https://doi.org/10.48009/2_iis_2010_104-113

DESIGNING AN INTRODUCTORY PROGRAMMING COURSE USING GAMES

James Harris, Georgia Southern University, jkharris@georgiasouthern.edu
Vladan Jovanovic, Georgia Southern University, vladan@georgiasouthern.edu

==========

ABSTRACT

There is a clear need in computing to attract and retain good students. One way is to incorporate game programming into the curriculum [8, 10, 11, 14, 22]. This paper presents the detail design of an introductory, leveling, programming course using game programming, elaborates on the course outcomes and expectations regarding student achievement, and discusses curriculum context, operational details, and the benefits of the standardized approach selected for the course delivery. It is of interest to those who would like to alter the curriculum of their introductory programming course using a contemporary theme in order to motivate a generation raised on gaming.

Keywords: Introductory Programming, Visual Basic, Games Programming, Course Outcomes

INTRODUCTION

The course CSCI 1230, Basic Programming is an introductory course in programming using the latest edition of the Visual Basic language. This first programming course serves to provide baseline programming skills as a prerequisite for entry into the computer science and information technology curriculum, specifically to Principles of Programming I for the computer science students, and the Introduction to Java for students enrolled in information systems and information technology programs. The course was originally designed in 1995 to help students without any programming experience. The course also serves as a computing requirement option for students of education and other majors outside the college of Information Technology (hosting CS, IS and IT programs). The course's traditional emphasize was on essential procedural programming concepts and the use of programming as a problem-solving tool with an ever increasing coverage of object-oriented concepts. The perennial problem of delivering too much syntax with a little long term effects on programming skills acquisition forced us to reconsider both the student's motivation and the cognitive demands of programming (and even what it means to know how to program). Consequently, games were adopted as a domain of examples allowing us to communicate a rich set of relevant concepts in an evolutionary manner. Guzdial and Soloway [8] found that games are an intrinsic motivator for students who are learning to program. Visual Basic is an ideal environment for teaching game programming in an introductory course because a novice programmer can, in a short period of time, learn enough VB to produce impressive looking GUI programs [10].

In this paper we present an outcomes based, course design in which most basic and some advanced concepts are presented using simple games. The focus on games is followed as much as possible, and came about to satisfy three objectives:

a) to motivate students,
b) to retain students, and
c) to introduce students to game programming as a field- job specialization.

Our motivation as teachers stems from analysis/feedback by students and instructors from years of teaching introductory programming to the multitude of students with very diverse abilities and backgrounds in a broad variety of educational programs. In the literature [8, 10, 11, 14, 22] games are discussed as a viable approach in support of teaching and learning. None of the game specific sources such as [9, 19, 20, 21] or typical textbooks [1, 2, 3, 4, 5, 7, 12, 13, 15, 17, 18] are broad enough to be used as the course textbook, but collectively were the source of most exercises and assignments in our pilot course delivery.

COURSE DESIGN

CSCI 1230, Basic Programming, introduces students to the syntax and semantics of a modern programming language and most of the key programming constructs of structured, object-oriented, and event driven programming. Both the specifics of the Visual Basic language and the software development process using a professional tool, MS Visual Studio Integrated Development Environment (IDE), are sufficiently explored in this course to achieve certain outcomes (see Table 1 in the

appendix, "Course Outcomes mapped to Program outcomes and Expectation levels".)

A variety of methods of instruction are used including, but not limited to, lecture, laboratory demonstrations and practices, readings from textbook and periodicals, analysis of interesting Visual Basic programs, as well as those from the large standardized set of group and individual programming assignments. The self study portion also involved using Video Notes supplied with the modern textbooks [7]. The course meets for three, 50-minute class periods per week for approximately 15 weeks, plus a scheduled final exam which includes the presentation of a final project, the result of a collaborative work by student teams to design, develop, test, document and present an original Visual Basic game application. A typical week of classes is as follows:

- A short concept session including demonstrative examples of short live programs followed by an in class hands on tutorial providing immediate feedback,
- A supervised, hands on lab session involving a more substantial programming problem, an assignment integrating and extending upon concepts previously presented. This assignment is announced in advance, requires only minimal documentation and may be completed in class by the best prepared students, but is allowed to spill over class time encouraging students to collaborate and seek consultation from tutors and instructors to complete the assignment before the next session,
- A consolidation session improving upon alternative solutions presented. After the in-class assignment, a more challenging take home assignment is given.
- Homework (HW) is assigned with an open ended option for more advanced students that may otherwise be bored and with some supporting hints and/or clarifying specifications. Students are given opportunity to ask for clarification as most HWs are announced well in advance. The HW type assignment is characterized with more involved documentation and more extensive coding demands and is expected to be ready for the next week, typically at the time of a supervised lab session in the next cycle. More advanced subjects such as graphics, arrays etc. require multiple sessions of each type, however, the sequence follows the same pattern.

