Serious Game for Introductory Programming

António Coelho[1,2], Enrique Kato[1], João Xavier[1], and Ricardo Gonçalves[1]

[1] FEUP/DEI, Rua Dr. Roberto Frias, s/n 4200-465, Porto, Portugal
[2] INESC Porto, Campus da FEUP, Rua Dr. Roberto Frias, s/n 4200-465, Porto, Portugal

Abstract. For beginners in computer programming, the learning curve can be in many cases quite steep, especially if it is their first contact with this area. Plus, the traditional learning methodologies are usually based on doing countless exercises that aim to cover many areas, but are often disconnected from each other and can become tiresome, as they offer little immediate rewards to the student.

Nowadays serious games technology offers tools that may have potential to help computer programming students to become more engaged on their learning through a `learn while having fun' approach. This paper aims to generally describe our approach on the creation of a platform for deploying serious computer games for the teaching of any computer programming language. We will begin by describing the game mechanics, followed by the general system architecture and its data model, finalizing with a small conclusion.

Keywords: serious games, programming, e-learning, unity3d, domjudge.

1 Introduction

Serious computer games have emerged in recent history, but serious games have always been part of human culture as far as we know. Of course in ancient times, serious games took the shape of campfire fables told by grown ups to children to pass some kind of knowledge. Through these fables children were able to better understand the serious concepts being passed on to them instead of just being told directly.

Today's computer ever developing technology allows us to create more and more complex systems that can help us use gaming for learning purposes, allowing us to go beyond the simple fables told in the ancient times. Learning how to program can be enhanced and encouraged through this type of approach, by creating a serious game that is both a conduit of knowledge and experience and at the same time a fun task. But it is important to keep in mind that the serious games' emphasis must be placed on the educational objectives rather than the fun part, for they are primarily tools of education and not games to entertain [1].

The challenge we address in this paper is the building of a computer platform that allows the deployment of serious games that aim to assist on the learning of programming fundamentals with as much level of customization as possible, when creating a new game. Also, it has to allow the teacher to supervise, follow up students' progress and give them feedback. For this end, the game engine Unity3D was selected due to its rich features, its growing community, available resources and its ability to be deployed on the Web [2].

M. Ma, M. Fradinho Oliveira, J. Madeiras Pereira (Eds.): SGDA 2011, LNCS 6944, pp. 61–71, 2011.

2 Related Works

Inside the digital domain, a serious game can be defined as a contest played with a computer, which uses entertainment to develop training for military and corporative skills, education or use on health, public policy and strategic communication [15]. In education, research and interest has grown rapidly, and this can be noticed, as many European projects concern on design of educational games [14].

It is a well documented fact that there is a problem with Computer systems majors [8], since universities experiment a decrease on student demand [12], and abandonment because the low motivation from students towards difficult programming courses [14]. As this is of great concern between researches, heads are being turn towards using video games as a motivator for students in computer curricula and research [10].

Reviewing the work done by others brought on the table ideas on what can be applied onto our approach for the programming serious game. There are different approaches that researches have taken to tackle the problem on student motivation for Computer Science courses. Projects like Alice [9] and MUPPETS [13] propose software tools that allow students to create animations and virtual worlds through “programming-like" graphic commands. Another approach is the one taken by researchers at the University of North Carolina with the Games2Learn project in which they are making undergraduate students create small games about programming principles for novice students, like the game Wu's Castle [10] that helps students understand better the concepts of arrays and loops.

With a bigger gaming approach there are projects like IBM's Robocode, now public via SourceForge project, a framework for programming tanks and compete against tanks of other players in a multi-user environment [13]. Epsitec develops games like Colobot [11] where the objective is to solve several missions, which range from gathering resources, building structures, commanding robots to perform tasks and defend the land, a complete game in 3D. The Meadow is a game where developers created a custom engine to meet their university course expectations, where students control virtual sheep via C-Sheep programming language [8].

Authors agree that the new generation of students, the \Plug & Play" generation is more guided towards visuals and 3D environments, and so there is an interest on these kinds of interfaces to meet the expectations from these students [8]. On our approach, we decided to try with a 3D world since immersion is an important aspect in our project that may help students to continue with their programming studies. Also to meet the requirements of the courses given at the University of Porto, we are developing the project as an open platform, that could adapt to any programming paradigm and languages used at the faculty.

