Creating and Employing On-line Dynamic Learning Objects for an Introductory Programming Module

Poppy Pickard*, Peter Chalk** and Ray Jones**

*Department of Computing and Electronic Technology, Bolton Institute of Higher Education, United Kingdom
**Department of Computing, Communications and Mathematics, London Metropolitan University, United Kingdom

The paper shows how learning objects can be designed to promote a constructivist learning environment whilst maintaining flexibility for reuse and repurposing. These learning objects, which support an introductory Java programming module, are employed across two UK HE institutions that both use the same virtual learning environment (WebCT) to link to these objects, together with a graphic software library creating a rich and varied learning environment. Collaboration was ongoing during the parallel process of development and delivery. The first semester is evaluated, and suggestions are made for future work.

Keywords: learning objects, pedagogy, VLE, programming, Java, repurposing, visualisation.

1. Introduction

Succesful learning takes place when learners are able to construct their own meaning and understanding from a rich and varied pallette of learning materials. Creating such an environment can be costly for a single author, particularly if the learning materials are single use. This paper tracks the development and use of freestanding and reusable learning objects for the teaching of an introductory Java programming module.

The context in which this development took place was within two UK institutions, London Metropolitan University and Bolton Institute of Higher Education. UK universities are increasingly dealing with a larger and more diverse study body, particularly in popular disciplines like computing. This in turn has resulted in a growing national concern regarding the teaching of introductory programming. An increasing number of students claim to find programming difficult and try to avoid it during later stages of their degree course. Even universities with students with higher entrance qualifications than the two in this study, report on this trend. [6]

In response to this problem a study of first year programming was undertaken at London Metropolitan University (formerly University of North London) between 2001-2002. The study, which recommeded pedagogic principles such as congnitive apprenticeship and a spiral curriculum for confidence building, was a catalyst for the development of the learning objects and graphic software library described in this paper.

A development team was formed in spring 2002 to take this forward. One of the team subsequently moved to Bolton Institute of Higher Education which enabled the planting of the work in a second institution at developement stage. After an exciting first semester of using these new learning materials, the project is showing very promising results.

2. Pedagogic Context

Starting to program involves learning the language's formal and detailed grammar, with all its associated meanings and also learning to operate within an environment where program editing, compilation and execution are managed. Both the language itself and the programming environment can present their own

difficulties, which can mean that when errors occur the learner often cannot "see" what is happening. This can give rise to a lowering of confidence, in particular for mature learners, who make up a considerable part of our student body.

The project sought to address these issues by providing a rich and supportive environment in which visualisation was a key factor. [9]

The learning environment would use a cognitive apprenticeship model, where students would be encouraged to develop the cognitive skills necessary for programming in Java, whilst being encouraged to extend their knowledge through exploration and investigation. [3]

The structural aspects of the Java language would form the subject matter for learning objects which would be visually rich using animated explanations, simulations and quizzes giving immediate feedback. Programming practice would use a complementary graphic software library that would scaffold students in the initial stages of learning by hiding some of the complexity of Java and giving visual feedback. Even using "public static void main" can be difficult for beginners.

The order of teaching was designed so as not to overload students with too many variants of essentially the same topic. For example many programming courses teach all three types of iteration or loops at once, and introduce all the data types at once, which can cause confusion for the novice programmer. The aim of this design was to provide a spiral curriculum where topics are introduced and then revisited to give additional variations or deeper meanings. [7]

3. Learning Object Design

Much work has been emerging over recent years on standardization and packaging of learning objects by organisations like the IMS and IEEE. This work has no doubt influenced and accelerated the development of e-learning materials, fostering the important precepts of reusability, extensibility, accessibility, flexibility, interoperability.

