TEACHING PROGRAMMING USING VISUALIZATION

Andrew Rudder andrew.rudder@gmail.com
Dr. Margaret Bernard mbernard@fsa.uwi.tt
Shareeda Mohammed shareeda.mohammed@gmail.com
The University of The West Indies
St. Augustine Campus, Trinidad and Tobago.

ABSTRACT
Teaching computer programming to young students is a major challenge. This paper describes a web-based course for teaching programming using visualization and a gaming theme. Concepts are taught using real world examples that the target students can identify with; in this way some of the problems related to understanding abstract concepts are alleviated and learning occurs in context. The full range of multimedia technology including animation, sound and video are used to immerse the student in an environment where learning is fun and visual display of the concepts reinforce learning. A gaming theme is used for formative assessment. Students are provided with 'game' activities such as "spot the error", "predict the output" and "put in order". These enhance critical thinking. They test comprehension and knowledge as well as higher order thinking skills.

KEY WORDS
Computer programming, Visualization, Web-based Education

1. Introduction

The teaching and learning of programming is a challenge to both students and teachers alike. Research indicates that this is a universal problem. Some believe this is partly due to the abstract nature of many of the concepts. This difficulty has prompted researchers to investigate tools and approaches that may ease the difficulty of teaching and learning programming [3].

Students often have difficulty in developing algorithms to solve even simple problems; whilst there may be understanding of the knowledge of the topic area, how to apply this knowledge is lacking. This is a common problem in the area of programming. By its nature, the area of programming has a mathematical background. However, the skill of applying it requires an almost artistic development. The traditional approach to teaching programming has been largely abstract and mathematical in nature. Development of programming skill however, requires the user to take a real world problem and construct an algorithm to solve it. This requires the user to see the relationship between programming and the real world and translate one into the other. Technology can facilitate learning by providing real world contexts that engage learners in solving complex problems [2]. Interactive learning environments based on games offer appropriate contexts for active learning [4].

This paper describes the design and development of a web-based system for teaching programming. The system combines educational principles, visualization and web design and development strategies to produce an interactive web based course. The course is designed for young users (age group 14-17) in Caribbean schools. One of the guiding principles of the system is to use visualization to bridge the gap between real world situations and programming concepts. The system includes an environment that aids students in developing the basic elements of procedural programming including control structures, documentation, coding, program design and testing. Macromedia Flash MX 2004 is used to provide interaction and develop event driven content so that learning is active. The system is geared to be used either as a teaching aid or a self learning tool by individual students.

The rest of this paper is organized as follows: Section 2 describes the use of visualization in developing content for individual lessons. The guiding principles and design considerations are discussed. Section 3 describes the design of the evaluation content pages. This follows a gaming theme that is fun and reinforces learning. Section 4 provides the course architecture and the development environment. Some concluding thoughts are given in Section 5.

2. Design of Lesson Content

The content of each lesson was partitioned into its essential elements and then presented in discrete informational units (learning objects). The lessons were designed to leave the student with a solid understanding of the core concept(s) involved in a particular topic area. Each concept is introduced with as little "baggage" as possible in small clear packages.

By using programming terms with increasing regularity throughout the unit lessons, the student learns to use the terminology, and more importantly the context and associated information with it. Additionally, by repeating terms met in previous lessons or by repeating processes in some examples, reinforcement of these ideas is achieved.

The student therefore does not have to learn definitions by rote, but rather in a contextual natural environment.

Real world examples are used in all illustrations. Describing a principle or concept in a vacuum makes it more difficult for a student to grasp, since it does not relate to anything they are familiar with. Every student's experiences will be different. The illustrations used in the lesson content are typical scenes in the Caribbean that the intended target audience can relate to. One of the problems associated with teaching programming is that students find it hard to solve real world problems using programming. Using real world examples as a tool for teaching programming concepts better equips students to apply concepts and bridge the gap between the real world and computers. In this way, the student from the inception of his/her learning, will be directed toward developing the relationship required between the real world and the logical programming that is required to map it.

