Informatics in Education, 2007, Vol. 6, No. 2, 269–282
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Design and Implementation of a Logo-based Computer Graphics Course

Pavel BOYTCHEV
Dept. of Information Technologies, Faculty of Mathematics and Informatics, Sofia University
5, James Bouchier, Sofia 1164, Bulgaria
e-mail: boytchev@fmi.uni-sofia.bg

Received: August 2007

Abstract. Two years ago the Faculty of Mathematics and Informatics at Sofia University makes a decision to design a new series of Logo-based courses which make use of the modern technology. The pedagogical component of the challenge is to design a multidisciplinary course suitable for students with different skills and interests. From a development perspective the challenge is to build an entirely new one. And finally the course must be attractive regardless of the seriousness and complexity of the topics included in it.

The paper discusses the structure of the course including the final weeks when topics emerging from students' course projects are taught. Each lesson from the course is based on sets of sample programs representing the general lifecycle of software development. This includes designing, coding and debugging. Samples are created on-the-fly, thus different instances of the course results in different final projects. Lessons are interactive and students may interfere with the direction of demonstrated software development.

Key words: Logo, computer graphics, course, Elica.

Preface

The Logo programming language has been traditionally used in the classroom to describe, to explain and to explore the fundamental principles of Computer Science. The properties of this language make it an advantageous choice for a first programming language. It is not only the simplicity of the syntax that makes it beneficial to students, to their teachers and to the overall learning process. Another factor is the immediate access to drawing functionality via Turtle Graphics. The widespread use of Turtle Graphics is a unique phenomenon which has some negative effect on the public opinion about Logo. Namely, Logo might be considered as just a system for doing graphics with a turtle. The opposite opinion is also still widespread – some turtle graphics environments and libraries are considered to be Logo.

The postponed negative effect of these opinions is that some teachers and parents think of Logo as of a childish language, not appropriate for doing serious stuff. Apparently what is considered as serious, is merely everything which has direct positive impact on entry and exit exam.

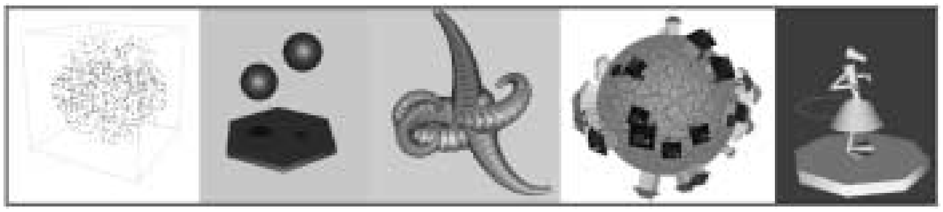

Fig. 1. Snapshots of projects developed through the course.

The Faculty of Mathematics and Informatics at Sofia University, has been using Logo for some decades in various courses – from teacher training courses to e-Learning and Technology Enhanced Learning (TEL) courses (Nikolova and Sendova, 1995). The faculty members were not only using Logo in their classes, but also they were developing new Logo versions. The first one being Plane Geometry System, later renamed to Geomland, released more than 20 years ago, to the latest Elica Logo, which is still under active development (Elica, 2007).

Three lessons from the course are sketched in the paper. The first one is taught in week 4 and spans over Computer Science, Calculus, Analytical Geometry; and Applied Statistics and Probability. The lesson in week 6 is focused on composition of complex movements and their synchronization. It uses elements from Computer Science, Geometry, Physics, and Trigonometrics. The third lesson is about relative transformational geometry and its application in the form of Turtle Graphics. It uses elements from Physics, Robotics, Biology and Art. Snapshots from the projects are shown in Fig. 1.

A few of the students' course projects are also presented in the paper – an animated 3D model of the Solar system, a transformational geometry impression called "United Colours of Elica", and a 3D model of the Faculty building.

The presented Logo-based Computer Graphics course 'conquers' educational territories from the dominating C and C++. It is taught since 2005 and is well accepted by students. They find it both interesting and useful for their education. The future of the course is very promising. A textbook is planned, as well as extension of the teacher-student interaction beyond the time frame of the course.

The Challenge

In the spring of 2005 the Department of Information Technology makes a decision to provide greater support to the development of Logo, as well as to revive its use in the courses. Thus the main challenge is formulated as to *design and implement a new series of Logo-based courses* which make use of the modern technological achievements.

