[2], *p;
 s1[0]=cr1.cl; s2[0]=cr2.cl; s1[1]=s2[1]=0; nr1[0]=cr1.t+'0'; nr2[0]=cr2.t+'0'; nr1[1]=nr2[1]=0;
 cleardevice(); setcolor(YELLOW); settextstyle(TRIPLEX_FONT, 0, 2); outtextxy(90,110,"Jucator 1");
 outtextxy(440,110,"Jucator 2"); //Juc 1
 setcolor(LIGHTBLUE); rectangle(60,150,200,350); setcolor(WHITE); floodfill(61,151,LIGHTBLUE);
```

```
 if (cr1.cl<=4)  setcolor(RED); else  setcolor(BLACK);
 settextstyle(DEFAULT_FONT, 0, 8); outtextxy(105,215,s1); settextstyle(DEFAULT_FONT, 0, 5);
 outtextxy(62,152,nr1); outtextxy(162,312,nr1); itoa(n1,p,10); settextstyle(DEFAULT_FONT, 0, 3);
 setcolor(YELLOW); outtextxy(110,370,p); //Juc 2
 setcolor(LIGHTBLUE); rectangle(410,150,550,350); setcolor(WHITE); floodfill(411,151,LIGHTBLUE);
 if (cr2.cl<=4)  setcolor(RED); else  setcolor(BLACK);
 settextstyle(DEFAULT_FONT, 0, 8); outtextxy(460,215,s2); settextstyle(DEFAULT_FONT, 0, 5);
 outtextxy(412,152,nr2); outtextxy(512,312,nr2); itoa(n2,p,10); settextstyle(DEFAULT_FONT, 0, 3);
 setcolor(YELLOW); outtextxy(460,370,p);}
void reguli()
{ char s[]="              **** REGULI DE JOC ****\n\n\
  Fiecarui jucator i se vor distribui aleator 16 carti numerotate de la 1 la 8. Jucatorii vor intoarce cate o
carte  de deasupra mini-pachetului. Jucatorul cu cartea cea mai mare  va lua cartile si le va pune sub
cartile sale, acestea urmand a fi repuse in joc. In caz de egalitate numerica, jucatorii vor intoarce un
numar de carti egal cu numerele cartilor  initiale. Jucatorul cu cea mai carte dupa razboi ia toate
 cartile puse in joc. Pierde jucatorul care ramane fara carti.";
 int i,ct=0; for (i=0;i<strlen(s);i++) {cout<<s[i];  if (kbhit())ct=1;if (ct==0)delay(50); }getch();
}
void main()
{ int gdriver = DETECT, gmode, errorcode;
  initgraph(&gdriver, &gmode, "");
  distribuire_carti(c1, c2);
  reguli();  getch();
  while(c1!=NULL && c2!=NULL)
   { d1=stj1();     d2=stj2();     afisare(d1,d2);
     if(d1.t<d2.t) { ad(c2, d1, ul2);          ad(c2, d2, ul2);              }
     else if(d1.t>d2.t) { ad(c1, d2, ul1);    ad(c1, d1, ul1);                 }
     else {     egalitate(d1, d2);     }       delay(dly);
      if (kbhit())       dly = 300;   }
   getch();
}
```

Bibliografie

[1] Tudor Sorin , Vlad Hutanu, *Manual de informatica intensiv cls. XI,*Bucuresti 2006.
[2] Mariana Milosescu, *Manual informatica intensive*, Bucuresti 2006.
[3] Carmen Popescu,,Monica Oancea, Georgeta Preda *Culegere de informatica pentru clasa a X-a si a XI-a ;Sibiu 2007.*
[4] www.deviantart.com
[5] www.yahoo.com

Diana Nastase
Colegiul National “Gheorghe Lazar”
XI- G Intensiv informatica
Str. Gheorghe Lazar, nr.3
Romania
diana_n13@yahoo.ca

Adina_Gaja
Colegiul National “Gheorghe Lazar”
XI- G Intensiv informatica
Str. Gheorghe Lazar, nr.3
Romania
adina_gaja@yahoo.com

Claudiu Bruda
Colegiul National “Gheorghe Lazar”
XI- G Intensiv informatica
Str. Gheorghe Lazar, nr.3
Romania
claudiu725@yahoo.com

Sesiunea Națională de Comunicări Ştiințifice a Studenților
"Imaginație, Creativitate, Design, Dezvoltare"
Editia a II-a, Sibiu – România, 2010

SWOOP – A P2P Application

Victor-Gabriel Savu, Andra-Florina Mureşanu
Coordonator: prof. Delilah Florea

Abstract

When trying to communicate online, people can use two different methods: either use a server based service or try out a peer to peer application. P2P raises a set of questions: where is the other person and how do I reach him? (DNS is hard to use when the other person has different IPs at different locations, NAT poses problems), how can you be sure that the person behind the IP is the one you want to talk to?

