Aplicații MDL în didactica informaticii

Daniel Safta

Universitatea Tehnică din Cluj-Napoca,
Facultatea de Automatică şi Calculatoare

Str. G.Bariţiu nr. 26-28

davonkeep@yahoo.com

Dorian Gorgan

Universitatea Tehnică din Cluj-Napoca,
Facultatea de Automatică şi Calculatoare

Str. G.Bariţiu nr. 26-28

dorian.gorgan@cs.utcluj.ro

REZUMAT

În acest articol vom prezenta aplicaţii ale modulului de dezvoltare a lecţiilor (MDL) - componenta de modelare a soluţiilor - în două lecţii de programarea calculatoarelor, lecţii realizate după o nouă abordare în didactica informaticii: modelarea vizuală a algoritmilor şi evaluarea lor pe baza formelor metaforice. Am evidenţiat impactul procesului didactic implementat şi a metodelor didactice aplicate asupra celor două tipuri de utilizatori: studenţi şi profesori. Studiul s-a realizat comparativ, în două tipuri de lecţii, 2D şi 3D.

Cuvinte cheie

Instruire asistată de calculator, modelare vizuală, proces didactic, gândire algoritmică, limbaj de programare.

Clasificare ACM

K. Computing Milieux, K.3 COMPUTERS AND EDUCATION, K.3.1 Computer Uses in Education, *Computer-assisted instruction (CAI).*

INTRODUCERE

Programarea calculatoarelor a fost întotdeauna considerată de către mulţi autori ca fiind piatra de temelie în procesul de dezvoltare a minţilor tinere [1]. Ca atare, s-a presupus că învăţând să programeze, studenţii vor dezvolta abilităţi de rezolvare a problemelor. Într-adevăr, experienţa în dezvoltarea algoritmilor orientează gândirea individului spre o mai eficace capacitate de problematizare şi de abstractizare, o mai bună analiză a problemelor şi îmbunătăţire în luarea deciziilor. Dar pentru a ajunge în acest punct, există o nouă piedică, poate chiar mult mai costisitoare pentru unii decât beneficiile care le aduce - trecerea de la un statut de utilizator la programator [2].

Lecţiile create în MDL sunt centrate pe student, pe modul în care abordează el noţiunile introductive de programarea calculatoarelor oferindu-i o interfaţă orientată spre modelarea vizuală a algoritmului. Se evită astfel simpla editare de cod sursă, aceasta realizându-se automat. Libertatea de exprimare în generarea soluţiilor se datorează strategiei didactice implementate: în procesul de evaluare a soluţiei se urmăreşte în mare măsură forma finală a modelului descris şi abia la etapa de validare se pune accentul pe paşii intermediari [3].

Sistemul din care face parte modulul prezentat a fost descris în [3] şi [5]. Procesul didactic propus elimină impasul în care se află orice student începător în programarea calculatoarelor - el trebuie sa depăşească două obstacole simultan: să dezvolte o gândire algoritmică şi în acelaşi timp să înveţe să descrie algoritmi într-un limbaj de programare [4].

PROCESUL DIDACTIC

Procesul didactic implică patru etape: transmiterea de noi cunoştiinţe; exemplificări şi analogii; rezolvarea problemelor şi evaluarea soluţiilor propuse; realizarea feedback-ului [5].

Rezolvarea problemelor se face prin interacţiunea a trei componente (Figura 1): modelarea soluţiei, interpretarea acţiunilor intreprinse şi transcrierea lor în instrucţiuni.

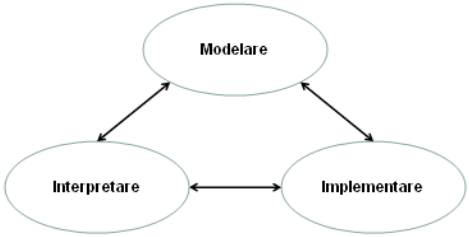

Figura 1 Rezolvarea problemelor

SPAŢIUL DE MODELARE A SOLUŢIILOR

MDL este implementat în principal în Visual C++ [6]. Mediul în care se desfăşoară lecţiile poate fi 2D sau o scenă 3D. Pentru fiecare lecţie, obiectele sunt definite prin atributele lor prin comportament. Pentru a le putea reutiliza în aplicaţii ulterioare, baza de date MDL stochează toate obiectele disponibile. Există astfel posibilitatea editării acţiunilor permise într-o anumită lecţie generând obiecte noi.

Generarea obiectelor

Scena 3D este creată folosind OpenGL [7]. În cazul în care profesorul posedă aptitudinile necesare, el poate crea obiecte 3D într-un mediu de modelare la alegere, iar apoi să le exporte în format .obj [8].