Pedagogical Challenges

The creation of a new stream of courses would have greater academical value if it is adaptable to the specific needs of the teacher and the course. For example, the Logo-based

Computer Graphics course could be taught to Bachelors and to Master's programmes. It could be taught to students in Informatics, Mathematics, Applied mathematics, Mechanics, etc. Many of the students attending the courses are supposed to be familiar with some of the basic concepts of computer science – algorithms, procedural and functional programming, OOP, etc. However, the courses should be accessible to students which have no significant (or in some cases any) experience in these areas either because they are freshmen, or their specialty does not require the full extend of the programming skills. Examples of such students are those who are studying for becoming teachers in Mathematics or Informatics (Vitukhnovskaya, 2005).

The initiative for designing new courses clearly stated that they should cover all previous courses, so the educational plan of the Faculty is not invalidated. Additionally, the courses should provide enough complexity and robustness so that new students from other programs could also join. For example students from the Computer Graphics Master's Programme are also supposed to visit these courses.

Development Challenges

The Logo implementations used by the Faculty in the past are getting more outdated, and the new versions of these implementations are not available due to various reasons. This poses the challenge what programming environment to use for the courses (Laucius, 2006). It is decided to use Elica Logo as a software backbone for the course and to design new courses based on the advanced features of this Logo dialect (Boytchev, 2005). Switching to a new Logo dialect with a wider but different range of features makes most of the already existing teaching materials obsolete. The design of new courseware would lead to major changes in the lessons' content and will affect the actual teaching procedures. This poses a unique challenge as to how to design and implement new courses in a way that the overall advantage is much bigger than the overall disadvantage.

Psychological Challenges

Charm is one of the most undervalued factors for modern courses. Teaching Computer Science and Computer Graphics at university level is treated as a serious endeavor. The matter is heavy, sophisticated and knotty. Whether it is attractive to students or not is not as important as the content of the course. The new courses which are to be developed need this charm in order to win students' hearts. Initially the courses start as selective, so it is important to design the course in such a way that students are willing to enroll not only to gain some credits, but to learn something useful. The so called Nintendo generation (West, 1995) poses new requirements for courses, especially those related to computer graphics. Providing adequate level of charm is essential factor to meet these requirements.

Course Structure

A typical course spans over 15 academic weeks and includes 30 lecture hours plus 30–60 lab hours. Traditionally students are evaluated several times during a course and eventually they have an exam, which comprises the biggest part of their final score. The final exam usually has two components – practical and theoretical.

The Logo-based courses could follow the same settled structure, but instead, it is decided that the way to measure student's efforts, knowledge and imagination is to provoke their creativity in the creation of a Logo-based course projects. As a result there is no examination synopsis of Elica, and students are focused on their projects long before the examination session.

Introductory Section

The course is divided into three sections – see Fig. 2. The first section takes three weeks (i.e., 6 academic hours). It is dedicated to the introduction of Logo and Elica. During the first week, while students get accustomed to Elica, they learn the basics of the Logo language, including data types, all reserved words and some functions for words and lists processing.

The second week is left for iteration, recursion and custom operators. The third introductory week is dedicated to OOP features of Elica. Students learn various methods for class definition, object using and manipulation of OOP entities.

The Core Section

The core part of the course takes roughly 8 weeks and covers topics directly related to computer graphics, 3D modeling and animation. The lessons also utilize implicitly knowledge from the conventional computer science subjects, like algorithms and optimizations. It is not possible to make a precise topic-week relationship, because the course is continuously adapting itself to the level of the students. Follows a list of the basic topics in the core section:

- 1D/2D objects: points, lines, segments, rays, circles, ellipses, squares, rectangles.
- 3D objects: cubes, parallelograms, spheres, cones, cylinders, spline surfaces.
- 3D spaces: coordinate systems, transformations (translation, scaling, rotation).
- Custom user-defined objects, custom methods for object rendering, support for nested graphical objects, local transformations.
- Color, RGB color space, lighting, object materials, fog effects, textures.

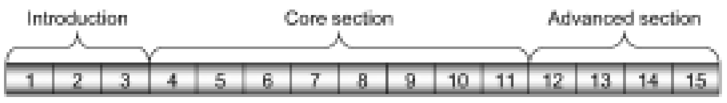

Fig. 2. Course sections – weekly structure.

- Turtle graphics in 3D space, hierarchical joint models, creating complex android models.
- View points, flying through space, perspective and orthogonal projections.
- Animation, smooth movement, techniques for achieving realism – resistance, inertia. Simulation of physical movements, composition of various movements.
- Building graphical user interfaces, event handlers, capturing mouse activity, responding to user interaction, building interactive environments.