A tool made in house can also be designed to attend the needs of the specific population of the university and be modeled to reach the academic objectives of their courses.

3 The Mechanics

Every game has a set of rules that indicates how it is meant to be played by the players - the game mechanics - serving as a basis for the gameplay; if the mechanics are designed focused on the player, the final game may have a stronger quality.

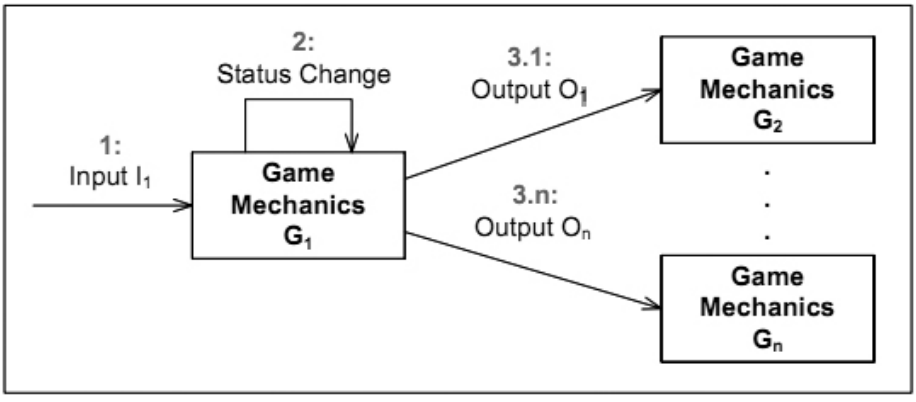

Fig. 1. Game mechanics interaction

Mechanics of serious games are similar to those found on commercial video games, but since the main objective of each is different, there are some differences to be aware. Whereas commercial video games' main purpose is to entertain and the mechanic used in them allows for various modes of play, they incorporate outcomes of leisure along with strong focus on presentation and storyline; serious games for education have a different path. Educational games focus on learning outcomes that are dependent upon an appropriate pedagogy and the underlying game mechanics and how the content is integrated into the game so the learning is intrinsic to play [4].

The system proposed in this paper is an educational platform providing serious games designed taking in consideration a set of specifications to meet the academic objectives to which it is aimed. Therefore, the game mechanics for this platform have to be designed and implemented with flexibility to provide distinct learning objectives for different programming courses. This paper will cover the core mechanics proposed for the nature of the serious games to be provided by the platform.

3.1 Project Specification

The project is being developed with an instructive stance, and since the definition of serious games differ from that of commercial video games, the specifications and objectives are defined clearly within the academic vision of the course.

The core mechanic should be mostly about solving problems via code understanding and programming, adding playful mechanics seen in commercial games as a way to aid in the immersion of the player and improve a continued use of the game. During programming it is common to have errors; the mechanic has to support this approach without penalizing the player too much. The design should ensure that the player keeps trying to solve the exercise, meaning that motivation is at hand and that the challenge does not compromise the advance of the student within the game. The general design of the game mechanics must be open enough to be able to cover other languages or technology courses. Teachers should be able to customize the game design to adapt it to their courses. Therefore, the design of the game should be general and open, in order to allow for further edition of the design, while keeping the flow and script of the game.

3.2 Proposed Core Mechanics

Given the three main specifications of the project, it can be said that the game core mechanic is basically to interact with the world via coding and scripting. The game is based on the programming fundamentals class, so the main learning objective is to use programming concepts to advance through the game. This section describes the core elements of the mechanics that meet the aforementioned specifications.