The design of the lesson content took cognizance of the fact that there are multiple learning styles. The content in the lessons are presented in a manner that appeals to learners different learning styles. In order to facilitate all these learning diversities, animations were used where possible. These animations included textual information, as well as graphic and auditory elements in an attempt to immerse students in an effective learning environment that appeals to their senses. By using relevant graphical representations and animations, students will better grasp the concepts required as they can visualize the concepts and how they occur.

2.1 Visualization in lesson content

We will illustrate the use of visualization in teaching programming by examining the design of the content for teaching two control structures: Selection (If …Then … Else) and Iteration (While Loop).

2.1.1 If … Then … Else control structure

The domain problem is one students can easily identify with. Students are initially faced with the real problem of deciding what to do. Puffy (our character) has a certain amount of money and he wants the buy a bike. (Fig 1 a).

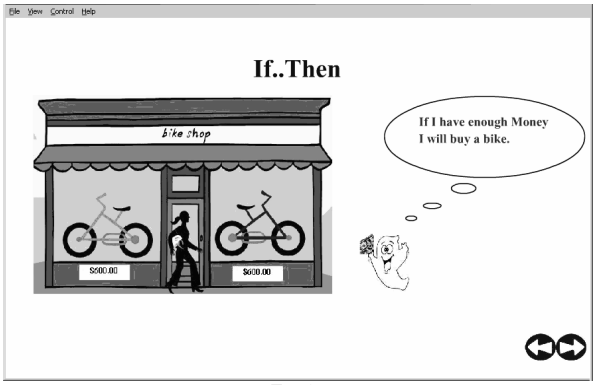

Fig 1a

Fig.1b shows the problem reduced to a decision about the amount of money Puffy has. The problem statement is described in English narrative form followed by the pseudocode equivalent. This allows the learner to quickly translate the narrative text to a more programming-like form. The audio overlay explains the two forms so students learn from what they see and what they hear. The repetition reinforces learning.

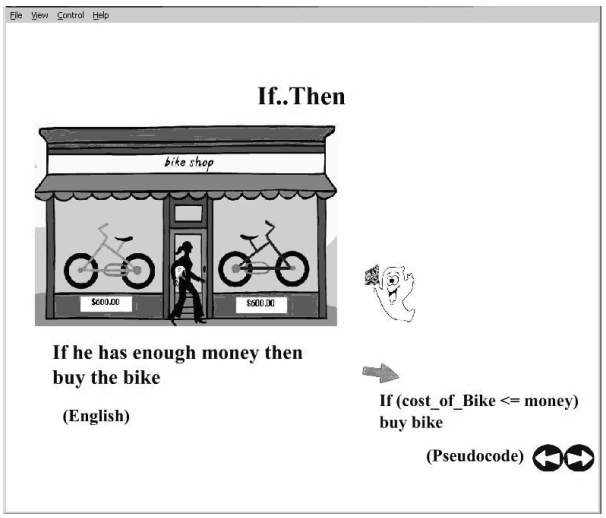

Fig 1b

Fig 1c introduces the flowchart using animation to build the flowchart in a natural sequence. It depicts different but equivalent views of the If .. Then statement. The If ..Then statements are introduced logically and sequentially with the If ..Then …Else being introduced only after the student has practiced and been evaluated with the If ..Then.

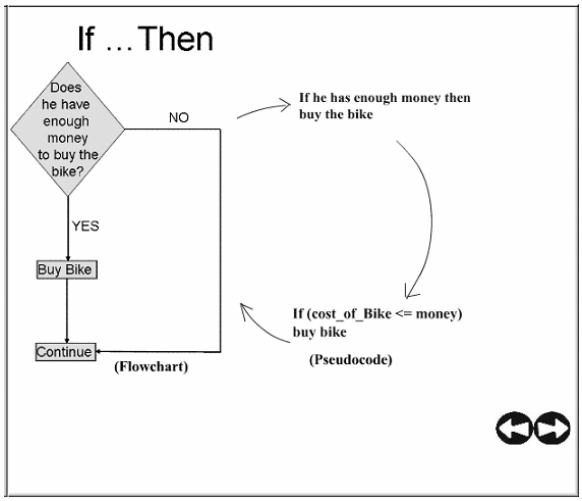

Fig 1c

2.1.2 Iteration (While Loop)

The problem domain is: the sun is shining and Puffy is playing ball (animated) (Fig. 2a). Audio is used to further enliven the scenario. Sounds of birds and the bumping ball can be heard. The condition and the activity that is