In order to figure out a way through which people can communicate with other people without using complicated IPs or ports, we have come up with a solution that implements P2P in the background and offers high level services to the user.

This application suits the regular costumer (who can chat and send files through SWOOP) as well as businesses, allowing them to build another application (only needing to implement OSI layers 6 and 7) based on the provided SWOOP services.

1 Introduction

In the attempt of simplifying the management and operations of servers within companies and offering in the same time an alternative to server based solutions to the end user, we have created an application that suits both businesses and regular Internet users, providing them with the opportunity to use a peer to peer system in a very simple manner.

In this paper we will describe the theoretical model on which SWOOP, our application, was built and its actual implementation.

The Gnutella protocol and all associated applications offer a similar functionality, but only at an elementary level, lacking basic authentication (and indirectly locating of a known user), security or any other services besides file-sharing. Thus, SWOOP provides a more complete feature set including the capacity to find a specific person using an advanced form of authentication system and security by making encrypted communications possible when required. The main characteristic of SWOOP is that it can be extended to suit the needs of virtually any type of user, because other programs can use its underlying systems to offer that specific needed feature.

Our paper is structured in 4 main parts. The first chapter describes the links that users form and

how they are propagated across the network. In the second part we explain the authentication and security system that is used by SWOOP. The third chapter contains the actual protocol used by the main application to communicate to other instances. Finally, the last part shows how SWOOP can be extended by other programs.

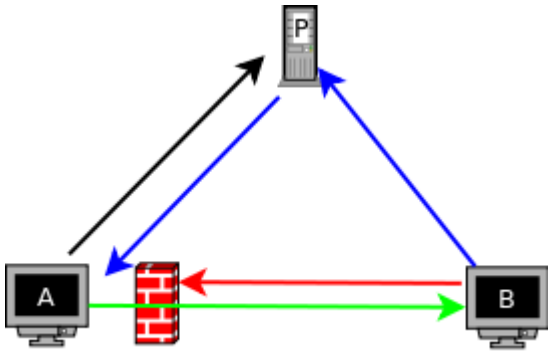
2 The Network

Definition 1 *A Node is a running instance of the SWOOP application having an associated address consisting of an IP address and a port which identifies itself using a predetermined public key.*
Definition 2 *A link is a connection between two nodes.*

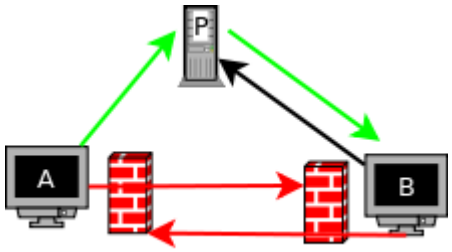
Nodes must be accessible from a remote address (port is open and accessible from the Internet) so that incoming links can be formed. The next subchapter defines the behavior when this condition is not met.

2.1 Proxy

The first case solved by a proxy is when only one node resides behind a firewall (**A**) that does not permit access, but the calling node (**B**) has an open connection. **A** notifies another node (**P**) that it should be accessed through it. **P** then becomes a proxy for **A**. When **B** is searching for **A**, **P** responds. **B** then sends a message to **A** through **P** so that **A** can connect to **B** forming a link.



When both nodes forming a link are behind a firewall, a full proxy is required. The SWOOP Application refuses such functionality by default because of the high bandwidth requirement, but some users will enable it. A full proxy must retransmit all messages between **A** and **B**.