- The player-token of the game is a character and its robot companion. This token receives input from the player to interact with the virtual world of the game. The main character is the one that interacts with the virtual world while the robot sidekick is the one that gives academic advice to the player and prompts the main mechanic of the game: programming;
- The main interaction with the world is done by coding through the terminals scattered in the game. Each terminal presents one or various programming quests to solve. The correct solution of a problem triggers an interaction with the objects on the room, giving access to other areas, clear a hazardous object of the area, or get more info of the game plot and story. If the obligatory quests are not cleared the player will not be able to advance;
- The world of the game is designed as a series of levels that have one to N number of interconnected rooms to explore. The rooms have several kinds of objects that construct a general puzzle. To clear it and continue the game, players have to solve the programming tasks;
- Players can keep a mini inventory of items found during exploration of the levels. Items can be used to aid the player, to complete other quests or educative information about the problems that are being solved;
- Players increase their global and level score depending on the quests solved within the level. Getting some special items and solving optional harder quests increases the final score for the player. The score can be followed by students and teachers;
- Penalization to the player comes as reduction of points from the score. A quest has a number of points assigned if completed successfully. Incorrect code input has a limited point reduction from the full point value of each quest. A compiling error has minimal effects since it may be the most common problem. Several failed attempts of a compiled code mean more points deducted from the initial value of the quest;
- If the player reaches a limit number of tries, another path can be open to help develop the abilities of the student. This will bring a series of rooms that would have easier and guided challenges to helps student develop skills. At the end the character gets back to the room he was supposed to reach, to try to fulfill the normal quests;
- Code clues come from recovered logs, from the Robot Sidekick or by intervention of the teacher of the course. Two types of help are planned. The first is a set of tips and lessons that give a little bit more insight on the programming concepts seen during the level. The second one is the set messages that the Robot Sidekick or teacher can give as tips if the player starts to fail at the quests.

These mechanics are the ones that define the main functionality of the game and serve as a basis to have an overall open mechanics that can be applied to other courses. By open it is meant that on a matured version of the game, a group of academic staff could edit it to construct a new game using concepts of other informatics areas and courses. The instructors editing the game could apply other type of problems, tutorials and clues to the game to prepare it for another class, without having to edit or design the core functionalities of it. This is the main problem we seek to solve and an important objective when designing the game mechanics.

4 The Implementation

In order to successfully implement the mechanics discussed in the previous chapter, it was decided to use a client-server architecture, as shown in figure 2. From this image three distinct packages are visible:

These packages are connected to each other through either a local or a wide area network, such as the Internet. At the core of this platform the server package manages all aspects of the platform. The client package is the interface which users use to interact with the platform and finally, the DOMjudge package handles all source code evaluation. These components will be further explained in the next sections.

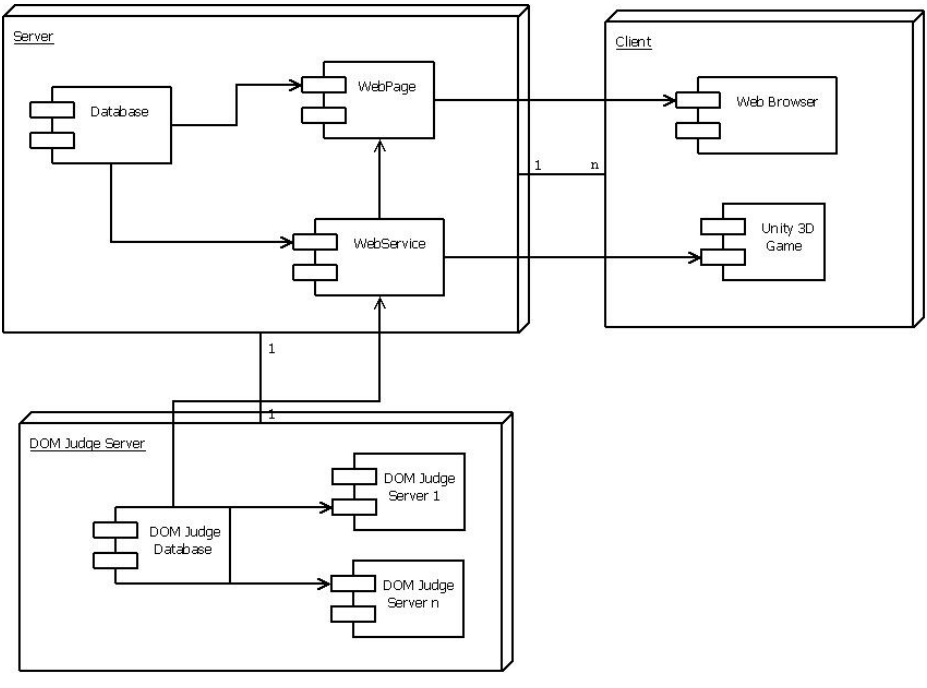

Fig. 2. Platform architecture

At this stage, both the server package and the client package have been partially implemented, and a prototype has been created as a proof of concept. The server package is still missing the web page, but the web service and database have been implemented. On the other hand, the client, although working well, will only be considered fully implemented once the server is complete and functional.