Dacă nu sunt familiarizaţi cu modelarea 3D, profesorii pot încărca în lecţie obiecte predefinite sau pot apela la administratorii MDL pentru crearea de noi obiecte.

Într-un mediu 2D, profesorii pot crea cu uşurinţă obiectele necesare folosind o tabletă grafică, un mediu de desenare sau pot pur şi simplu să preia obiecte predefinite din baza de date MDL.

Atât obiectele create cât şi cele selectate sunt apoi încărcate în lecţie. Următoarul pas presupune ca profesorul să editeze comportamentul lor prin stabilirea

unor acţiuni şi specifice fiecăruia şi de a personaliza modul de interacţiune dintre ele.

4.2 CASE STUDIESSTUDII DE CAZ

Scenă 2D

Lecţia se va modela într-un mediu 2D. Obiectele ce vor construi forma metaforică au fost create de profesor cu ajutorul unei o tablete grafice.

Consideraţiile teoretice studiate sunt definirea şi descrierea structurii alternative. Aplicaţia practică presupune selectarea maximului dintre două numere reale. Forma metaforică utilizată este comparaţia între două cantităţi şi selecţia celei mai grele.

În lecţie se modelează problema determinării maximului a două numere reale.

Procesul are loc, după cum urmează:

1. Studentul vizualizează setul de obiecte disponibile, acţiunile asociate şi modurile de interacţiune premise (Figura 2). Setul este format din robinetul de apă, doi recipienţi şi un cântar. Acţiunile asociate sunt pornirea sau oprirea robinetului; interacţiunile permise sunt deplasările recipienţilor în scenă.

Recipienţii - caracterizaţi printr-un un indicator de nivel numit "Quantity indicator" şi o acţiune de tip "Grab" care permite studentului deplasarea lor în scenă.

Robinetul este manipulat prin intermediul întrerupătorului ON/OFF implicând acţiuni de tip pornit/oprit. La acţionarea butonului de măsurare, cântarul indică greutatea maximă dintre cele două talere ale sale.

2. Studentul modelează soluţia problemei. O variantă corectă este de a umple cei doi recipienţi cu apă, plasarea pe talerele cântarului şi vizualizarea rezultatului procesului de cântărire (Figura 3).

Primul pas este citirea datelor de intrare, adică umplerea celor doi recipienţi cu apă la nivele corespunzătoare celor două numere date. Acest lucru se realizează prin plasarea consecutivă a fiecărui recipient sub robinet şi acţionarea butonului ON/OFF.