2.2 Link discovery

In order for a node to form a link with another node the latter's address must be known. When the application is closed all connections are lost so a node must store a list of former addresses to be able to reenter the network. When a new connection is established the node updates its database of known addresses with the information received from the new node.

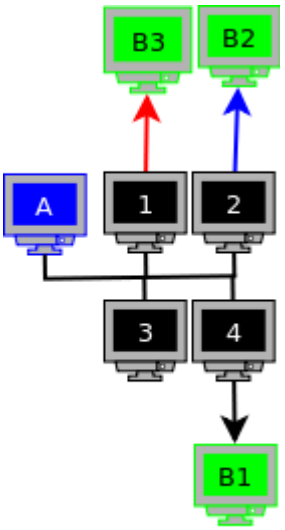
Nodes maintain the following information about known addresses:

a) **live connections:** These connections are currently active so the addresses are up to date.
b) **recent connections:** The node has found a node at these addresses, the time the connection was lost is also recorded.
c) **learned connections**: The node has received these addresses from other nodes so it is uncertain of their validity. These are used as a last resort when all known recent connections are exhausted.

The address database contains a large number of records to make sure that a node always has someone to connect to.

2.3 Search

When looking for a specific node, the application firstly searches through the live connections. If it does not find a link to the node, it tries to create one. The method used is to locate the node among the recent or learned connections.



If all else fails, the node asks all its linked nodes to execute the same search and send back the results. For example, when **A** is searching for **B** and a link does not exist between them, **A** will call for **1, 2, 3** and **4** to search among their links. **1** returns a learned connection (the red arrow), **2** brings back a recent connection (the blue arrow) and **4** returns a live connection (the black arrow). This is why **A** will reach for the address it received from **4**.

3 Security

3.1 Methods

Swoop uses RSA public key cryptography to ensure the authentication of the nodes and the encryption of the messages when required. Symmetrical key cryptography cannot be used in this scenario because the secure sharing of the key is impossible.

3.2 Authentication

Each node is identified by its public key and authenticates itself during the beginning of a connection. Because of the 2048 bit key used it is very unlikely that a collision can appear. In this case however the private key is already compromised and both users must choose another key.

3.3 Encryption

Because of public key cryptography, node A can securely send any data to another node, B, knowing only B's public key. Nevertheless, encryption is processing-intensive and is disabled by default in most cases.

4 Protocol

All communication between the nodes is handled by the scala remote actors library. This offers a simple and reliable method to transport the high level data across the network.

The following messages are defined:

```
class Communication(time:Date)
class Message extends Communication(new Date())
```

These are the base classes for all communications between nodes.

```
case class End extends Message
case class Accept(to:Message) extends Message
```

```
case class Refuse(to:Message) extends Message
```

These are utility classes to signal choices.

```
case class TestConnect(port:Int,rand:Int) extends Message
case class TestResponce(rand:Int,youreaddress:Address) extends Message
```

These two classes are used by a node to determine if he has an open port.

```
case class Connect(me:PKey,you:Option[PKey],
        youreaddress:Address,myaddress:Option[Address],
        proxy:Boolean,sign:Option[Array[Byte]]) extends Message
```

This class will precede all communication between to nodes. Each one identifies itself and proves its identity by signing the same message and storing it in the sign field.

```
case class Encrypt extends Message
case class SendE(key:Option[PKey], message:Array[Byte]) extends Message
```

If a node requests encryption and the second one accepts all further messages will be encrypted and sent using the sendE class. The message field contains the serialized message in its encrypted form.

```
case class Proxy extends Message
case class ProxyResponce(address:Option[Address]) extends Message
case class ProxyFor(key: List[PKey]) extends Message
case class FullProxyRequest(to:Address,key:PKey) extends Message
case class CallRequest extends Message
case class SendS(key:Option[PKey], message:Message) extends Message
```

The situations in which a proxy is used are described in chapter 2. These are the classes that initialize the specific modes and enable communication.

```
class Connection(key:PKey,address:Address,last:Date)
case class ConnectionsRequest(all:Boolean) extends Message
case class CurrentConnections(who:List[Connection]) extends Message
case class MyRecentConnections(who:List[Connection]) extends Message
case class AllConnections(who:List[Connection]) extends Message
```

When a connection is established the two nodes share these classes on request to extend their databases. The CurrentConnections class is always sent.