4.1 Server Package

The server as whole can be described as the heart of this project, pumping data to its components and other packages. Its mission is to interact with the other packages by managing and processing all information. This package is composed by three components:

- Web page - This component serves two main purposes: first it serves as the way through which the teacher creates and configures the game with tools like map editor and quest/exercise editor (more on this in the next chapter), and secondly it serves as a conduit for the teacher to follow his students progress through the game as well as it allows each student to see his own progress and receive feedback from his teacher;
- Database - This is where all the data regarding both the web page and the game is stored;
- Web service - This component handles all major game logic, controlling the game behavior and feeding it with all the necessary data required to build the virtual world with which players interact. This includes map data, quests (exercises), objects, tips (knowledge teachers may wish to add as a an extra way of helping their students with the exercises) and game plot information.

4.2 Client Package

To access the web page and the game, the user simply needs a recent version of any of today's browsers like Mozilla Firefox, Opera, Internet Explorer, etc.

The game itself was created through Unity3D Game Engine (free version) which allows the deployment of rich 3D applications that can be embedded into web pages. Through this functionality, the game will be able to be embedded into the web page, providing an all-in-one package for the students, automating and simplifying its access. This means, by login into the web page, the student automatically has access to the game without any need for manual software installation other than the Unity3D Web Player plug-in for web browsers.

The game itself is able to generate distinct games (but within the same scope) on run time, based on the data it receives from the web service. The game is meant to be mostly an end terminal and as such, most decisions are made on the server side, as explained before. This greatly limits the possibility of cheating by attempting to alter the game normal functionality.

The game starts by requesting game settings information, like the name of game or the location of the textures used in the game. The selected textures are then pulled

back to the game for later usage in building the virtual world. Once this is done, a menu appears requesting the player to log in. By logging into the game, the server creates a session (or regenerates it if one was already assigned to that particular account) and returns the session and profile information back to the player. After logging in successfully the player can select one of ten different profiles, each one giving him the chance to start the game from scratch.

Upon selecting a profile, the game shows introductory menus that, depending on what was defined on the database, can show for example the initial plot/story information and after the intro a menu describing the game controls and how to interact with the game is shown.

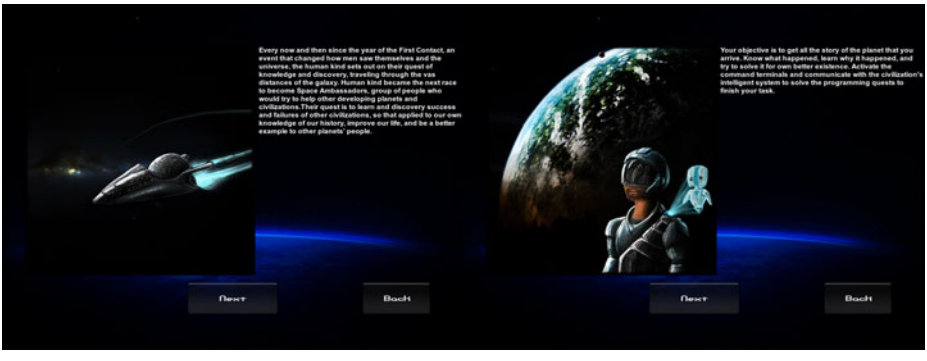

Fig. 3. Intro menus present in the prototype

Following the help menu, depending on the profile settings, the game requests the map where the player is supposed to be and a virtual 3D world is generated. Within this newly created world, the player is able to interact with several objects and move freely through it. These objects are:

- Quest terminals - These objects are the way through which players accept programming exercises which allows them to progress in the game, like enabling or disabling another object that prevents them from continuing the game, and by which they submit the program they believe to be a solution to the exercise;
- Data pads - These can be used to extend the game plot or give information about an exercise to help the player solve the problem;
- Force fields - These can be placed in narrow places, for example to prevent access to an area;
- Teleporters - These teleport the player to a specific location;
- Lights - These provide illumination to the scene;
- Doors - These allow the player to travel between maps;
- End game object - It looks like a door but it is meant to be the game exit, the final door, and by using it, a menu appears congratulating the player for completing the game.

