Educational Tool for Understanding Algorithm Building and Learning Programming Languages

Anshul K. Jain, Manik Singhal, Manu Sheel Gupta

Abstract—**This research paper presents a new concept of using a single tool to associate syntax of various programming languages, algorithms and basic coding techniques. A simple framework has been programmed in Python that helps students learn skills to develop algorithms, and implement them in various programming languages. The tool provides an innovative and a unified graphical user interface for development of multimedia objects, educational games and applications. It also aids collaborative learning amongst students and teachers through an integrated mechanism based on Remote Procedure Calls. The paper also elucidates an innovative method for code generation to enable students to learn the basics of programming languages using drag-n-drop methods for image objects.**

Index Terms—**Code Generation, Collaborative Learning, Educational Software, GUI Programming, Programming tools.**

I. Introduction

Education is the most important ingredient of life. It gives an individual, capability to address the world in all walks of life. And, with changing times, technology is playing a key role in imparting education by making information easily accessible in a fun and an engaging manner. Educational software applications have been playing a pivotal role in imparting education. However, it is quite ironical that there has been very little evolution in the field of teaching algorithms and programming in an interesting manner through technological aided learning [14]. The current methodology to teaching programming and algorithms still requires large volumes of text reading, which at times makes the entire scope of learning monotonous and boring. Students generally find it difficult to understand the importance of developing algorithms and logic building exercises. It is important to note that if logic-building techniques are developed, programming becomes much easier to work with. The ability to think logically is integral and needs to be enriched.

There are applications that teach algorithms in interesting ways. 'Scratch' [12], a block-based programming language, is designed to facilitate media manipulation for novice programmers. It has been engineered using Squeak, a programming language based on Smalltalk implementation [4]. 'Scratch' indeed has the qualities of intriguing a beginner into developing programs and learning objects with its drag and drop features based on Lego blocks.

However, Scratch does not provide the capabilities of peer-to-peer learning, sharing and critiquing of ideas during the development of algorithms and programs, an integral component of engineering in the *flat* world. It thus needs an enhancement in its approach towards learning, which includes sharing of resources, knowledge and ideas.

In this paper, we focus on the development of a collaborative framework for sharing ideas and implemented codes so that students can engage with their peers during application development cycle. We have also build this technology with a key priority on making it independent of underlying application format in order to prevent beginners from switching between applications, which adds a lead time and reduce their concentration dramatically. It is desirable to develop tools of the 21st century on the lines of an 'information appliance' a term fabricated by Jef Raskin [10]-[11] that can provide multiple functions within single application window, very much like an iPhone. Another major improvement in the framework is the code generation capability in multiple programming languages for the algorithm constructed using dragging and dropping of images. This enables the students to relate their logic and program from an application development perspective thereby motivating them to use these programming languages more effectively.

Section II of the paper explains the graphical user interface design of the framework, and section III describes the code generation module. Section IV focuses on the tool's capability to enable collaborative learning, and section V supplements it with descriptive images and workflow diagrams. In section VI, we conclude the paper with an aggregation of concepts for engineering learning tools for the 21st century.

II. User Interface

A. Drag-n-Drop Blocks

The application has various categories of blocks like variables, operators, flow of control etc. Each block category is distinct and its sub categories use the same structure as their main category. Each category has a finite number of blocks, which can be dragged and dropped to create a program. Thus, a program's structure can be designed by arranging desired blocks in a logical arrangement with detachable pieces of code, which can be integrated easily.

SVG [5] or Scalable Vector Graphics is used to create XML based two dimensional vector graphics. It is very important to find a responsive and an error-free solution to user's interaction with the computer. Without this set of capabilities, an application is generally considered useless,

Manuscript received July 16, 2010.

Anshul K. Jain is with the Computers Department, Netaji Subhas Institute of Technology (Delhi University), Sec-3 Dwarka, Delhi 110075 India (phone: +91-9868209309; e-mail: anshul.jain@ nsitonline.in).

Manik Singhal is with the Computers Department, Netaji Subhas Institute of Technology (Delhi University), India (phone: +91-9873079495; e-mail: manik.singhal@ nsitonline.in).

Manu Sheel Gupta was with Information Technology Department, Netaji Subhas Institute of Technology (Delhi University). He is the founder, Software for Education Entertainment and Training Activities (SEETA), India (e-mail: manu@seeta.in).

irrespective of its functional offerings. The dragging and dropping [13] of blocks were implemented with a design centered towards its ease of use and its sensitivity towards the response from the user.