Advanced Section

The last few weeks of the course are left for topics which are not taught during the basic course, but are related to the course projects selected by the students. During the second half of the core section students determine the topic of their projects. If the projects use specific objects or techniques, then they are included in the advanced section.

As a result of this layout, the advanced sections for different semesters are always different. They depend entirely on what students chose to do. Here are some examples of advanced topics:

- simulation of water waves,
- run-time modification of classes and objects,
- building Logo libraries and software packages.

The Exam

About the middle of the course, weeks 8 to 10, students are encouraged to propose their projects. If accepted they can start working on them immediately. Some proposals might not be accepted either because they are too complex or are too easy. Complex proposal are simplified to a level which will make them doable in a reasonable period of time. Projects that are too simple or too easy to implement in Elica are loaded with some extra features.

Students may request additional information about issues related to their projects. In this way they determine the topics for the third section of the course (Stoyanova, 1999). When they think they are ready with the projects they submit them electronically for evaluation. If submission is done in advance, students get reply with the current rating and suggestions how to improve it. If students have time and are willing to get a higher rating they can adjust their projects and resubmit them. This project-evaluation scheme is repeated several times.

For the last 4 semesters this scheme proves to be successful for students. Most students resubmit their projects twice. Very few of them send three or four versions. The possibility to have their projects reviewed and resubmitted makes students more comfortable with the course and lets them display their creativity and imagination.

Class Structure

The structure of each individual class is a small copy of the whole course structure. It starts with an easy to understand basic concepts, and then it goes to details and advanced ideas, and ends with explanations-responses to students' questions.

Classes are extremely interactive. The topic is presented as series of programs which are typed and executed in real time. The image from the teacher's computer is projected on a big screen and students can follow the process of programming from the beginning to the end.

Usually the first sample program is just few lines long. Next samples just add some features to it thus building more complex programs. One of the co-ideas of the course is to demonstrate the process of real-world programming. The selected technique of erecting a full-functional program from scratch is a perfect live scenario for students (Dagienė, 1999).

One of the most enjoyable by the students moments is when the program does not behave correctly. Such situations are handled by on-the-fly debugging and modification of the code. The teacher's thinking is vocalized and projected on the screen without interruptions, so students learn various techniques to resolve bugs.

Quite often students are asked to provide ideas how to solve a problem. Most of the cases they suggest interesting solutions which are immediately tested. There are also cases when the suggestions do not solve the problem. Such ideas are also tested, because they are an excellent opportunity to practice important problem-solving and pitfall-avoiding skills.

All samples during each class are archived and provided to the students, so they can replay the whole lesson, and do further explorations with the code. Some are getting the sources right after the lesson; others are downloading them from an online repository (Elica Repository, 2007). At the beginning of the course many students use a paper notebook where they try to copy by hand all sample programs. However they soon find that it is impossible to cope with the dynamics of a live coding. The sample programs are constantly changing while testing different ideas and debugging, so students realize that it is much more valuable to grasp the ideas and the techniques, rather then to memorize the exact programs.

The selected structure of the classes is well accepted by students. However, it imposes additional requirements for the teacher. The most obvious one is that it is unavoidable to write only correct programs. A single class may have 20 to 30 program samples, and each of them is a potentially dangerous place for the teacher to make a mistake. Some of the mistakes are intentional, but others are not. The latter makes teaching very demanding, because the teacher should extract educational value not only from good examples, but also from bad. On the other hand unintentional errors place the teacher in a stress situation, because s/he has to provide a solution with an adequate reasoning in a limited time frame.

Multidisciplinarity

Each lesson from the course has a main topic about features and techniques from the area of programming with Elica. Additionally, each lesson is enriched with many smaller fragments from other subjects, mainly from Mathematics and Physics. Some lessons 'rent' ideas from Biology, Psychology, Astronomy, Geography, Statistics and even Arts. Multidisciplinarity[1] is an intentional feature of the course which makes classes more interesting and demonstrates the application of software in various scientific and artistic domains (Sendova, 2006). To demonstrate the multidisciplinarity of the lessons, three of them are briefly described below.

Points (*week 4*)

The first graphical lesson is scheduled for week 4 of the course. It starts with the introduction of the most primitive graphical object in Elica – the point. A point is defined by 3 numbers which stand for its coordinates. The initial examples used in this lesson are used to describe how the graphical system is initialized, how to visualize the coordinate system, and how to create a point at some position. During the lesson the students face a short challenge – to create 1000 random points within a virtual cube. The second challenge is more complex: to create the points in a sphere, not in a cube – Fig. 3. Some of the students do not know how to do this, others suggest using the formula of a sphere $x^2 + y^2 + z^2 = r^2$.