If some concepts present a problem (on Labs or HWs), a remedial session improving upon samples of student's work is inserted. The advanced students are encouraged to help others and explore in depth before the class moves on to the next subject. Most concepts are illustrated with games with most of exercises, labs and HW assignments also based on games Here are some examples of applicable games with the key concepts that they illustrate:

a) Striker with controls/timers
b) Picture Dictionary etc. with VB controls
c) Dice and Slot Machines, with enumeration, random numbers, and functions,
d) Frasier diagnostic simulation with strings processing,
e) Black Jack and educational games with alternatives
f) Hangman with loops and menus, and graphics
g) Poker with arrays (also elements for 3D games,)
h) Juke Box with audio and media player,
i) Tic Tac Toe and Sudoku with control arrays,
j) Memory games and the likes with O-O
k) Tetris, Space Invaders, with advanced 2D graphics
l) Multi-player 3D games (optional) for advanced students

Professional issues addressed include requirements, architecture and GUI documentation, black box testing, detail design and deployment. A typical course schedule is shown below in Table 1.

Weeks	Schedule of Classes by Topic	Assignments
Week1 **Week 2**	- Course: Requirements and Organization - Demonstration of the use of VB - *VB Controls and events* - Tutorials + **LOGO** as a Splash Screen	 Ch 1 - tutorial exercises Ch 2 - tutorial exercises **Lab1-Logo plus Ch2-P2**
Week 3	- Project **Standards** +Visio: Documentation Variables, Input and Output	Ch3- tutorial exercises **HW1-Picture Dictionary**
Week 4 **Week 6**	**Design and Logic Ch 4** - Relational and Logical Operators, - If Blocks and Select Case Block - String processing; **Frasier** - Enumeration with **Dice** (Craps) game GUI Design: With elements of Graphics GDI+ - Additional Controls, Timers- **Striker** game	Ch 4-tutorials **Lab 2 –Black Jack** **HW 2- Slot Machine**
Week 7 **Week 8** **Week 9**	Ch 5 Loops *and* Testing for VB programs; **Multicurrency Converter,** data validation, - Using Visual Studio Debugger Ch 6 Procedures, Subs and Functions + Media, VTOC diagrams and modular design and testing. Additional content: *games* **Hangman, Snake, etc**	**Ch 5- tutorials** Ch 6- tutorials Ch 7- tutorials **Lab 3: JukeBox**
Week10	Ch8-Arrays including multidimensional Arrays - Additional: Control Arrays, **Security Pad** - case study **Sudoku**	Ch8- tutorials **HW3: Poker** **Lab 4: Tic Tack Toe**
Week12 Week13 Week15	**Intermediate VB for Game Programming;** - case studies: **Tetris, Space Invaders etc.** - advanced game concepts, 3D Direct X	**HW 4: Memory Game**
Week16	**Final Exam**	**Creative Game-Term Project**

Table 1: A Sample Course Schedule for a Course in Visual Basic with Game Programming

CONCLUSION

We are still working out fine balances among constant content/delivery innovation, individual motivation/experimentation and standardization of expectations and the verification of achievements. We plan on systematically comparing our newly established game programming approach with the traditional approach this fall. Four sections of VB will be taught, two with a game programming emphasis and two without. Student course evaluations, dropout rates, student success rates (percent passing the class, and percent with A grade) and student and instructor surveys will be used as measures. In addition, we will compile and analyze past several years of student success rates with the traditional approach to the introductory programming course, with the same instructors as well as with several different ones, all in order to establish a relevant but unbiased baseline. One working hypothesis is that more programming concepts can be covered with better student retention and higher success (level of outcome achievement). We are certainly not the first to advocate using games first (23) but intend to make this a strategic choice, spreading this motivating influence throughout the curriculum.