There are two key challenges in this implementation. First, the lead times introduced between the movements of mouse and image can deteriorate the usability of the tool making it inconvenient for the user. Second, the code for this feature implementation should be optimized to cost minimum CPU cycles. We overcame these challenges with the choice of our programming language for the implementation of this feature. Python [7]-[15] scored well in comparison to the other programming languages in its capability to help us arrive at a good conclusion on the above challenges. Certain events pre-defined in PyQt [6]-[16] library for python and Nokia's Qt framework were re-implemented, and were connected using the blocks.

B. Script Window

The script window is a helping hand for the language module. The script generator within the text-pane displays the syntax when a particular block is clicked. It is very important that we keep the software lightweight and modifiable for enhancements and its reusability.

An important feature of this functionality is that the text inside the script editor is made editable. Also, notes about the usage of blocks that the student wants to save for future references can be taken as its description appears in the editor. After the next block is selected for preview, a prompt to save the code of the previous block comes up, and on saving the code, its text is written into a file inside the programming language module. Using this feature, the users can develop a help manual of various blocks in a single programming module. Thus, the program can be personalized using this key functionality.

C. Web Browser

It is helpful that the students are able to find the information when they need it. Also, it is not good to have the students switch back and forth from application to browser and vice-versa. Keeping these key priorities in mind, a web browser that provides students access to rich information was developed within the application framework.

By providing a web-browser inside the application framework [2], students never leave the environment they work in, and feel enthusiastic about the subject matter. The browser together with the editable script window is a key attempt towards improving access to information. Another key aspect to be looked into is to develop a clear strategy for feature inclusion in the browser. Browsers are navigational tools, and ought to be light- weight to improve usability.

A number of layout engines are available for the web pages. Presto, Gecko and WebKit are the commonly used ones. The PyQt bindings for Python incorporate the WebKit layout engine. It is interesting to note that the popular browsers like 'Safari' and 'Chrome' use WebKit as their layout engine. And, these two layout engines are used amongst the fastest browsers available today in the consumer market. Thus, it was very encouraging to use Webkit as the layout engine for our application framework. In 300 lines of code, we developed a browser that implements functions for keeping history of navigated pages, Reload, Stop and Home widget.

The 'url-changed' function defined in PyQt for WebKit toolkit was used in the implementation of the browser. This function is called as and when the user changes the URL in the address bar. The other widgets have action events associated with them like Back, Forward, Reload, Stop and going to the Home page. Below is a code snippet of the url-changed function:

```
def url_changed(self):
page = self.ui.webView.page()
history = page.history()
if history.canGoBack():
self.ui.back.setEnabled(True)
else:
self.ui.back.setEnabled(False)
if history.canGoForward():
self.ui.next.setEnabled(True)
else:
self.ui.next.setEnabled(False)
url = self.ui.url.text()if url[0:4] is not
'http':
url = 'http://www.'+ self.ui.url.text()
self.ui.webView.setUrl(QtCore.QUrl(url))
```

This browser without any further enhancements was tested and compared against other available browsers on the same machine. The benchmark was based on cumulative scores measured in operations per second, rendered frames per second and JavaScript rendering speeds.

Table I Benchmark results for different browsers

Browser	**Points scored in Benchmark[8] (more is better)**
Our Browser	2304
Internet Explorer 8	871
Mozilla Firefox 3.5.7	2025
Mozilla Firefox 3.6	2854
Opera 10.10	2289
Safari 4.0.4	3183

III. Code Generation

Students will get an understanding of algorithm building and syntax of particular languages after experimenting with the application for a while. The next important step is learning to build real codes. This aspect of programming has been addressed up by the application. There are number of programming modules available for the application. In every programming module, there are files that connect each block with its appropriate syntax and important explanation in reference to the selected language. When the student clicks on any block image, the contents of the text file corresponding to that block and the current selected language gets displayed in the script window and the necessary explanation is displayed too.

After a code using blocks has been created and run, pseudo-code is generated and displayed in the script window. This gives the students an overview of an implementation.

ISBN: 978-988-17012-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

As the student builds up an algorithm using blocks, the name of the corresponding block file is logged in a separate file, say file 'A', along with the number of parameters required by that block and location of these parameters (an index to another file 'B'). The actual values entered by the student for the parameters needed by a block get stored in file 'B'. A separate file is maintained corresponding to each block containing the pseudo code for that block.