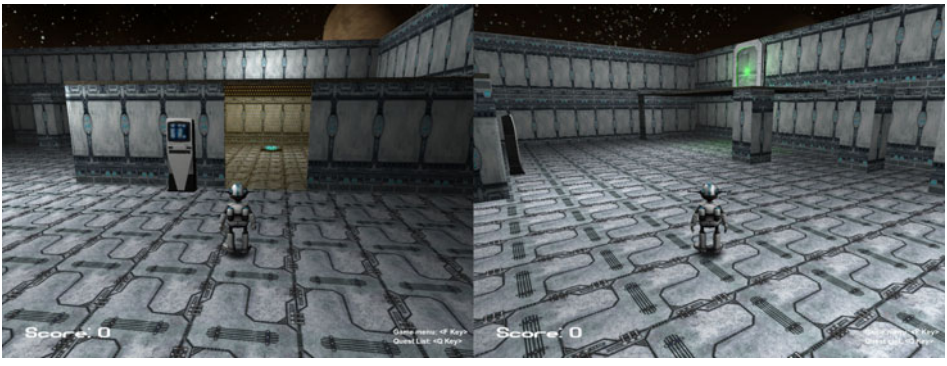

Fig. 4. In-game images

The quests, as mentioned before, are in fact computer programming exercises presented in a way that it feels it is part of the plot and the player needs to solve it if he desires to continue his adventure. Figure 4 shows two images of the first map of the prototype created to test this platform. The image on the left shows a terminal and to the right, a force field. The terminal contains the following quest:

Disable the Force Field

This force field receives energy from five power generators.

To bring it down, get the value of the five power generators and return the number of generator that has the least amount of energy available.

Objective: Write a program that receives 5 numbers and returns the position of the smallest of all.

Behind the force fields there is a teleporter which needs to be activated by another terminal. After activating and using the teleporter, the player is teleported above and has to hop around the room, like a normal platform game, to reach the door that leads to the next room. The exercises required solutions to be programmed in C++ language, but could easily be any other language. More on this reviewed on the next section.

4.3 DOMjudge Package

In order to evaluate the programming exercises, the selected system was DOMjudge, which is known for running programming contests like the ACM-ICPC regional and world championship programming contests [7].

The basic functioning of the automatic assessment can be described as follows:

- A solution for a problem is submitted by a team and stored in the database. Upon compiling and running, both compiler and program outputs will be stored and accepted or rejected;
- The first available Judgehost checks a not judged submission against the input-output data and marks it as judged;
- The result is automatically recorded and the team can view the result and the scoreboard is updated.

DOMjudge benefits from a distributed architecture, based on a client-server framework. Its foundation lays on the main DOMjudge server, which runs a MySQL database for keeping the submissions and in a variable number of Judgehosts that can be set up to mark the submissions (as it is shown in figure 2). The authors developed the system with security as one of the main concerns, providing detailed documentation on how to keep the installation secure and fail-proof [6].

DOMjudge is open-source, supports a wide array of programming languages and can be easily configured to support more. Moreover, a computer-based assessment system that integrates with Moodle was already developed and tested in this faculty taking advantage from this system [5]. It features a secure connection to a DOMjudge server through the use of web services and not only it serves as a proof of concept, but also as a basis for applying those web services in the game.

So, at each submission the system first starts by asking the server to accept the quest and if all goes well the profile is updated by setting the quest as an accepted quest. At this point a solution can be submitted in order to solve the problem, but only one at a time. Once a solution is submitted, the web service verifies if it is a valid submission and if so, adds it to the DOMjudge database. The DOMjudge then treats the source code, as described above. The game, on the other hand, will go into a state of waiting for a result, in which at each every few seconds sends a request to the web service for a status update on the solution result. Once the result comes in, it is processed in the web service, making all the necessary logic, and the game mimics the same logic, but the web service takes precedence over the game data.

5 Tools Used

To accomplish this project several tools are being used:

The web service is programmed in C# over .NET 4.0 Framework and to connect to the databases, MySQL Connector 3.3.6 is being used. The database used on both packages are MySQL >= 5.2. In order to run the web service, IIS 5.0 and 7.0 were tested and work perfectly over Microsoft Windows XP Professional or Microsoft Windows 7 Professional.