One of the remaining problems is to vary HW assignments to prevent reuse of code solutions without sufficient intellectual and creative engagement among parallel course sections and successive semesters. We are of the opinion that with increased faculty commitment and better student

motivation positive outcomes will fully materialize and we might be able to restructure CS1+CS2+Data Structure sequence that comes after the Intro course discussed here. Recently we started offering a game programming certificate to CS majors and this is already proving to a great recruiting tool.

We are also working on a game programming framework (see Appendix, Figure 5) for integration with professional game engines and similar artifacts later in the curriculum.

REFERENCES

Bell, D., Par, M. Visual Basic.NET for Students, Addison Wesley 2003.

Bradley, J.,Millspaugh,A., Programming in Visual Basic 2008.

Crew, T.,Murphy, C., Programming Right from the Start with Visual Basic.NET, Pearson 2004.

Deitel,P,, Deitel, H.,Visual Basic 2010 How to Program, Pearson 2011.

Ekedahl, M., Guide to Developing and Implementing Windows-Based Applications with Microsft Visual Basic.NET, Thomson 2004.

Flynt, J. Software Engineering for Game Developers, Thomson 2005.

Gaddis, T.,Irvine, K., Starting out with Visual Basic 2010, 5th edition, Addison Wesley 2011.

Gudzdial, M., Soloway, E., (2002), Teaching the Nintendo generation how to program, Communication of the ACM, Vol. 45 N4, pp 17-21.
Harbour, J. Visual Basic Games Programming with Direct X, Premier Press 2003.

Hu, M. (2008): A Framework for teaching Novice VB Programming Using Motivational Game Scenarios, 20th Annual Conference NACCQ, New Zealand, pp 89-96.

Ling, W., Programming Sudoku, Apress 2006.
McKeown, J., Programming in Visual Basic 2010, Cambridge University Press, 2010,

Patric, T. Start to Finsih Visual Basic 2005, Addison Wesley 2005.

Purewal, T., Bennet, C., (2006), A framework for teaching polymorphism using game programming, Journal of Computing Sciences in Small Colleges, V22, N2, pp 154-161.

Reynolds-Heartle, R., OOP with Visual Basic.NET Step by Step, Microsoft Press 2002.

Schneider, D., An Introduction to Visual Basic 2010, 8th edition, 2010.

Shelly, G. Hoisington, C., Visual Basic 2008, Course Technology 2009.

Stephans, R., Visual Basic 2005 Design and Development, Wrox 2007.

Walnum, G., Teach Yourself Game Programming with Visual Basic in 21 days, SAMS, 2001.

Walsh, P., LaMothe, A., The Zen of Direct 3D Game Programming, Prima 2001.

Weeler, D, et all, Beginning .NET Game Programming in VB.NET, Apress 2004.

Xu, C., (2006), Why and how to teach game programming, FIE Conference, Las Vegas, pp 215-220.

S. Leutenegger and J. Edgington, "A games first approach to teaching introductory programming", in *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07)*, pp. 115–118, Covington, KY, USA, March 2007.

APPENDIX

Table2: ABET Learning Outcomes

Learning Outcomes: CSCI 1230 Introduction to Basic Programming	**Mapping to ABET Learning Outcomes**
Understand and use fluently basic blocks of VB.NET - syntax/semantics, identify and fix syntax errors, compile, build and execute short (Visual Basic) programs using Microsoft's Visual Studio Integrated Development Environment	1a, 1i
Understand and use comments, constants, variables - names, types, and size	1a, 1i
Ability to create GUI's using controls	1a, 1i
Ability to write code using built in arithmetic operators and expressions	1a, 1i
Ability to select appropriate if, for, while and do-until selection and iteration statements and combine them in a well structured manner	1a, 1i
Ability to design logic for small functions and sub programs	1a, 1i, 2b
Ability to understand syntax and use of text file I/O	1a, 1i
Understand how to read, identify, write and use test drivers and test cases for testing simple Visual Basic classes and programs	1a, 1i
Ability to create, implement, and use one and two-dimensional arrays.	1a, 1b
Ability to use an IDE to strep through and debug code	1a, 1b
Ability to create simple games, including elaborate interactions, graphics, sound and other media	1a,1b,1c,1d,1f,1h,1i,2a,2b