being performed are then presented (Fig. 2b) allowing students to associate the condition of the While loop with the activity.

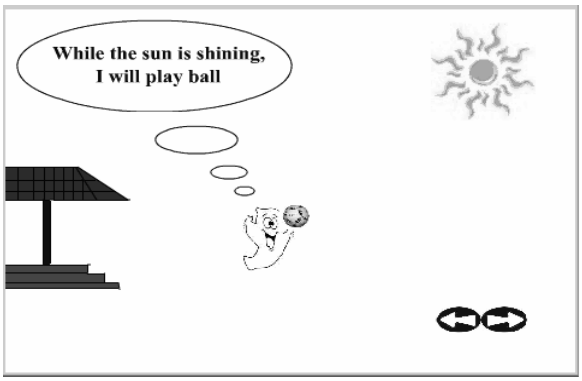

Figure 2a

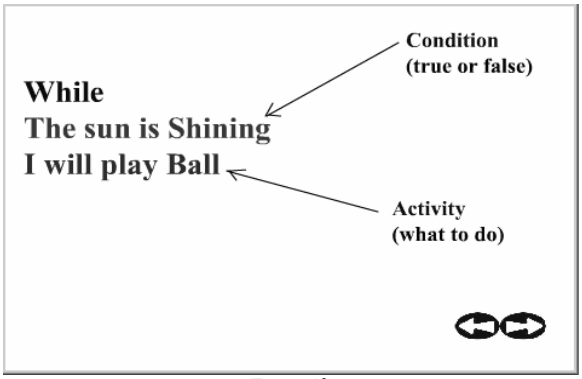

Figure 2b

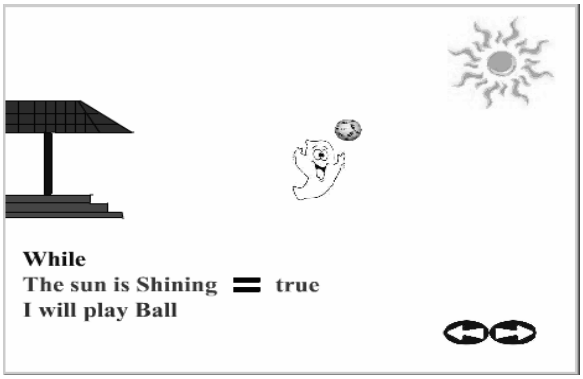

Figure 2c

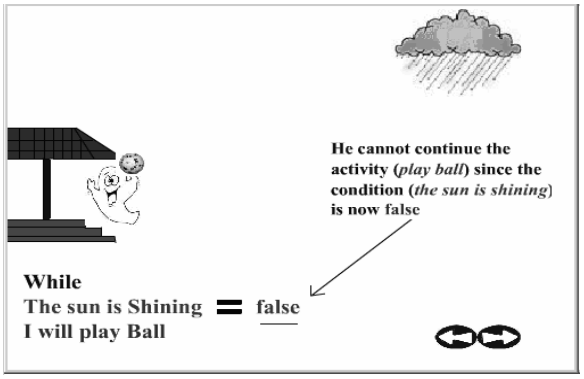

Figure 2d

An events then occurs which terminate the condition of he while loop exemplifying to students the boundaries of the while condition (Fig 2c and 2d).
The animation provides continuous movement so what students view is not a set of disjoint frames but a movie that they can relate to.