The most often suggested solution is to create random points in a cube, but keep only those which are internal to the sphere. This is one of the milestones in the lesson, because students realize that the created points are less than 1000.

The lesson continues in the direction of how to fix this. Several solutions are suggested and tested. One of them is to keep track of the number of internal points and terminate the loop when 1000 is reached. Another solution is to check for validity before the creation of a point. Throughout a series of tests and modifications students learn to be doubtful about a program that looks correct.

Once they calm down thinking they've done all possible solutions, they are asked to find a way to directly create a point inside the sphere without the need to test for inclusion. Some students try to give up, others test teacher's sense of humor by making all the points at (0, 0, 0). With help from the teacher students find two more solutions – one based on Cartesian coordinates, and another based on polar coordinates.

At the end of the lesson students are directed back to the wrong solution of 1000 points in a cube some of which are also in a sphere. This wrong solution serves as a basis to solve a new problem – how to find the approximate value of π exploiting the bug in the program. It takes some time to conclude that the number of points in the sphere and in the cube depends on the ratio of the sphere's and cube's volumes which is $\pi/6$. Table 1

[1]Although the course uses elements from other subjects the main focus is always on the graphics with Logo. Topics from non-programming subjects are used mainly to give ideas for projects and to illustrate the interconnection of sciences.

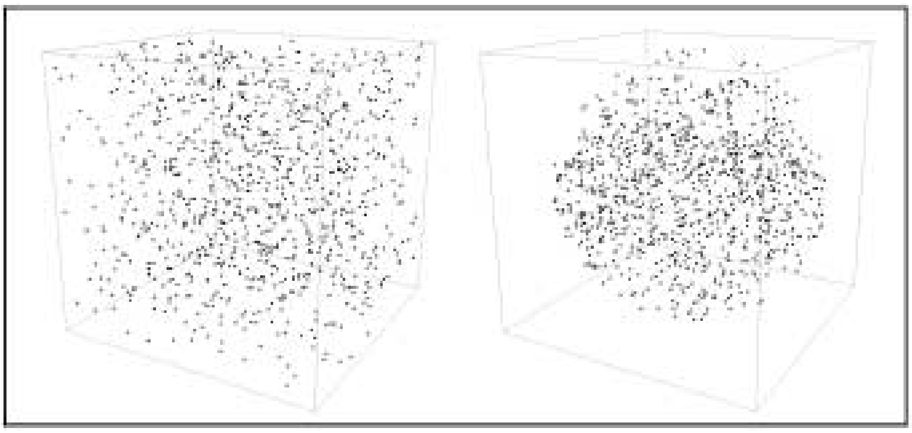

Fig. 3. Random points in a cube and in a sphere.

Table 1

Calculated approximation of π

	10	**100**	**1000**	**10000**
Test 1	1.8	3.12	3.22	3.15
Test 2	3.0	3.66	3.05	3.16
Test 3	2.4	2.82	3.23	3.09

shows the approximate values for π calculated for cases with 10, 100, 1000, and 10000 points.

This first graphical lesson spans over a few subjects studied in the faculty. Students learn some basic concepts from Computer Science (various loops, conditional execution, counters), Calculus (function composition), Analytical Geometry (Cartesian and polar coordinates, equations for cubes and spheres, volumes); and finally – Applied Statistics and Probability.

Bouncing Balls (*week 6*)

The bouncing balls lesson is taught at week 6. It is the first time students learn how to simulate physical properties of objects through their movement. The final goal of the lesson is to show how to make two balls bouncing on a flexible plate – Fig. 4. The balls have different bounce periods. The plate starts to vibrate whenever it is hit by any of the balls. The vibration gradually fades if no hit is encountered for some time. Both balls have shadows cased on the plate. While balls go away from the plate, shadows become smaller and lighter.

The series of programs in this lesson starts with the simple case of a ball going linearly up and down. Students realize immediately that such movement is not visually acceptable because it is physically incorrect. The next step is to replace the linear movement with

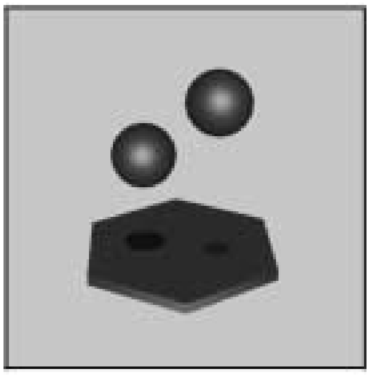

Fig. 4. Bouncing balls.