```
case class Search(who:PKey,all:Boolean = false) extends Message
case class SearchReply(reply:Connection) extends Message
```

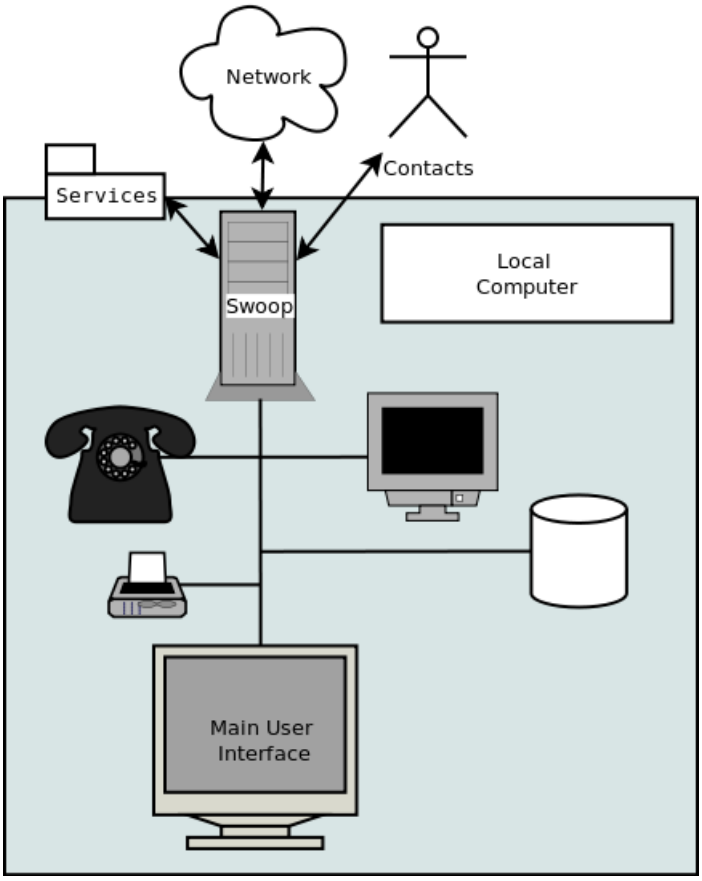
The above classes are used to implement search.

```
case class DataTransport(channel:Int,data:Array[Byte]) extends Message
case class OpenChannel(application:String,channel:Int) extends Message
```

Any application can open a connection to a remote node, SWOOP handling the multiplexing and negotiation.

5 SWOOP

5.1 SWOOP service

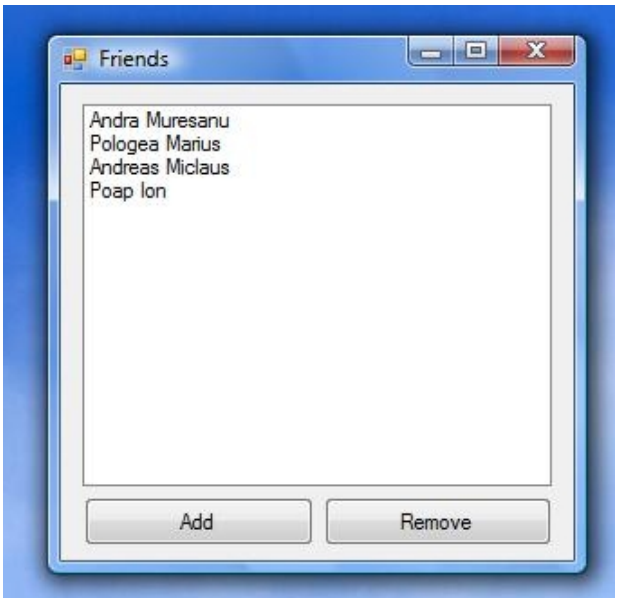


SWOOP starts at system start-up, opens a port (by default 3325) and listens for a connection from the main interface. All communication is done using that socket.

The Main User Interface first sends an authentication packet that was predetermined and is known to both application and service. All commands for the server are written in Standard ECMA-262 ECMAScript Edition 3 (JavaScript) and are interpreted by the service using Rhino.