The IEEE standardization draft defined learning objects as:

"a learning object is defined as any entity, digital or non-digital, that may be used for learning, education or training." IEEE [5]

This is a very broad definition and is pedagogically neutral. Thus there is no reference to the size, scope or authoring of any learning object. Boyle suggests:

"there is a marked limit to the productive reuse and repurposing of learning objects that have not been designed for these purposes in the first place." [2]

From a theoretical perspective Boyle [2] goes on to argue for a design that synthesizes software engineering and pedagogic principles. Good commercial software from its inception is designed to be maintainable, often achieved through modularity. A learning object needs to be cohesive, i.e. to do one thing and one thing only. Thus a learning object could be mapped to a learning outcome or some clearly stated learning goal. There should be minimal binding or interdependency between learning objects, i.e. the educator should be able to use them in any desired order. Boyle calls this the principle of 'decoupling', which is critical for re-use.

In order to provide pedagogical richness a compound learning object was conceived. The compound learning object is presented through a single web page, Figure 1, featuring a header for the title; a main body where a complete and succinct exposition and example of the learning content is given. This could itself be seen as a learning object and is indeed referred to as a text learning aid later in this paper. The right-hand

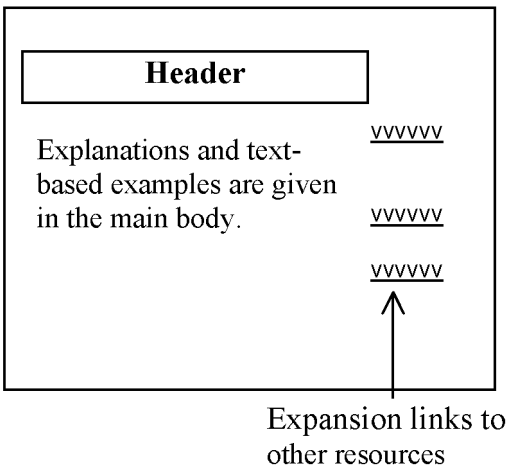

Fig. 1. Schematic layout for a compound learning object.

column is used for autonomous links, which offer a variety of independent extension activities, e.g. visualizations using Flash animations, further examples, quizzes, etc.

The compound learning object layout is very simple, the URL expansion links in the right hand column are purposely separated from the main text to give the level of modularity that will ease maintenance and future repurposing providing a second level of decoupling.

3.1. Development Work

Development of compound learning objects started around July 2002, the academics were responsible for the design of each compound learning object and they worked closely with a developer who created the main page of each compound learning object to a standard format taking into account all the recent accessibility legislation. [10] A multimedia developer was responsible for the Flash animations. These compound learning objects underwent a spiral development process. Through discussions between the academics and developers various improvements and standardisations were brought into use. More than half of the learning objects were in place when delivery of the new Java modules started in autumn 2002. Later learning objects were authored and developed during the semester with almost daily conversations between team members in London Metropolitan University and Bolton Institute.

3.2. Example Compound Learning Object

To date, 50 learning objects have been developed. Figure 2. shows part of a link from a compound learning object about While Loops.

This is a page from an animated description of a Java while loop, which shows a submarine moving down through the water as its position, is changed.

The learning objects are complemented by a graphic software library, which is used in the programming environment.

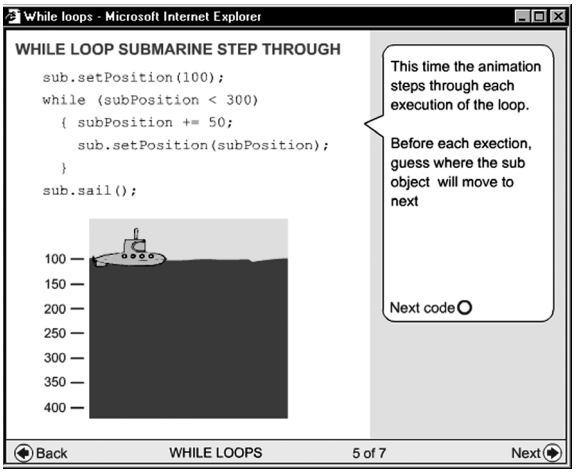

Fig. 2. Learning object animated explanation.

4. The Graphic Software Library

The graphic library was written to complement the pedagogy of the project. Students receiving a visual output from a program could construct their own interpretation of this output and feel motivated to use different inputs to test that understanding.