When the student executes the algorithm, file 'A' containing sequence of blocks in algorithm is read line by line and the data from file containing pseudo code corresponding to block in last read line is retrieved and stored locally. The values of parameters corresponding to this block are also read from file 'B' and stored locally. These parameters are then inserted at appropriate positions in the retrieved pseudo code and the final pseudo code for this block gets displayed in script window. This process of generating pseudo code is repeated for each block and a complete pseudo code for written algorithm is displayed in the script window.

IV. Collaboration

With the increasing social activities, there should be some way through which students can bring this aspect to their learning also. Learning in collaboration is most effective form of learning. It improves quality of work and helps students come out of their own logical world. This tool fulfills this requirement with the help of Python. RPC (Remote Procedure Call) connections belong to the niche of Operating Systems[1]-[3]. After a student has created code he can send the code to connected peers via LAN or Internet using RPC connection. The other student receives the code in the block form in his window exactly as it was created, which he can then edit, or review and then send back to the sender. With this feature student can not only share their creation and gain appreciation, but also allow teachers to review/edit their work from anywhere when necessary [18].

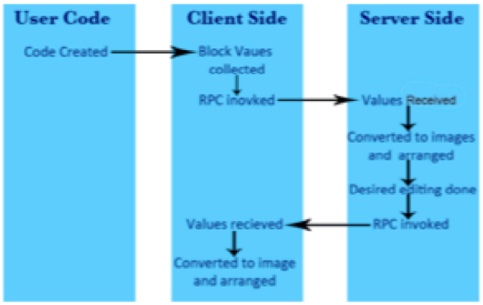

Fig. 1 Schematic flow of control using inbuilt sharing

This tool does not need any form to send request to server; it can send this information directly from the interface. We have used Python RPC over other RPC available to provide faster connection and better troubleshooting in case of any connection failure. Students are also provided with the facility of multicasting their work to many peers with just sending/receiving information through multiple ports. This asset can be used to connect a student with many of his friends at a time and share his creation simultaneously with multiple users. As the application is aimed at students new to computer, security becomes very important aspect of networking. When a RPC connection is made, it always sends the identity of the client computer to the server. With a simple function defined at the server end the authenticity of the client can be verified and any anonymous connections can be straight forward refused. Thus security issues are also effectively administered.

V. Explanatory Images

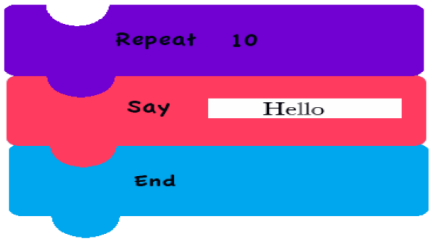

Fig. 2 An overview of a basic code designed by student

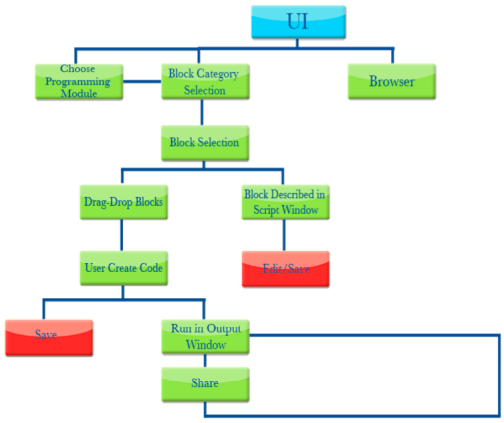

Fig. 3 A schematic flow diagram of working of Aura

Fig. 4 An approximate layout of different components

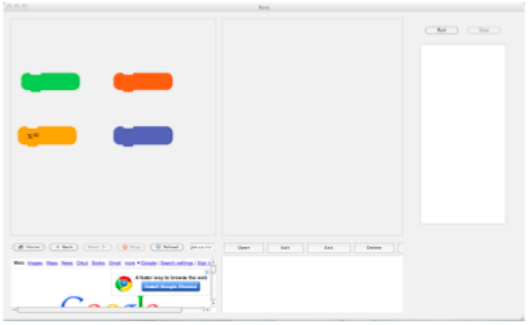

Fig. 5 A basic structural view of application

ISBN: 978-988-17012-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

VI. Conclusion

This paper comprehensively presents an educational tool for learning algorithms and programming languages. The tool's user interface caters to all the needs of students at single place; it includes web browser, script window, code generation module and sharing support. The tool has a feature of inserting programming modules that can help students to learn multiple programming languages using single interface helping them understand the importance of basic logic building and how different languages have different constructs for same function. The code generation module is necessary to let students learn different programming languages constructs. The sharing support provided for collaborative learning has been implemented keeping in mind its ease of use, faster connection support, multicasting capability and other security considerations. All these features combined are important from students' learning perspective and thus we went on to implement these requirements in a single package as a software application for the use by students.