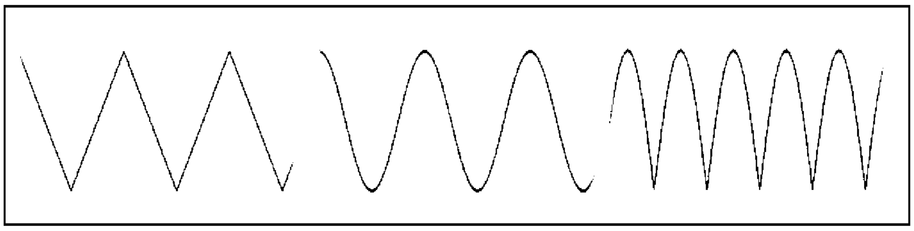

Fig. 5. Linear, sine and absolute-sine bouncing.

sine function[2]. Now the movement of the balls near their highest positions is acceptable, but they do not bounce off sharply from the plate. This gives the clue to use the absolute value of sine – Fig. 5.

By using $|\sin|$ students learn that very often physical movements in an animation are not based on the physically correct formulae. Because of performance issues some calculations are replaced by faster one. That is why $|\sin|$ can effectively replace the calculation of a parabola.

Sine is heavily used throughout the whole lesson. It is ‘responsible’ not only for the bounce, but for another 3 actions – the vibration of the plate based on variation of $\sin(x)/x$, the rotation of the plate which smoothly alternates between clockwise and counter-clockwise, and the flying of the view point around the scene.

An interesting problem with the bouncing balls is the synchronization of various movements. The bouncing of the balls is independent on the vibration of the plate. The program must explicitly synchronize both motions in order to trick the viewer to think that vibration is caused by the hit.

The “Bouncing balls” lesson is another example of a multidisciplinarity. It practices skills in several subjects – Computer Science, Geometry, Physics, and Trigonometrics.

[2]By describing the properties of the bouncing the students are asked to think of a function with similar properties. Some students come up with the idea of sine or cosine quite quickly, others need more hints.

Relative Transformational Geometry and Turtle Graphics (*week 8 and 9*)

Transformational Geometry deals with affine transformation of objects. It is heavily used in many Elica lessons, because the original shapes of all objects are the canonical solids (like cube $1 \times 1 \times 1$, or sphere with radius 1). All other objects are generated by transforming the canonicals. Relativity in geometry is when each object has its local coordinate system, and other objects bound to it are defined in terms of its local system.

The usage of relative transformational geometry is based on transformational matrices. One interesting application is how to stack transformations for chained objects. Because of the flexible structure of each lesson, the same lesson taught in two different years are different. The main topic is the same, but the set of samples and especially the final programs becomes quite different. For example, “Dandelions” from Fig. 6 are the final program of the lesson from December 2005, while “Octopus” is from the same lesson two semesters later.

“Octopus” gives birth to other program which is included in the Online Elica Museum (Elica Museum, 2007). The third image in Fig. 6 is a snapshot from “Larnaean Pentapus” exhibit from the museum. Using Relative Transformational Geometry prepares students to accept easily the benefits of Turtle Graphics. Relativity is the nature of turtle’s motion and thus it has common grounds with Differential Geometry.

The first 5–10 programs in the lesson are used for introduction to 2D and 3D turtle graphics. A typical example of this introduction is to make a turtle crawl on the surface of a cube and build various objects on each face, or to crawl on a sphere – see Fig. 7.

The more advanced usage of Turtle Graphics is to create the objects drawn earlier with relative transformational geometry, but this time using turtles. Students see how the reimplementation becomes shorter and easier to understand, because turtles and relative transformations have a common mathematical background. The rest of the lesson is dedicated to step-by-step building and animation of a humanoid body – legs, hands, torso, and head, traversed by an invisible turtle. The animation is done by synchronous changes in the joints – every joint has a local coordinate system and is a place where the body parts have some level of motion freedom.

Most often the created body is of a dancer playing with a hoop – Fig. 8. The right-most image is from the “Circus Dancer” exhibition form the Online Elica Museum.