The graphic library was designed around two packages. One package provided classes for primitive shapes such as rectangles, ovals and lines, as well as a die, which could be rolled to give a value from 1 to 6, and a text output facility for writing on the output window. The second package of utilities included an input box and a timer. Java was taught from an object's first approach where the students would first learn to create and use objects from pre-written classes. Although an object's first approach is the subject of much debate, it has been implemented successfully in many contexts. [1, 4] It can also be argued that from a standing start as a programmer, it is easier to learn the objects paradigm from the outset, rather than meet it after having learnt to program procedurally. The graphical screen objects also provided a perfect metaphor for Java objects making it a natural route to follow.

Students are provided with a summary of classes and methods available in each class, which they are encouraged to use freely in the early weeks of the course. This then sets the scene for writing their own classes and methods for use in aggregation and then inheritance. The classes

of the graphic library are all visible to the students so they can at any time explore the code in these classes and see how, for example, inheritance has been used to simplify the writing of classes.

The familiar programming constructs such as iteration are used to animate the graphic objects or a decision could be made on the result of 'rolling' a die object.

Students were often invited to 'create' their own graphic pictures from the classes available to them. For example in Figure 3 one student used loops to create a face with moving eyes and an opening and closing mouth, from a simple filled and unfilled oval. Animations such as these are fun and motivating.

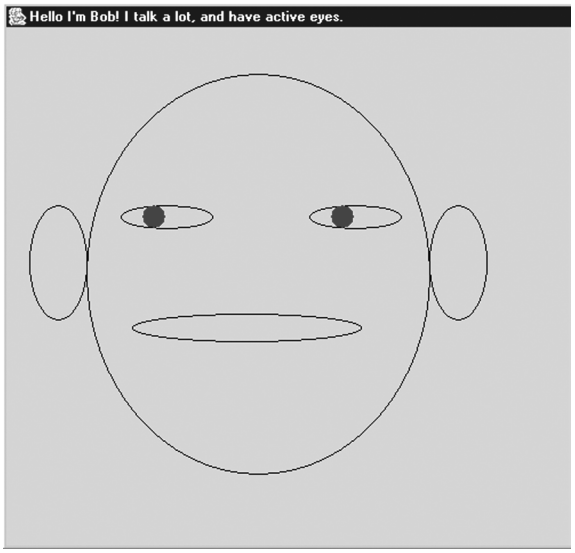

Fig. 3. Graphic picture.

The Integrated Development Environment (IDE) JCreatorLE was chosen for use with the software library. It is produced by Xinox Software and is freely downloadable. It is reasonably easy to install on top of Sun Microsystem's Java 2 Software development Kit and it also allows the creation of application templates which were utilised for holding the graphic software library.

5. Employment of Learning Objects

The learning objects were made available to students through a virtual learning environment (VLE). A VLE is a piece of software that can manage a student group, deliver learning content, provide assessment, discussion and mail facilities. The one used for this project was WebCT [11]. London Metropolitan University had many years experience of using WebCT, Bolton Institute of Higher Education was in its first year of using this VLE. One advantage of using WebCT, apart from its commonality between the two institutions, was the tracking facility it offered. It is possible to collect data for use of a single page, in this case a learning object.

Besides linking to the compound learning objects the VLE contained lecture slides, weekly practical programming activities based around the learning aids, sample Java code, feedback from weekly practicals.

The testing facility of WebCT was used for surveys and multiple choice quizzes which formed part of the assessment. Writing of multiple choice questions was shared between the two institutions, meaning a sufficiently large database of questions was created to allow each question to be chosen randomly from a set of questions on a particular topic.