References

[1] Andrew and Bruce, Feb 1984, 'Implementing Remote Procedure Calls', *RPC connections,* (ACM transactions on Computer System, vol. 2), Available: http://citeseerx.ist.psu.edu

[2] T. Antero; M. Tommi; I. Dan; P. Krzysztof; Jan. 2008, 'Web Browser as an Application Platform: The Lively Kernel Experience', *Web Browser as application tool*, (SMLI TR-2008-175), Available: http://portal.acm.org

[3] B. Brian; A. Thomas; L. Edward; L. Henry; Feb 1990, 'Lightweight Remote Procedure Call', *RPC connection Optimization*, (ACM transactions on Computer System, vol. 8), Available: http://citeseerx.ist.psu.edu

[4] I. Dan; K. Ted; M. John; W. Scott; K. Allan; 1997, 'Back to Future: the story of Squeak, a practical Smalltalk written in itself', *Overview of Squeak Programming Language*, (OOPSLA 1997, pp. 318-326), Available: http://portal.acm.org

[5] B. Daniel; V. Ronald; C. Christopher; Oct. 2004, 'SVG for Educational Simulations', *SVG to developing graphics for students*, (Conference on Information Technology, pp. 43-49), Available: http://portal.acm.org

[6] David, 'PyQt Whitepaper', *PyQt whitepaper*, Available: http://riverbankcomputing.co.uk

[7] Frank, 'Python in Education: Raising a Generation of Native Speakers', *Python usefulness to programming learners*, Available: http://ojs.pythonpapers.org

[8] FutureMark Browser Benchmark Service, Available: http://service.futuremark.com/peacekeeper

[9] Guilherme, 'PyGTK, PyQt, Tkinter and WxPython comparison', *Comparison in various GUI python bindings*, Available: http://ojs.pythonpapers.org

[10] Ingo and Mirko, Oct. 2003, 'Test Suite Design for Code Generation Tools', *Ways to test code generators*, (Automated Software Engineering, 2003 Proceedings), Available: http://ieeexpore.ieee.org

[11] Jef, Sept. 2003, 'User Interface Designers, Slaves of Fashion', *General talk on User Interfaces*, (Queue, Vol. 1, Issue 6), Available: http://portal.acm.org

[12] M. John; P. Kylie; K. Yasmine; R. Mitchel; R. Natalie; 2008, 'Programming by Choice: Urban Youth Learning Programming with Scratch', *General Description of scratch application*, Available at: http://web.mit.edu/~mres/papers

[13] Kori, Mar. 2001, 'Drag-and-Drop versus Point-and-Click Mouse Interaction Styles for Children', *Drag-Drop versus Point Click methods*, (ACM transactions on Computer System, vol. 8), Available: http://portal.acm.org

[14] Z. Mario; B. Ivana; O. Marin; 2008, 'Enhancing Software Engineering Education a Creative Approach', *Imparting software education through projects*, (International Conference of Software Engineering, pp. 51-58), Available: http://protal.acm.org

[15] Mark, Sept. 2009, 'Learning Python', *Beginner's book to python*, 4th edition, O'Reilly Publications, ISBN 978-0-596-15806-4

[16] Mark, Sept. 2009, 'Rapid GUI programming with Python and Qt', *Learner's Guide to PyQt Programming*, 2nd Edition, Prentice Hall, ISBN 978-0-13-235418-9

[17] Mitchell, Jan 2010, 'Bioinformatics Programming Using Python', *Book on data manipulation through Python*, O'Reilly Publications, ISBN 978-81-8404-898-8

[18] Robin, 'A Web-based System for Dynamic Teacher-Student Interaction in a Classroom Setting', *Teacher-Student interaction through web in classroom*, Jab 2007, (Journal of Computing Sciences in Colleges, Vol. 22, Issue 3), Available: http://protal.acm.org

[19] L. Simon; J. Jeo; D. Edd; W. Dave; Dec. 2001, 'Programming Web Services with XML-RPC', *Web services using XML-RPC connections*, O'Reilly Publications, ISBN 81-7366-207-X

[20] Timothy et al., Apr. 2008, 'Towards Universal Code Generator Generation', *Design paradigms for generic code generators*, (Parallel and Distributed Processing, pp. 1-8), Available: http://ieeexplore.ieee.org