The server sends asynchronous messages to the interface using a symbol protocol, a two-byte code representing the type of user input required. The interface must take action upon that message and send an appropriate command (in JavaScript) back to the service.

5.2 Interfaces



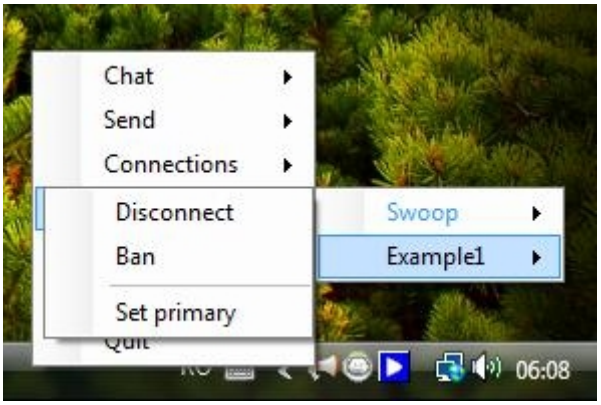
SWOOP is a service that cannot be used without the Main User Interface. By accessing this interface you can control all aspects of SWOOP (for example its settings).

5.2.1 Main Interface

The main interface must implement three aspects:

a) interaction with the user (for example: “Do you want to continue the communication even if the other node refused encryption?”)
b) ability to edit settings and lists (Friends, Ignore, Ban)
c) control over the other applications

The role of Main Interface can be taken over by any program connected to the service upon user input.



5.2.2 Secondary Interfaces

Secondary Interfaces use SWOOP for networking and offer additional functionality specifically requested or needed by the user.

5.3 Security for applications

SWOOP always keeps a port open so anyone on the local machine can connect to it. The application closes any connection that does not come from an unauthorized program or, if it is a new program, asks the user how to proceed.

6 Conclusions

SWOOP can be further developed to suit the needs of a large company whose employees from different departments would collaborate more effectively in providing feedback to each other. For example, a lot of useful documents that are available for the marketing department of a company could simply be transferred to the sales managers and help them elaborate a strategy to increase the sales of their products. This is also a way to lower the production costs, because there is no need for large datacenters to support their employees.

Furthermore, SWOOP and other peer to peer applications could one day replace the server client paradigm due to the rise in bandwidth available to the regular user.

References

[1] Martin Odersky, Lex Spoon, Bill Venners, *Programming in Scala*, Artima Press, Mountain View, California, 2007
[2] http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf
[3] https://developer.mozilla.org/en/Rhino_documentation
[4] http://tools.ietf.org/html/rfc3447

SAVU Victor-Gabriel
C. N. “Samuel von Brukenthal”
Matematică-informatică, informatică intensiv
Piaţa Huet nr. 5, Sibiu
ROMÂNIA
E-mail: victorsavu3@gmail.com

MUREŞANU Andra-Florina
C. N. “Samuel von Brukenthal”
Matematică-informatică, informatică intensiv
Piaţa Huet nr. 5, Sibiu
ROMÂNIA
E-mail: andra.muresanu@gmail.com

LISTA DE AUTORI

1. **ALEXE Iulian**
Universitatea “Lucian Blaga” din Sibiu
Facultatea de Ştiinţe
Informatică
Str. Dr. Ioan Raţiu, nr. 5-7, Sibiu
ROMÂNIA
E-mail: iulian.alexe@ulbsibiu.ro

2. **BAN Adriana**
Universitatea "Lucian Blaga" din Sibiu
Facultatea de Ştiinţe
Str. Dr. Ioan Raţiu nr. 5-7