A selected programming exercise was uploaded to the WebCT assignment page each week for marking and feedback. Each of these programming exercises could earn 1final module mark, which proved very motivating. Supervised timed programming exercises were given on two occasions as part of the more formal assessment; again these were uploaded and marked within WebCT. Each student therefore had access to his or her assessment results to date within the VLE. Students also made good use of the discussion and e-mail provided by WebCT there was some useful sharing of programming tips as well as the inevitable queries to the module tutor. There was very much a sense in which the students became a community of learners within the VLE. [12]

Although the learning aids were associated with particular programming exercises by placing links to the learning objects on just one page, this encouraged students to dip in and out of these at will, whilst still having the structure of a well scaffolded spiral curriculum. Students made good use of this facility and would often revisit a learning object to check the Java syntax before using it in a current program.

6. Evaluation

An evaluation framework for the first semester was managed by a researcher who was part of the design team. Students responded to three questionnaires, start, mid and end of semester, together with individual interviews conducted several weeks into the course. WebCT provided page-tracking statistics to monitor the use of learning objects.

The in-course assessment and examination provided further basis for comparison together with results from previous cohorts taking the same subject.

The mid semester questionnaire (Figure 4.) gives a reflection of student opinion half way through the module. It is surprising how closely the Bolton and London opinions match despite the difference in student bodies: London Metropolitan University with 600 students studying for HND, BSc and MSc again on a variety of computing courses and Bolton with 120 students studying for a BSc in a range of computing disciplines such as Computer Games Software Development. The age profile of the Bolton students is lower than that of London students, the cultural diversity at Bolton is narrower than London and it is also only 5% female compared to 25% in London.

How useful are:	Very useful	Useful	Not very useful	Use less
Lectures?	37%	48%	13%	2%
	31%	*50%*	*16%*	*3%*
Lab exercises?	42%	44%	13%	1%
	48%	*42%*	*9%*	*1%*
Text book?	18%	38%	27%	18%
	8%	*39%*	*17%*	*7%*
Text learning aids?	29%	64%	8%	0%
	15%	*80%*	*5%*	*0%*
Animation learning aids?	46%	44%	10%	1%
	36%	*50%*	*12%*	*2%*
Quizzes in learning aids?	34%	56%	8%	2%
	24%	*59%*	*17%*	*0%*

Results for London in normal type, and for *Bolton in Italic*.

Fig. 4. Results from questionnaire — usefulness of module components.

Students are showing a very positive response to the learning objects (referred to in the course as learning aids) and also the lab exercises which were based around using the graphic software library.

Further feedback on learning aids was sought through structured interviews primarily undertaken at London Metropolitan University. Students were asked if they had used a specific learning aid, 78% had used the text based aid and 81% the animated learning aids. Some of the students' comments give more insight into their views.

Usefulness of text-based learning aids:

"Good – better than reading a big book. Better on the eye."

"Very useful – no problem understanding it."

"Not useful – would prefer more teaching."

Usefulness of the animated learning aids:

"Good, you can see the code, shows what's going on when you press run. Interactive."

"Good – shows step by step the program. Animations help a lot."

"Nice, but no code – can't make the hammer or nail in Java or the horse run."

Usefulness of the quizzes:

"OK – helps you to look closely at the syntax of the code to get the right order."

"Good – reconfirms you know what you're doing."

"OK – not hard enough or complex enough."

The best feature of the course:

"Learning aids very helpful as I don't have all the books."

"Everything is on the web. Can access from home."

The majority of the comments listed above are positive in keeping with the pattern of the comments made. The proportion of negative comments made about the learning aids was about 12%.

Tracking statistics showed that a proportion of students were regularly using the learning aids. Usage tended to peak just before assessments, indicating revision usage.

The end of module results have shown a marked improvement in both institutions, when comparing pass rates with the previous year (Figure 5). The figures are for those completing the module.

Course	Percentage Increase in Pass Rate
MSc London Metropolitan	+12
HND London Metropolitan	+15
BSc London Metropolitan	+19
BSc Bolton	+23

Fig. 5. Increase in pass rates.

It is perhaps yet early to conclude which aspect of a radically changed course has led to this success.

7. Ongoing and Future Work

The Java module is continuing to be taught during the second semester in both institutions, but with smaller cohorts. A second version of the graphic software library has now been written and this is being piloted at Bolton. Some unused code has been removed from the classes in order to improve readability.