As for the game, it is supported by the engine Unity3D, which allows web deployment. This engine also supports both Microsoft Windows Operative Systems as well as Mac OS X, which both were tested and ran flawlessly during the tests to the prototype.

The DOMjudge servers, during the tests, were deployed on two virtual machines running Debian GNU/Linux, while the database and control website were on a third virtual machine, also running Debian GNU/Linux.

6 Conclusions and Future Work

At this moment, both the game and the web service are implemented, remaining the website with its game generation and teacher student interaction features.

In order to test the platform, a game prototype was implemented and tested by a group of students. The participants' feedback on the game was very positive and at

the end of the tests each of them filled a survey. This survey gave a new perspective, showing what still needs to be improved, like having more types of objects with which to interact with, adding sound to the game and the amount of information regarding the solutions evaluation results.

The web page with its configuration capabilities and teacher/students interaction is the main part that remains to be completed. As this gets implemented, the prototype will come in handy to test and help deciding the best ways the implementation process should follow. These include changing map formats that allow different and richer map layouts, communication protocols, exception handling, more objects, different ways to interact with quests and the way they behave upon completion.

Acknowledgements. This work is partially supported by the Portuguese government, through the National Foundation for Science and Technology – FCT (Fundação para a Ciência e a Tecnologia) and the European Union (COMPETE, QREN and FEDER) through the project PTDC/EIA- EIA/108982/2008 entitled 3DWikiU 3D Wiki for Urban Environments.

References

1. Wiberg, C., Jegers, K.: Satisfaction and learnability in edutainment: A usability study of the knowledge game laser challenge at the nobel e-museum (2003), http://www8.informatik.umu.se/~colsson/cwkjhci03.pdf
2. Petridis, P., Dunwell, I., de Freitas, S., Panzoli, D.: An engine selection methodology for high fidelity serious games. In: Second International Conference on Games and Virtual Worlds for Serious Applications (2010)
3. Fabricatore, C.: Gameplay and game mechanics design: A key to quality in Videogames (2007), http://www.oecd.org/dataoecd/44/17/39414829.pdf
4. Ulicsak, M.: Games in education: Serious games. Future Lab, http://media.futurelab.org.uk/resources/documents/lit_reviews/Serious-Games_Review.pdf (June 2010)
5. Pacheco, P.: Computer-based assessment system for e-Learning applied to programming education. Masters thesis (2010)
6. Eldering, J., Kinkhorst, T., Warken, P.: DOMjudge Administrators Manual (2010)
7. Eldering, J., Kinkhorst, T., Warken, P.: DOMjudge - programming contest jury system (January 2011), http://domjudge.sourceforge.net/
8. Anderson, E.F., McLoughlin, L.: Critters in the classroom: a 3D computer-gamelike tool for teaching programming to computer animation students. In: ACM SIGGRAPH 2007 Educators Program, 7es. ACM, New York (2007), http://portal.acm.org/citation.cfm?id=1282048
9. Cooper, S., Dann, W., Pausch, R.: Alice: a 3-D tool for introductory programming concepts. Journal of Computing Sciences in Colleges 15, 107 (2000), http://portal.acm.org/citation.cfm?id=364161
10. Eagle, M., Barnes, T.: Experimental evaluation of an educational game for improved learning in introductory computing. ACM SIGCSE Bulletin 41(1) (March 2009), http://portal.acm.org/citation.cfm?
11. Epsitec Games. Colobot. Epsitec SA, Belmont(2010), http://www.ceebot.com/colobot/index-e.php

12. Muratet, M., Torguet, P., Jessel, J.-P., Viallet, F.: Towards a Serious Game to Help Students Learn Computer Programming. International Journal of Computer Games Technology, 1–12 (2009), http://www.hindawi.com/journals/ijcgt/2009/470590/
13. Phelps, A.M., Egert, C.A., Bierre, K.J.: Multi-User Programming Pedagogy for Enhancing Traditional Study: An Environment for both Upper and Lower Division Students. Education, 8–15 (2005)
14. Shabalina, O., Vorobkalov, P., Kataev, A., Tarasenko, A.: Educational games for learning programming languages. System, 79–83 (2008), http://sci-gems.math.bas.bg:8080/jspui/handle/10525/1136
15. Zyda, M.: From visual simulation to virtual reality to games. Computer 38(9) (September 2005), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1510565