Following the demonstration of the concept, the student is presented with a typical problem used in traditional teaching of programming: counting from 1 to 10 (Fig 2e). The content here shows the While Loop and traces through the program showing the values of the variable 'number', including the exit from the While Loop. The flowchart and pseudocode follows.

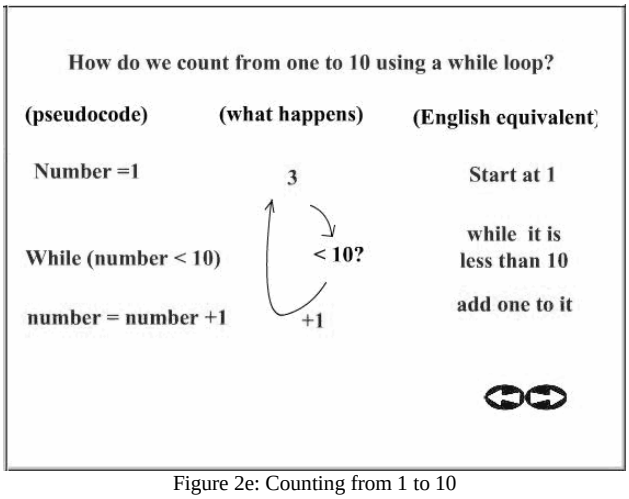

Figure 2e: Counting from 1 to 10

3. Design of the Evaluation content

Children and adults alike are intellectually engaged by computer games. Computer games immerse individuals in visual environments in which they have to make decisions and choices and reflect on these choices. Games enhance critical thinking. Computer games provide fun, motivation, structure, social interaction in virtual worlds, doing sparks creativity and provides ego gratification all while the individual learns. It is with these aspects and advantages in mind that the evaluation content follows a gaming theme. The evaluation section is intended to be an aid to learning to reinforce the lesson and can be used by the student for self assessment. These activities are designed to help novice programmers understand how programs are executed, focusing particularly on program flow. Games have been used in other systems for teaching programming, for example [5] uses word games that manipulate strings and number games that manipulate integers and random numbers.
The forms of evaluation are designed to test the six levels of Blooms' taxonomy [1]: Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation. The evaluations begin with assessing specific levels of the students' understanding. This is important since in order to teach the higher levels, for example the Application

and Synthesis, the lower levels such as the Knowledge and Comprehension for an area must be relatively strong. This hierarchical approach facilitates students identifying gaps in their knowledge and hence being able to strengthen these areas. Various types of evaluations are employed in the web-based course with each type testing a specific level in Bloom's taxonomy. In this paper, we here examine three of these evaluation types, "spot the error", "predict the output", and 'put in order".

The "spot the error" test (Fig 3a) evaluates the Comprehension level. Students must understand and interpret the meaning of the lines of code. The "spot the error" type test uses the facts and concepts gained in the Knowledge level and tests to determine whether the student can identify errors in information that are displayed to them. This evaluation type aids in the development of a very important skill of programming, debugging skills. The errors may be logical or syntactical in nature. Once successfully completed, the student shows an understanding of the syntax and some logical flow of programming. There is immediate positive feedback provided to the student which serves as reinforcement as well as encouragement to the student.

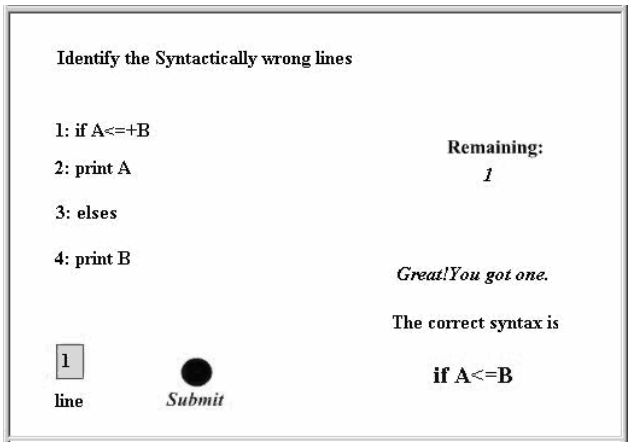

Fig 3a: Spot the Error

The "predict the output" test (Fig 3b) tests the Analysis level. Students are required to break down the code segment into its component parts, understanding what each part does in the given context. The "predict the output" test displays a piece of code and requires the student to "predict" or elicit the results of the code or changes in variables' values. The student must be able to clearly understand and trace through the snippets of code and correctly predict the outcome. This reinforces understanding of program execution and builds debugging skills.