1. General:
(a) An ability to apply knowledge of computing and mathematics appropriate to the discipline;
(b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution;
(c) An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs;
(d) An ability to function effectively on teams to accomplish a common goal;
(e) An understanding of professional, ethical, legal, security, and social issues and responsibilities;
(f) An ability to communicate effectively with a range of audiences;
(g) An ability to analyze the local and global impact of computing on individuals, organizations and society, including ethical, legal, security and global policy issues;
(h) Recognition of the need for, and an ability to engage in, continuing professional development;
(i) An ability to use current techniques, skills, and tools necessary for computing practice.
2. CS Specific:
(a) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices;
(b) An ability to apply design and development principles in the construction of software systems of varying complexity.

Table 3: Key concepts we present throughout the curriculum- only the first 15 in VB:

1. Context: Integrated Programming Environment, Application Types, Projects, Views
2. GUI: controls (forms, buttons, textboxes, picture boxes, etc.) their properties and values
3. Keywords, (built in) native types and statements: operands, types, values, operators
4. Library methods and parameters
5. Events and timers
6. Procedures and functions, return types, parameter passing by value and by reference
7. Structured blocks and their nesting (ifs and case based alternatives, and loops)
8. Graphics GDI+ and Media
9. Documentation, specification, description, representation
10. Black box test planning and execution
11. Debugging
12. Input Validation
13. Arrays and Files
14. 3D elements with Direct X
15. Deployment (Intro VB)
16. Assembly, builds, resources (CS1)
17. Exception handling (CS1)
18. Structures (CS1)
19. Modules (CS1)
20. Treads (CS1)
21. Classes and objects with their methods, (CS1)
22. Aggregation and Inner classes (CS1)
23. Interfaces (CS1)
24. Regression test automation (CS1)
25. Inheritance and Polymorphism (CS1)
26. Containers and advanced GUI design (CS2)
27. Advanced event driven programming (CS2)
28. Reflection (CS2)
29. Regular Expressions, Automata, and Functional programming (Theoretical Foundations)
30. Encryption Programming (Computer Architecture- Assembly programming)
31. Team programming, Configuration Management using shared repository, (Data Structures)
32. Collections (Data Structure)
33. Templates-generics (Data Structures)
34. Core algorithms (sorting, searching, merging, etc.) complexity analysis (Data Structures)
35. SQL, XML and LINQ (Data Base)
36. Socket Programming (Data Communication)
37. Web access to DB (Distributed Web-systems Design) including client and server side scripting
38. Web services (Distributed Web-systems Design)
39. Patterns, Architecture and Frameworks (Object Oriented Design)
40. Model Based Development and Testing Automation (Object Oriented Design)
41. Advanced Algorithms Design and Analysis (Algorithms Design)
42. Artificial Intelligence and Logic Programming (Artificial Intelligence)
43. Graphics programming, transformations, physics (Game Programming)
44. Multiplayer Games Design and Coding (Game Programming)
45. Program Generators and Meta Programming (Software Engineering)
46. Metrics and Quality Assurance (Software Engineering)
47. Capability, Maturity and Process Improvement (Software Engineering)
48. Etc.

Table 4: Documentation sample

Document	Project	Homework	Lab	Exercise	Example
Feedback-Cover Page (instructor adds)	X	X			
Feedback Code Scorecard (instructor adds)	X	X	X	**x?**	
Requirements Docs:					
Problem Statement-scope	X	X	X		
Use case -Scenarios	X	X			
Storyboard	X				
Other as appropriate- a student choice	X**?**				
Design-Planning-Users docs:					
GUI mockup	X	X	**x?**	**x**	**x**
Test Plan- and report	X	X	**x?**		
Event Plan	X	X	**x**	**x**	**x**
Structure and/or Class Diagram	X	X	**x**		
Installation-users-guide	X**?**				
Code docs:					
Source code	X	X	X	X	**x**
Properties	X	X	**x**	**x**	
Legend X- student; **x** – Instructor; **?-** possibly					