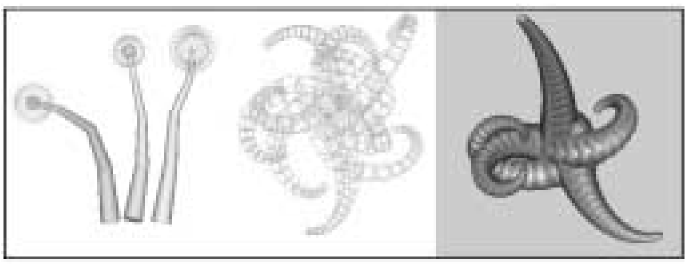

Fig. 6. “Dandelions” (left), “Octopus” (middle), and “Lernaean Pentapus” (right).

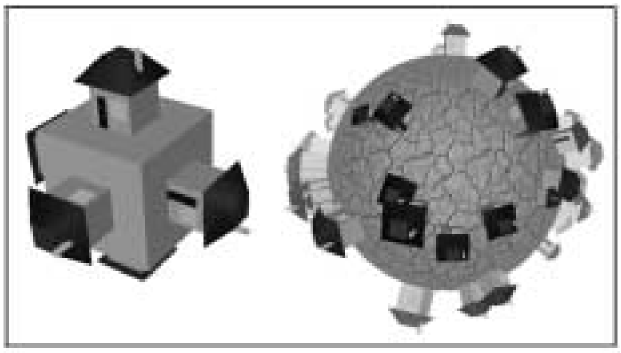
Fig. 7. Houses created by turtle crawling on the surface of a cube and a sphere.

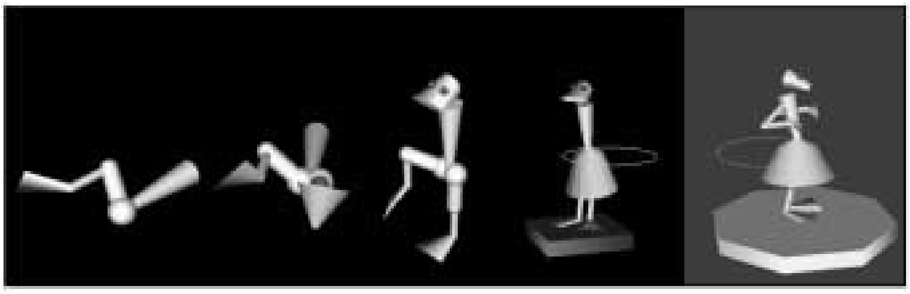
Fig. 8. Four phases of humanoid building (left) and the “Circus Dancer” (right).

This lesson teaches students how to program more complex systems. It makes use of various elements from Physics, Robotics, Biology, and Arts.

Course Projects Show-Cases

The Online Elica Museum is a section from the Elica site containing a collection of Elica programs, mostly related to animation of 3D objects. The programs are freely available as source code for everyone. Some of the course projects developed by students are so successful that they are included in the museum as independent exhibits. Of course, the original source code is polished to make it more representative and clearer, so that other students can learn from it. This section of the paper presents snapshots of several students’ course projects – some are published in the museum, others are not.

The project “Solar System” creates the Sun and the planets as 3D objects, and then animates the system by rotating the planets around the Sun – Fig. 9. Each planet is dressed in a texture taken from real astronomical images. Some details are added to make the animation more realistic – there are stars at the background, the planets spin around their axes too, Saturn and Uranus have semitransparent rings, and the Moon rotates around the Earth. The making of this project requires understanding of how textures are used and

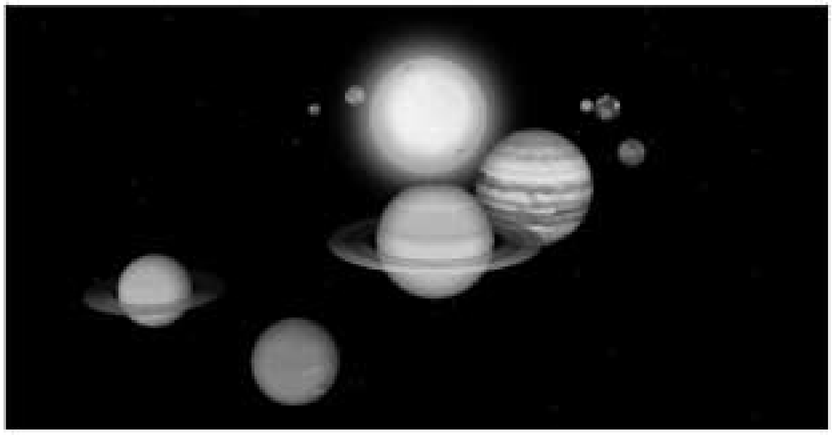

Fig. 9. Project “Solar System”.