The learning objects will remain the same for this semester pending the completion of the evaluation. A future aim of the learning object work is to create a shareable repository that can be used across different universities and managed by the LTSN (Learning and Teaching Support Network) National Subject Centre for Information and Computer Sciences part of which is based at London Metropolitan University [8]. This is in line with the e-learning movement nationally which is working towards standardization and reuse.

8. Acknowledgements

Our thanks to the following who have also made valuable contributions to the development of this work. Claire Bradley of the Learning Technology Research Institute which is based at London Metropolitan University, who managed the development of the learning objects and conducted the London part of the evaluation; Richard Haynes of the Teaching and Learning Techology Centre London Metropolitan University, who wrote the Flash code for learning object animations and Professor Tom Boyle project leader.

References

[1] BERGIN J., Why Procedural is the Wrong First Paradigm if OOP is the Goal. 2000 http://csis.pace.edu/ bergin/papers/Whynotproceduralfirst.html [28/2/2003]

[2] BOYLE T., Design Principles for authoring dynamic, reusable learning objects. Accepted for publication in *Australian Journal of Educational Technology* (AJET) 2003, 19(1).

[3] COLLINS A, BROWN J, NEWMAN S., *Cognitive Apprenticeship: Teaching the Crafts of Reading, Writing and Mathematics*, NJ: Lawrence Erlbaum; 1989, pp. 453–495.

[4] GREY D, MILES M., Teaching Java Objectively – Reflections on a Web-based Java Course, in: *The Sixth Annual Java & the Internet in the Computing Curriculum Conference*, University of North London 2002. http://www.ics.ltsn.ac.uk/pub/jicc6/ [3/3/2003]

[5] *IEEE Draft Standard for Learning Object Metadata* [2002]. pp. 6. http://ltsc.ieee.org/doc/wg12/LOM_WD6_4.pdf [26/2/2003]

[6] JENKINS T., On the difficulty of learning to program, in *Proceedings of 3rd Annual conference of the LTSN-ICS*, 2002. http://www.ics.ltsn.ac.uk/pub/conf2002/jenkins.html [6/3/2003]

[7] KOLB D.A., *Experiential Learning*, Englewood Cliffs: NJ, Prentice-Hall; 1984.

[8] LTSN http://www.ltsn.ac.uk/ [9/5/2003]

[9] PETRE M., BLACKWELL A. & GREEN T., Cognitive Questions in Software Visualization, in Stasko et al. (Eds.) *Software Visualization: Programming as a Multimedia Experience*, MIT Press; 1998. pp. 453–480.

[10] SLOAN D., *Creating Accessible e-learning content in Access All Areas: disability, technology and learning,* pp. 35–38, http://www.techdis.ac.uk/accessallareas/AAA.pdf [5/3/2003]

[11] WebCT http://www.webct.com/ [9/5/2003]

[12] WENGER E., *Communities of Practice: Learning, Meaning and Identity*, Cambridge University Press; 1998.

Received: June, 2003
Accepted: September, 2003

Contact address:
Poppy Pickard
Department of Computing and Electronic Technology
Bolton Institute of Higher Education
Deane Road
Bolton BL3 5AB
United Kingdom
e-mail: P.Pickard@bolton.ac.uk

Peter Chalk, Ray Jones
Department of Computing, Communications and Mathematics
London Metropolitan University
Holloway Road
London N7 8DB
United Kingdom
e-mail: P.Chalk@LondonMet.ac.uk
R.Jones@LondonMet.ac.uk

POPPY PICKARD teaches programming at Bolton Institute of Higher Education. Her research interests are in the use of e-learning environments and the pedagogic effectiveness of learning objects.

PETER CHALK teaches introductory programming to very large courses at London Metropolitan University. His main research interest is in the use of on-line technologies to support the diverse learning needs of a modern student.

RAY JONES is a senior member of the teaching staff in the Department of Computing at London Metropolitan University. His research interests are in the reuse and re-purposing of learning objects and the application of XML and related technologies in learning object design.