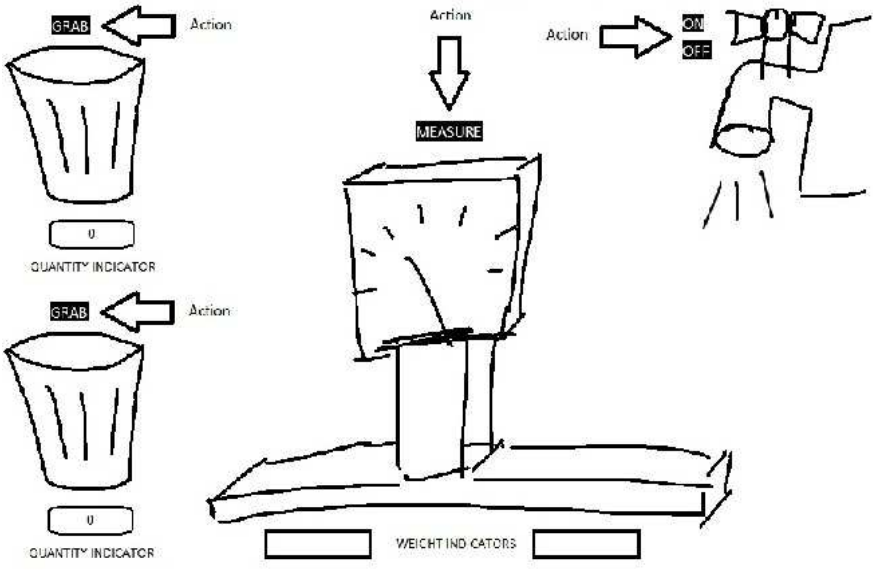

Figura 2 Setul de obiecte 2D date şi acţiunile asociate

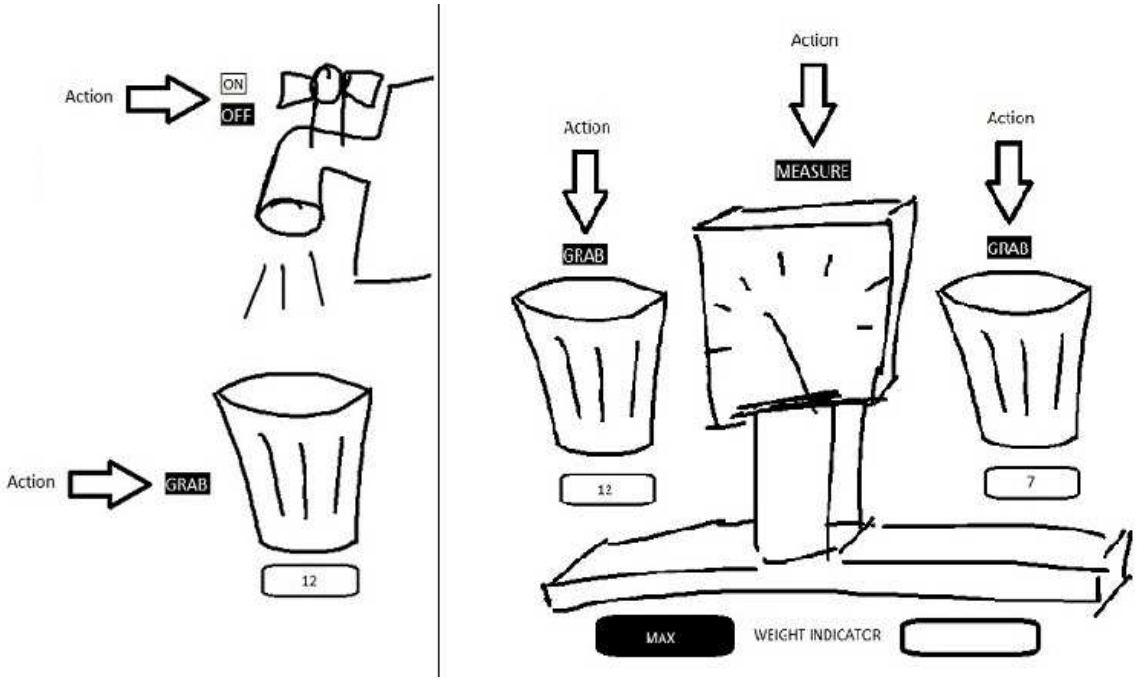

Figura 3 Procesul de modelare a soluţiei şi forma metaforică finală

După ce ambele recipiente au fost umplute corespunzător, studentul le plasează pe talerele cântarului şi acţionează butonul "Măsurare". Astfel îi este indicat rezultatul problemei, cea mai grea dintre cele două găleţi reprezentând maximul dintre cele două numere.

Scenă 3D

Lecţia se va modela într-un mediu 3D. Obiecte ce vor construi forma metaforică au fost create de profesor folosind 3D Studio Max [9] şi încărcate apoi cadrul lecţiei.

Consideraţiile teoretice studiate sunt definirea şi descrierea metodelor de sortare ce pot fi aplicate asupra structurilor de tip tablou unidimensional. Aplicaţia practică presupune sortarea în ordine descrescătoare a elementelor unui vector. Forma metaforică utilizată este aranjarea unui şir de butoaie în ordinea descrescătoare înălţimii lor.

În lecţie se modelează problema sortării descrescătoare a elementelor unui vector. Procesul are loc, după cum urmează:

1. Studentul vizualizează setul de obiecte disponibile, acţiunile asociate şi modurile de interacţiune premise (Figura 4).

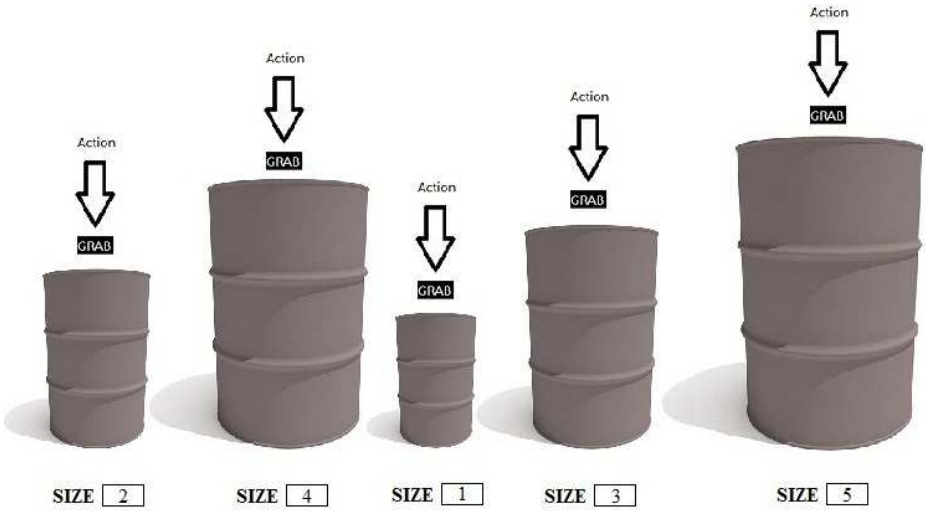

Figura 4 Setul de obiecte 3D date şi acţiunile asociate

Figura 5 Forma metaforică finală

Setul de obiecte este format dintr-un număr constant (5) de butoaie şi un buton de “Validare”. Acţionarea butonului de “Validare” va indica daca s-a realizat sau nu o sortare corectă. Interacţiunile permise sunt deplasări ale butoaielor în scenă.