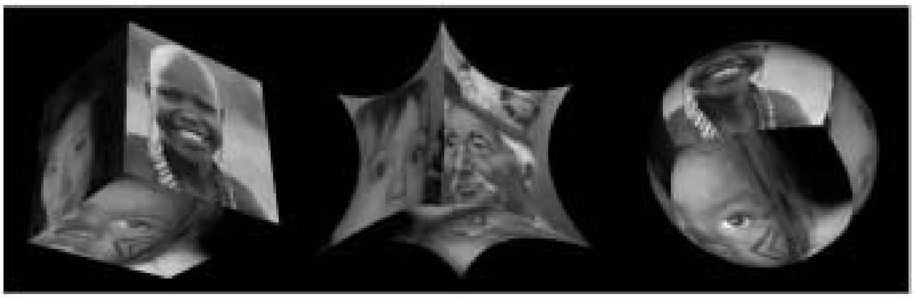

Fig. 10. Project “United Colours of Elica”.

some math skill to define the orbits and spinning of the planets as well as composition of several non-linear movements.

The second project – see Fig. 10 – is named by the student who wrote it “United Colours of Elica”. It features a series of animations of cube transformation. Each face of the cube is implemented as a spline surface which control points are defined by an invisible 3D turtle.

Another student’s project is to make a virtual 3D building of the Faculty of Mathematics and Informatics. A snapshot of this building is shown in Fig. 11.

Conclusion

Logo is used in primary and secondary school curricula, but it can be used in various courses at a university level too. This applies not only for light-weight courses, but also to traditionally heavy subjects like Computer Graphics where C and C++ are dominant programming languages. The course described here has been active for 4 semesters. The number of enrolled students is getting bigger, popularity rises. Anonymous questionnaires

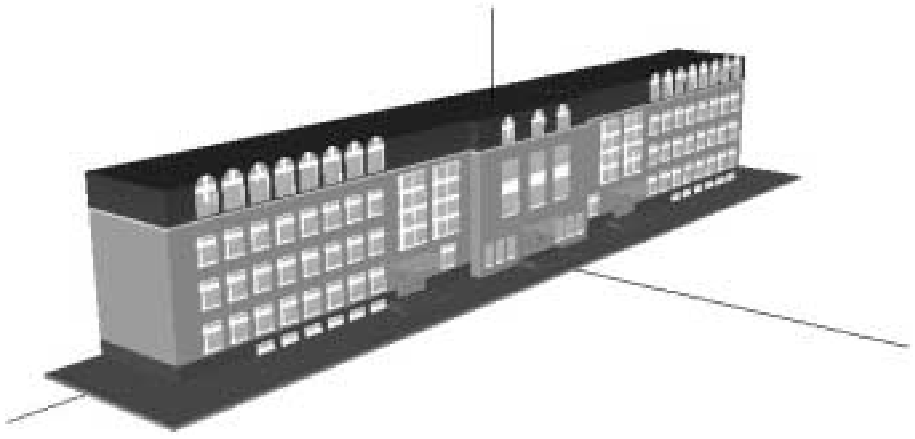

Fig. 11. Project “The Building of Faculty of Mathematics and Informatics”.

show that students definitely prefer interactive lessons. One of the most liked features is the life demonstration of programming which includes both coding and debugging. The project-based scenario of each lesson serves as a prototype of the lifecycle of a typical software development.

Another feature of the course admired by students is the possibility to unfold their imagination by working on a course project related to their personal interests. Multidisciplinarity builds bridges between programming and the highly varies skills of students – although they come from different specialities, they learn something new and interesting. The way how the topics in the course are designed, especially the custom-defined ones during the last few weeks, is a thing which helps student build awareness of and confidence in their own skills. This is due to the fact the students feel the course as if it is personally oriented towards them.

The future of the Logo-based course is very promising. There are several ideas for its further development. One of them is the writing of a textbook, which is a challenge of its own, because of the dynamism of the course topics. The first step is to collect enough variations for each topic.

Another idea is to move the Elica Repository to a more interactive online system – Moodle. This will add many new possibilities for student-teacher off-lesson interaction.

Since January 1, 2007, Bulgaria joined European Union. This had a significant impact on the educational system. Many Universities are now revising their Programmes because of expected increase of foreign students. One of the possible future plans of the Logo-based Computer Graphics course is to offer it to English-speaking audience.

References

Boytchev, P. (2005). Using Logo to model and animate. In Gr. Gregorczyk *et al.* (Eds.), *Proceedings of Eurologo 2005*, August, pp. 66–75.