The "put in order" test (Fig 3c) evaluates the Synthesis level. Typically, the "put in order" test consists of a brief description of the function a piece of code is supposed to perform, followed by numbered lines of code in arbitrary order. The student is required to reorganize the lines of code in such a way that the resulting order achieves the objectives set out in the initial description. To successfully complete these types of test, the student must apply what he/she has learnt in knowledge and understanding levels to complete a predefined task. The critical thinking skills of interpretation, analysis, evaluation and inference are required.

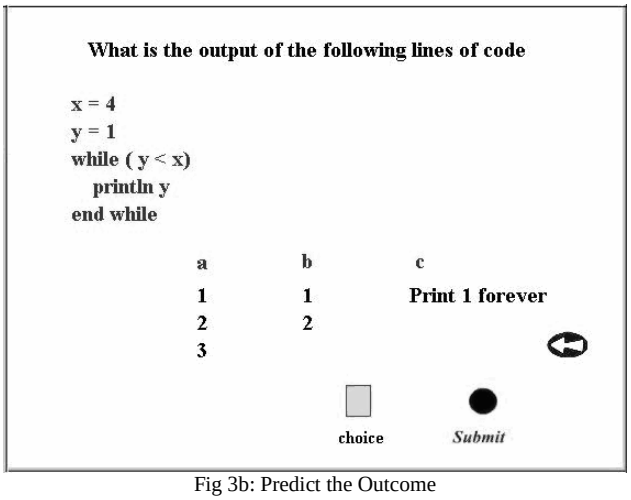

Fig 3b: Predict the Outcome

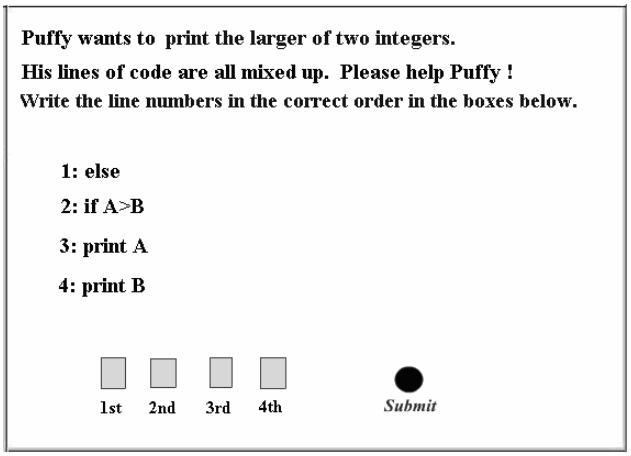

Fig 3c: Put in Order

4. Course development

Macromedia Flash content on the Web has become almost commonplace with a growing number of Websites mixing Macromedia Flash and HTML to add animation, sound, and video to create interactive pages. This interactive, web-based course was developed using Macromedia Flash MX. Macromedia Flash MX was used as the e-learning authoring tool for building a media-rich learning environment.

For this system the lesson content of each unit was subdivided into modules and each was created as a separate flash file. Each Flash file (*.fla*) has at least three (3) basic layers; action, text and sound/images. Other layers required included a background layer and alternate text layers. These layers served to keep the content

separate and organized. To add interactivity to the Flash applications ActionScript 2.0 was used.

The course website consists of frames (Fig 4).