Fiecare butoi este caracterizat prin înălţime, poziţia sa în scenă relativ la vecinul din stânga şi la cel din dreapta şi o acţiune de tip “Grab” care permite studentului să le deplaseze în scenă.

2. Studentul modelează soluţia problemei. Modelul descris de student trebuie să se potrivească întocmai cu forma metaforică aşteptată ca soluţie, adică fiecare element să se afle într-o poziţie descrescătoare faţă de vecinii săi (Figura 5). Se pot aplica acum diverse metode de sortare studiate într-o parte anterioară a lecţiei şi anume cea de transmitere de noi cunoştiinţe. Există şi posibilitatea de a aranja liber obiectele, studentul aplicând o metodă proprie însă, în acest caz, incidenţa obţinerii soluţiilor false are o creştere semnificativă. Asemenea situaţii au loc atunci când soluţia modelată de către student funcţionează pentru exemplul dat, este validată dar nu poate fi generalizată, nu reprezintă transcrierea unui un algoritm general valabil.

MDL stochează metode de sortare predefinite precum Bubble sort, Insertion sort, Shell sort, Merge sort, Heapsort, Quicksort, Counting Sort, Bucket sort, Radix sort şi Distribution sort [10], [11]. Acest lucru se datorează mai multor factori:

- pentru a preda metodele de mai sus folosind programarea prin exemplu;
- pentru a urma calea modelată de student şi a o confrunta cu soluţii valide;
- pentru a oferi, la cererea studenţilor, indicii spre calea optimă de rezolvare. Aceasta se realizează prin analogia traseului descris de student cu cel mai asemănător traseu existent în baza de date cu soluţii.

CONCLUZII

Sistemul se caracterizează prin uneltele uşor de utilizat pentru profesori în dezvoltarea lecţiilor şi o componentă ce evaluează automat soluţiile date de către studenţi.

Studenţii sunt instruiţi într-un mediu de învăţare uşor accesibil, mediu ce stimulează un mod de gândire algoritmică şi în

acelaşi timp îi fereşte de constrângerile impuse de un limbaj de programare. Codul sursă se obţine într-un mod clar şi uşor de urmărit, prin asocierea acţiunilor modelate cu instrucţiunile unui limbaj de programare.

REFERINŢE

1. M. Gretschel, W. R. Pulleyblank, Mathematics of Operations Research, Volume 11, Issue 4, ISSN:0364-765X, INFORMS, Linthicum, Maryland, USA, 1986, pp. 537-569.
2. Andrew J. Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Joseph Lawrance, Henry Lieberman, Brad Myers, Mary Beth Rosson, Gregg Rothermel, Chris Sca_di, Mary Shaw, Susan Wiedenbeck, The State of the Art in End-User Software Engineering, ACM Computing Surveys (CSUR), Volume 43 Issue 3, ISSN: 0360-0300 EISSN: 1557-7341, New York, USA, April 2011.
3. Daniel Safta, Dorian Gorgan, LDS: Computer-Based Lesson Development System for Teaching Computer Science, Lecture Notes in Computer Science, Volume 6389, HCI in Work and Learning, Life and Leisure, ISBN:3-642-16606-7 978-3-642-16606-8, Springer-Verlag Berlin, Heidelberg, 2010, pp. 228-243
4. Daniel Safta, Dorian Gorgan, Recomandări privind dezvoltarea şi evaluarea soluţiilor de e-learning, Revista Romana de Interactiune Om-Calculator, Vol. 2, ISSN 1843-4460, 2009, pp. 105-110.
5. Daniel Safta, Dorian Gorgan, MDL - Modul de dezvoltare a lecţiilor asistate de calculator, Revista Română de Interacţiune Om-Calculator, Vol. 3, ISSN 1843-4460, 2010, pp. 75-80
6. Visual C++ Developer Center, http://msdn.microsoft.com/en-us/visualc/default.aspx
7. Open Graphics Library - The Industry's Foundation for High Performance Graphics, http://www.opengl.org
8. OBJ Files - A 3D Object Format, http://people.sc.fsu.edu/~burkardt/data/obj/obj.html
9. Autodesk 3ds Max Products, http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=13567426
10. Donald Knuth, The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition, ISBN 0-201-89685-0, Addison-Wesley, 1997.
11. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cli_ord Stein, Introduction to Algorithms, Second Edition, ISBN 0-262-03293-7, MIT Press and McGraw-Hill, 2001.