Dagienė, V. (1999). Logo and changes in learning: project-based methodology. In R. Nikolov *et al.* (Eds.), *Proceedings of Eurologo 1999*, August, pp. 179–186.

Elica (2007). *Elica Home Page*, `http://www.elica.net`

Elica Museum (2007). *Elica Online Museum*. `http://www.elica.net/site/museum/museum.html`

Elica Repository (2007). *Elica Samples Repository*, `http://www.fmi.uni-sofia.bg/Members/boytchev/elika/`

Laucius, R. (2006). Issues of selecting a programming environment or a programming curriculum in general education. In *ISSEP 2006, LNCS 4226*. Springer-Verlag, Berlin, Heidelberg, pp. 169–178.

Nikolova, I., and E. Sendova (1995). Logo in the curriculum for future teachers: a project-based approach. In M. Dúill (Ed.), *Proceedings of Eurologo 1995*, July, pp. 7–12.

Sendova, E. (2006). Handling the diversity of learners' interests by putting informatics content in various contexts. In *ISSEP 2006, LNCS 4226*. Springer-Verlag, Berlin, Heidelberg, pp. 71–82.

Stoyanova, N. (1999). The students – authors of tasks. In R. Nikolov *et al.* (Eds.), *Proceedings of Eurologo 1999*, August, pp. 334–339.

Vitukhnovskaya, A. (2005). Logo for the would-be teachers of the computer science elementary course. In Gr. Gregorczyk *et al.* (Eds.), *Proceedings of Eurologo 2005*, August, pp. 245–256.

West, I. (1995). Logo: forward 1 to freedom. In M. Dúill (Ed.), *Proceedings of Eurologo 1995*, July, pp. 87–92.

P. Boytchev is an associated professor at Faculty of Mathematics and Informatics, Sofia University. Author of Elica and a developer of educational software. Designs and teaches Logo-based courses "Computer Graphics", "Educational Languages and Environments" and "System and Environments for Electronic Education".

Logo kompiuterinės grafikos kurso planavimas ir diegimas

Pavel BOYTCHEV

Prieš dvejus metus Sofijos universiteto Matematikos ir informatikos fakultete buvo priimtas sprendimas suprojektuoti naujus Logo kursus, kurie turėtų naudos mokant šiuolaikinių technologijų. Pedagoginė šios naujovės dalis – parengti kursą, kuris tiktų studentams, turintiems skirtingus įgūdžius ir interesus. Jis turėtų būti visiškai naujoviškas požiūriui nukreiptas į ateitį. Pagaliau, šis kursas turėtų būti patrauklus nepriklausomai nuo jį sudarančių temų (dalykų) rimtumo ir sudėtingumo. Straipsnis aprašo ir pagrindžia kurso struktūrą. Kiekviena kurso paskaita pagrįsta daugybe modelinių programų, vaizduojančių bendrus programavimo pasiekimus. Tai apima ir projektavimą, ir programų kūrimą. Pavyzdžiai (modeliai) kuriami gana sparčiai, tokiu būdu gaunami skirtingų baigiamųjų projektų ir skirtingų pavyzdžių (modelių) rezultatai. Paskaitos interaktyvios ir vaizdžios, todėl studentai gali susipažinti su demonstruojamu programiniu produktu.

Straipsnyje nuosekliai aprašomos trys kurso paskaitos. Pirmoje paskaitoje per 4 savaites mokoma informatikos, integralinio ir diferencialinio skaičiavimo; analitinės geometrijos bei taikomosios statistikos ir tikimybių. Antra paskaita apima sudėtingų judesių sudarymą ir sinchronizaciją. Naudojami elementai iš informatikos, geometrijos, fizikos ir trigonometrijos. Trečia paskaita yra apie reliatyviąją transformacinę geometriją ir jos taikymą Vėžlio grafikai. Yra naudojami elementai iš fizikos, biologijos, dailės. Straipsnyje taip pat yra parodomi keletas studentų kurso projektų: Soliario sistemos animuotas erdvinis modelis, transformacinės geometrijos pristatymas, pavadintas "Jungtinės Elikos spalvos" ir fakulteto pastato 3D modelis. Pristatytas Logo kompiuterinės grafikos kursas nurungia mokomąsias teritorijas iš dominuojančių C ir C++. Jo mokoma nuo 2005 metų ir šis kursas labai patinka studentams. Jie mano, kad jis yra įdomus ir kartu naudingas jų pačių išsilavinimui.