Figure 1: Tic-Tac-Toe GUI

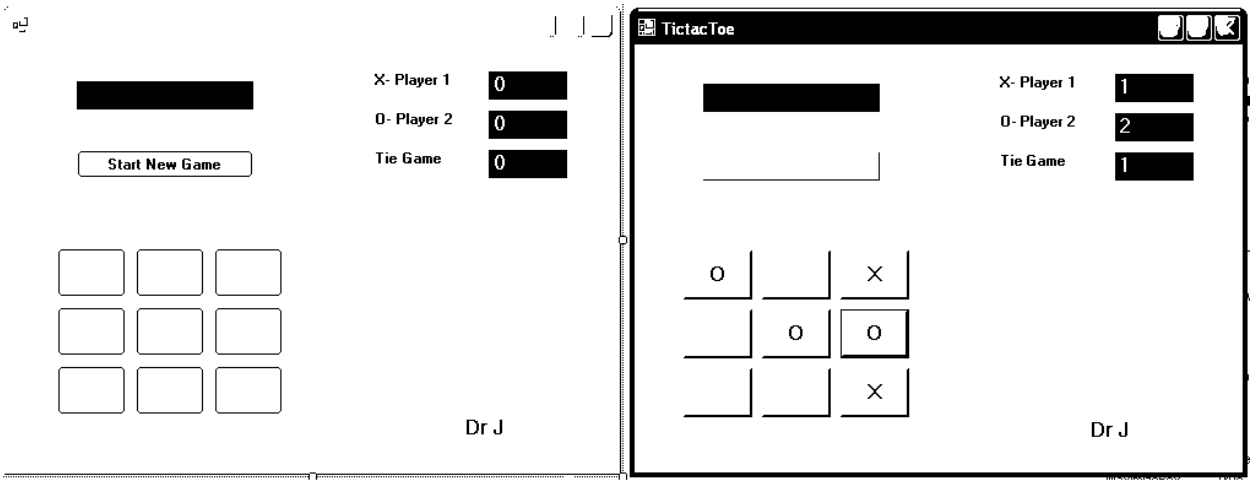

Figure 2: Tic-Tac-Toe Structure Diagram

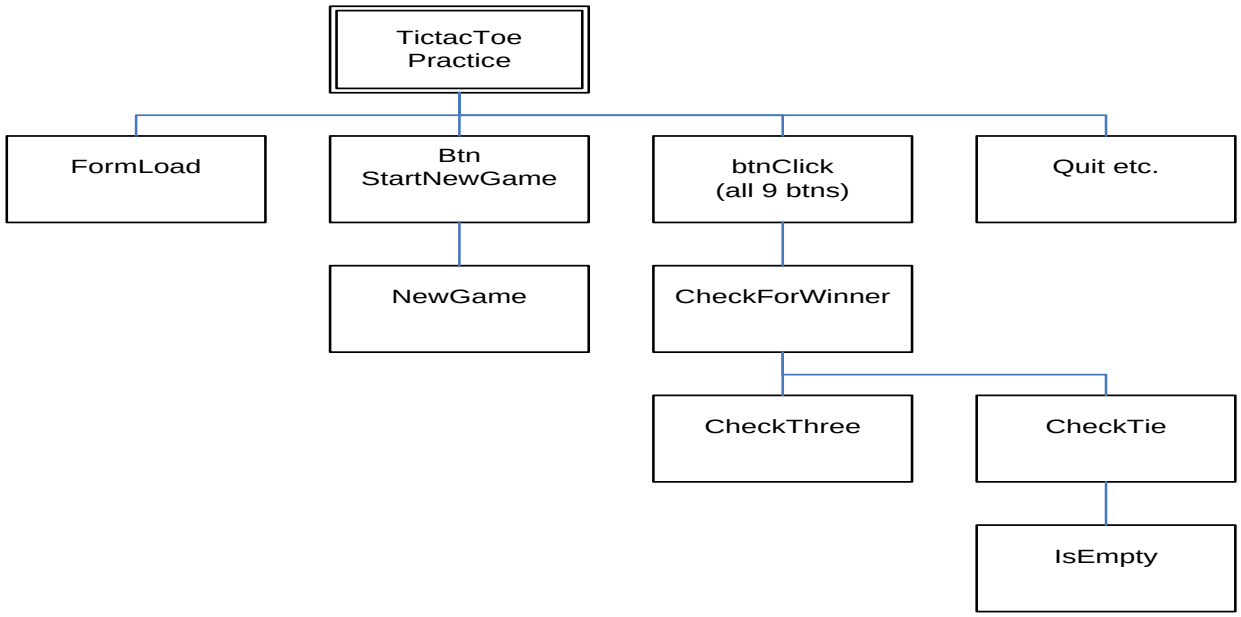

Figure 3: Hangman (simpler without menus)