ROMÂNIA
E-mail: ban.adriana@yahoo.com

3. **BOTA Florentin**
Universitatea „Lucian Blaga” din Sibiu
Facultatea de Ştiinţe
Informatica
str dr. Ioan Ratiu, nr 5-7, Sibiu
ROMÂNIA
E-mail: botaflorentin@yahoo.com

4. **BRUDA Claudiu**
Colegiul National “Gheorghe Lazar”
XI- G Intensiv informatica
Str. Gheorghe Lazar, nr.3
Romania
E-mail: claudiu725@yahoo.com

5. **BURCĂ-FLOREA Domnina**
Universitatea „Politehnica” din Bucureşti
Calculatoare
Spl. Independenţei Nr. 313, Bucureşti
ROMANIA
E-mail: domnina.burca@gmail.com

6. **CAPUZZO Francesco**
Universitatea Lucian Blaga
Facultatea de Ştiinţe
Informatică
Str. Dr. Ioan Raţiu, nr. 5-7, Sibiu
ROMÂNIA
E-mail: francesco.capuzzo@gmail.com

7. **CICONE Luiza**
Universitatea Politehnică Bucureşti
Calulatoare şi tehnologia informaţiei
Splaiul Independenţei nr.313, Bucureşti
ROMÂNIA
E-mail: luiza.cicone@rdslink.ro

8. **CONSTANTINESCU Marius**
Universitatea „Politehnica” din Bucureşti
Calculatoare
Spl. Independenței Nr. 313, Bucureşti
ROMANIA
E-mail: constantinescu.marius@gmail.com

9. **CONSTANTINESCU Tudor**
Universitatea Politehnica Bucuresti,
Facultatea de Inginerie în Limbi Straine
Splaiul Independentei 313,
Bucuresti
ROMANIA
E-mail: cttudor_kobe@yahoo.com

10. **CRISTEA Diana**
Universitatea "Lucian Blaga" Sibiu
Facultatea de Ştiințe
Str. Dr. Ioan Rațiu nr. 5-7
ROMÂNIA
E-mail: diana.cristea25@yahoo.com

11. **DODA Gheorghe**
Universitatea Lucian Blaga
Facultatea de Ştiințe
Informatica
Str. Dr. Ioan Rațiu, nr. 5-7, Sibiu
ROMÂNIA
E-mail: godlike_affairs@yahoo.com

12. **DRAGOMIR Alexandra Elena**
Universitatea Politehnica Bucuresti
Facultatea de Inginerie în Limbi Straine (FILS)
Electronica aplicata
Splaiul Independentei, 290
ROMANIA
E-mail: puttel_chermir@yahoo.com

13. **DROBOTĂ Florin-Robert**
Universitatea Transilvania Brasov
Informatică aplicată
Str. Iuliu Maniu, nr. 50, cod 500091, Brasov
ROMANIA
E-mail: robert.drobota@gmail.com

14. **FLESTER Iulian**
Universitatea Transilvania Brasov
Facultatea de Matematica-Informatica
Str.Iuliu Maniu,nr 50, Brasov
ROMANIA
E-mail: flester.iulian@gmail.com

15. GAJA Adina
Colegiul National "Gheorghe Lazar"
XI- G Intensiv informatica
Str. Gheorghe Lazar, nr.3
Romania
adina_gaja@yahoo.com

16. IVASCU Carina
Universitatea Transilvania Brasov
Facultatea de Matematica-Informatica
Str.Iuliu Maniu,nr 50, Brasov
ROMANIA

17. LAZAR Valentina
Universitatea „Lucian Blaga" din Sibiu
Facultatea de Ştiinţe
Informatica
Str. Ion Raţiu Nr.5-7, Sibiu, 550012, Sibiu
ROMANIA
E-mail: lyah_valy@yahoo.com

18. LUPESCU Grigore
FILS Politehnica Bucuresti
Computer Science, 2nd year
ROMANIA
E-mail: lupescu_grigore@hotmail.com

19. MORARIU Stelian
Universitatea Lucian Blaga
Facultatea de Ştiinţe
Informatica
Str. Dr. Ioan Raţiu, nr. 5-7, Sibiu
ROMÂNIA
E-mail: modev_st@yahoo.com

20. MUREŞANU Andra-Florina
C. N. "Samuel von Brukenthal"
Matematică-informatică, informatică intensiv
Piaţa Huet nr. 5, Sibiu
ROMÂNIA
E-mail: andra.muresanu@gmail.com