- The course navigation bar is in the top frame (under the course title)
- The unit navigation frame on the left
- The content frame on the right

4.1 Course Navigation Bar

The course navigation bar contains direct links to the home page and also the Units presented in the course (CXC Information Technology). As part of the instructional design of the course, the course content was divided into Units with Topics within those Units. In this course, we were working with a predefined syllabus; two of the Units were Programming Units and the other Units covered other information technology content. A course map; which is a map of the entire site designed to show how the pages relate to each other and how they work together as a whole; was developed. The course map captures all primary and in some cases secondary navigation.

The main responsibility of the course navigation bar is to update the unit navigation frame and hence allow access to unit content menus. It is the only area that allows movement between the units.

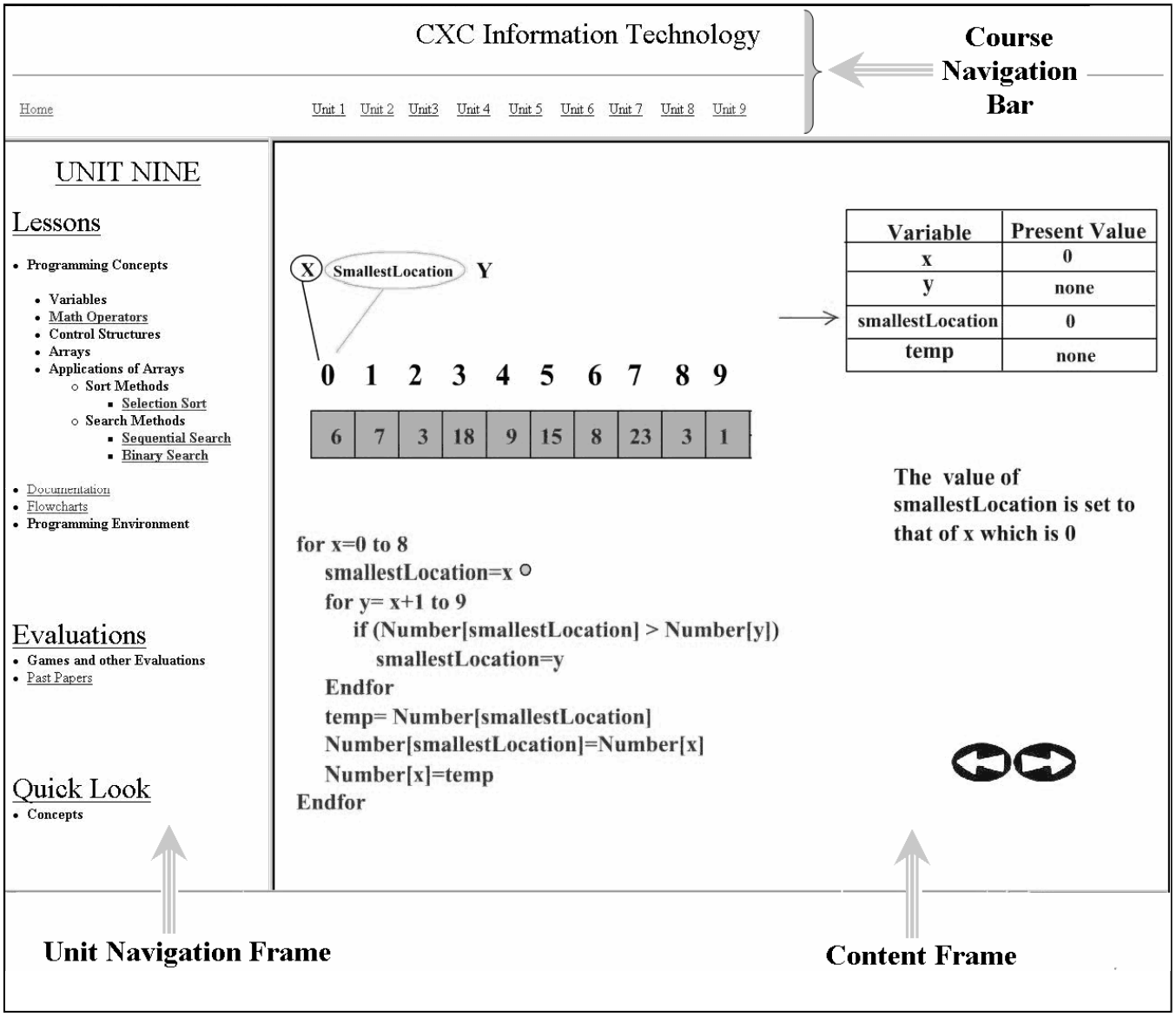

Fig 4: Layout of the Course Website

4.2 Unit Navigation Frame

For all units the navigation frame is organized in the same way. The frame has a header which displays the present unit the user is interacting with. This serves as a point of reference especially when switching between units directly. The content of the frame is split into three sub sections:

- Lessons
  The "Lesson" subsection is where the main ideas for each subtopic in the unit are shown. A student chooses the content he/ she would like to view. The content selected is then displayed in the content frame. By using a shallow tree structure, the complexity of the process to find the data required is reduced. Usually one or two sublevels need to be traversed in order to find the required content.
- Evaluations
  Each unit's evaluations are housed here. The evaluations are designed to test individual concepts, sections of the unit or the unit as a whole. It allows for self assessments and mapping of a student's progress through a lesson, unit and the course as a whole.
- Quick Look
  The idea behind the "Quick Look" section is to provide an easy way for the student to look up relevant terms related to the unit. Once selected, the basic overview of the concept is presented to the student, as well as links to the relevant lessons and other material if necessary. Quick look content was developed with the intention to provide bare facts and definitions to be available at a student's fingertips. Access is easy and fast. Since the audience is a diverse Caribbean one, explicit definitions are presented with no assumption to background and previous knowledge being made. Quick look is intended to be used during review or revision sessions as well as a quick way to clarify unknown terms while perusing a unit or lesson.