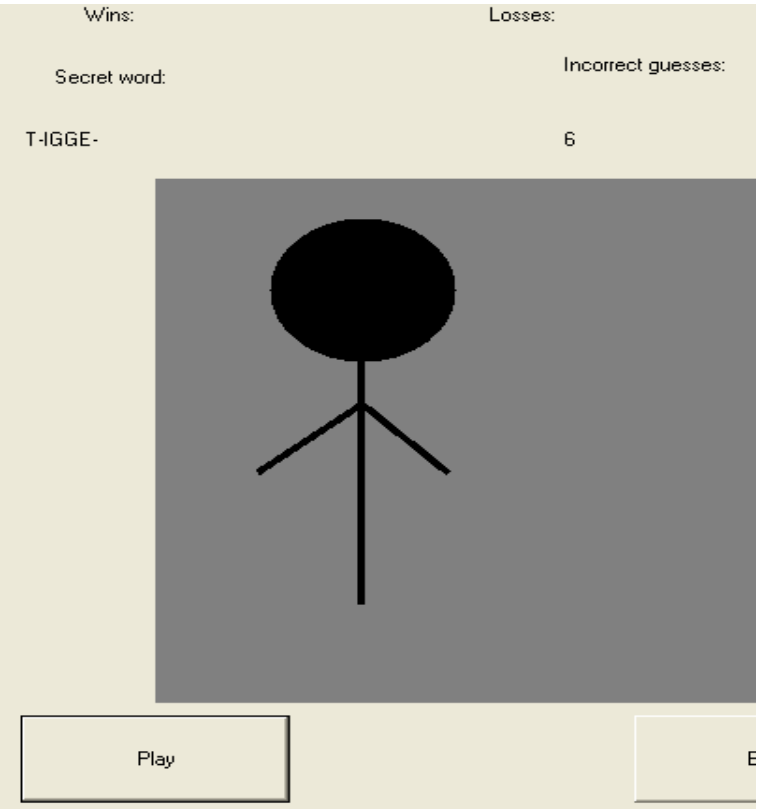

Table 5: Test Plan for Hangman

Test	**Test Plan Hangman V3**	Comments		Report	Pass
Number	Input-1	improvements	expected	actual	Fail
1			blank not OK		
2	Book		OK		
3	BOOK		OK		
4	book		OK		
5	Blank		OK		
6	Wastepaper		OK		
7	A		OK		
8	an		not		
10	watter hole		not		
11	37		not		
12	@		not		
13	Hypopotamus		too long		
14	xxxxxxxxxx…….xxx		not		

Figure 4: Stryke the Flying Dragon - click on the skateboard rider to score

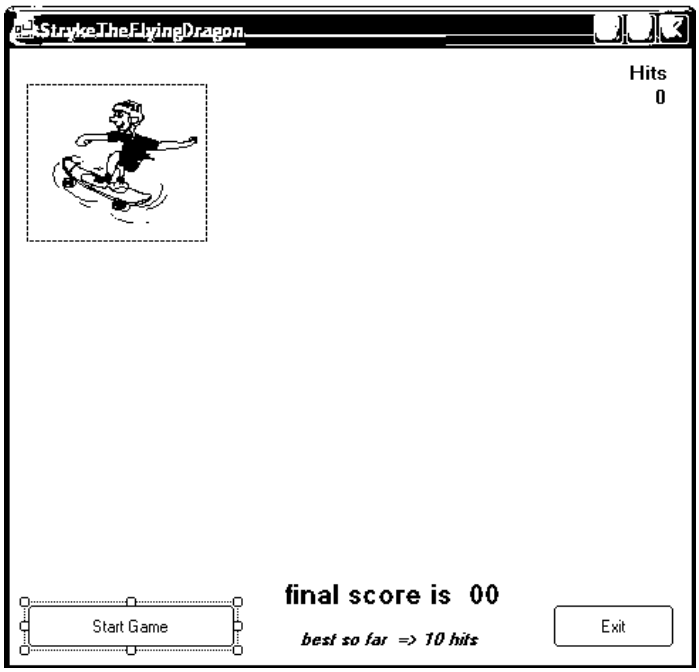

Figure 5: Meta-model for Game Design Frameworks