21. NASTASE Diana
Colegiul National "Gheorghe Lazar"
XI- G Intensiv informatica
Str. Gheorghe Lazar, nr.3
Romania
E-mail: diana_n13@yahoo.ca

22. NECHIFOR Laura
Universitatea Politehnica din Bucuresti
Electronica aplicata
Splaiul Independentei 313, Bucuresti
ROMANIA
E-mail: lauri_mari@yahoo.com

23. NECHIFOR Vasile Nicuşor
Universitatea „Lucian Blaga”
Informatică
Str. dr. I. Ratiu, nr. 5-7, cod 550012, Sibiu
ROMANIA
E-mail: vasilenicusor@yahoo.com

24. NEGULICI Teona Daiana
Universitatea Politehnica din Bucuresti
Electronica aplicata
Splaiul Independentei 313, Bucuresti
ROMANIA
E-mail: teodaia@yahoo.com

25. OBANCEA Dragoş Iulian
Universitatea „Transilvania” din Braşov
Facultatea de Matematică şi Informatică
Specializarea Informatică
Strada Iuliu Maniu, Nr. 50, Braşov
ROMÂNIA
E-mail: obancea_dragos@yahoo.com

26. PANDARU Raluca
Universitatea Transilvania Brasov
ROMANIA
E-mail: raluca_pandaru@yahoo.com

27. POPA Bogdan Constantin
Universitatea Transilvania
Facultatea de Matematica si Informatica
Specializarea Informatica Aplicata
Str. Iuliu Maniu, nr. 50, Cod postal: 500091, Brasov
ROMANIA
E-mail: bogdi@ymail.com

28. RADU Sorin Daniel
Universitatea „Lucian Blaga”
Informatică
Str. dr. I. Ratiu, nr. 5-7, cod 550012, Sibiu
ROMANIA
E-mail: radu_s_o_r_i_n@yahoo.com

29. ROMAN Cristina Elena
Universitatea Transilvania Brasov
Informatică aplicată
Str. Iuliu Maniu, nr. 50, cod 500091, Brasov
ROMANIA
E-mail: cri_roman@yahoo.com

30. RUSU Andrei
Universitatea Lucian Blaga, Sibiu
Facultatea de Ştiinţe, Sibiu
Str. Dr. Ion Ratiu, 5-7
ROMANIA
E-mail: rusu.andy1@gmail.com

31. SAVU Victor-Gabriel
C. N. "Samuel von Brukenthal"
Matematică-informatică, informatică intensiv
Piața Huet nr. 5, Sibiu
ROMÂNIA
E-mail: victorsavu3@gmail.com

32. STANCU Mihai
Universitatea „Lucian Blaga" din Sibiu
Facultatea de Ştiințe
Specializarea Matematica-Informatică
Bd-ul. Victoriei, Nr. 10, Sibiu, 550024
ROMANIA
E-mail: stancu_meehigh@yahoo.com

33. STOICA Lucian
Universitatea „Lucian Blaga", Sibiu
Master Tehnologia Informatiei
Str. Dr. I. Ratiu, Nr. 5-7
ROMANIA
E-mail: luci15ian@gmail.com

34. STOICESCU Monica
Universitatea Politehnica Bucuresti
Facultatea de Inginerie în Limbi Straine
Splaiul Independenței 313 Bucuresti
ROMANIA
E-mail: monica.stoicescu00@gmail.com

35. SZASZ Zsigmond-Attila
Universitatea Transilvania Brasov
Facultatea de Matematica-Informatica
Specializarea: Informatica
Iuliu Maniu 50, Brasov
ROMANIA
E-mail: zsigmond.szasz@xu.unitbv.ro

36. TAMASI Zoltan
Universitatea Transilvania Brasov
Facultatea de Matematica-Informatica
Specializarea: Informatica
Iuliu Maniu 50, Brasov
ROMANIA
E-mail: zoltan.tamasi@xu.unitbv.ro

37. VÎRNĂ Claudia
Universitatea Transilvania Brasov
Facultatea de Matematica-Informatica
Str.Iuliu Maniu,nr 50, Brasov
ROMANIA
E-mail: virna.claudia@gmail.com

38. VOLOSINCU Bogdan

Universitatea Lucian Blaga, Sibiu
Facultatea de Ştiinţe, Sibiu
Str. Dr. Ion Ratiu, 5-7
ROMANIA