4.3 Content frame

The content Frame is the window where the lesson content is displayed and the student's interaction with the content is performed. On first visiting the site, or by clicking on the "home" link in the course navigation bar, a brief set of instructions on how to use the site is displayed. Selection of a lesson topic from the unit navigation bar brings this area to life. All content from movies and slides to evaluations and quick look information are displayed, and interacted with from this frame.

5. Conclusion

Programming has traditionally been an area that is a challenge to most students. Static traditional teaching methods are often passive; they do not engage the student in active learning within a real world context. This paper presented a web-based learning environment that uses visualization techniques in an innovative manner to provide visual lessons on programming structures and control flow. Lessons are reinforced with game-like activities that the young digital generation can relate to and have fun with while learning the content. His course was evaluated by students, teachers and instructional designers. The feedback was positive. Students loved the look and feel of the course; they could identify with the visual content. The lessons flowed in a logical manner and concepts were indeed reinforced by using the narrative descriptions and pseudocode. The gaming activities were enjoyable and goal directed. The Quick Look frame that allows easy lookup of concepts is an interesting feature that students appreciated. Caribbean teachers and students can benefit from this course as the teacher can use it to structure his lessons and complement what is being taught in the classroom; the course can also be used as a stand-alone course for students.

References

[1] Bloom, B., Englehart, M. Furst, E., Hill, W., & Krathwohl, D. (1956). Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain. New York, Toronto: Longmans, Green.

[2] Duffy, T. M., & Cunningham, D. J. (1996). Constructivism: Implications for the design and delivery of instruction. In D. H. Jonassen (Ed.), Handbook of research for educational communications and technology. New York: Macmillan.

[3] Kelleher, C., Pausch, R., (2005). Lowering the barriers to programming: A Taxonomy of programming environments and languages for novice programmers, ACM Computing Surveys, Vol.37, No. 2, p.83-137.

[4] Prensky, M. (2001) Digital Game-based Learning, McGraw- Hill.

[5] Rajarvivarma, R. (2005). A games-based approach for teaching the introductory programming course, ACM SIGCSE Bulletin, Vol.37, No.4, p.98-102