Introducing Programming using "Scratch" and "Greenfoot"

H.S.Chandrashekar[1], A Geetha Kiran[2], B.Uma [3] , Sunita P[4]

[1]Department of Information Science and Engineering, Malnad College of Engineering, Hassan, India
[2]Department of Information Science and Engineering, Malnad College of Engineering, Hassan, India
[3]Department of Information Science and Engineering, Malnad College of Engineering, Hassan, India
[4]Department of Computer Science and Engineering, Malnad College of Engineering, Hassan, India

[1]hsc@mcehassan.ac.in
[2]agk@mcehassan.ac.in
[3]bu@mcehassan.ac.in
[4]ps@mcehassan.ac.in

Abstract:

Programming is claimed to be the "literacy of the twenty-first century", that it helps one to become a more empowered citizen. Although programming is a much valuable skill, novice students often find it very difficult and not interesting. "Scratch" and "Greenfoot" are two educational integrated development environments aimed at learning and teaching programming. Both software applications are freely available. Scratch is a graphical programming language developed at Massachusetts Institute of Technology, Media lab. Greenfoot aims at making the use of the standard language, Java, easy by providing a custom-designed environment that removes much of the complexity commonly associated with object-oriented programming. The proposed approach is to introduce programming through "Scratch" and "Greenfoot", where students experience programming by developing simple animation, games or simulations. This helps them to develop confidence and an interest to learn programming, thus setting a strong base to delve into programming using textual languages. The suggested method has been proven to provide a positive and supportive atmosphere in which students have acquired good programming skills.

Keywords- Programming, Scratch, Greenfoot

1. Introduction

Programming can enable scientist and engineers to work 10 to 100 times faster and to come up with solutions that are more creative. In terms of competencies for the twenty first century, programming skill is one of the important skills needed to survive in the information age in a global society [10]. Most teachers see the struggles of their students as they battle to gain grips with most basic programming skills.

Inherently, programming is not a difficult skill to acquire. Programming is all about problem solving, mathematical ability and common sense. This made us to observe and analyse precisely what makes programming difficult for large number of students.

We have been intrigued and inspired by the way children handle electronic gadgets and build new structures with a set of blocks. This is possible only because it is interesting, exciting and has immediate visualization. We wanted the process of programming to have a similar feel. To bring that element of excitement, it was decided to introduce graphical programming environment before beginning with regular text based programming languages.

2. A Brief look into Previous works

Research in the field of computer science education has highlighted that many students lack problem solving and computational thinking skills [3]. Calder et al.[6] have declared "Scratch software proved to be an engaging and relatively easy-to-use space for problem solving, which at the same time provide a worthwhile and motivating programming environment to explore mathematical concepts" . Filiz and Yasemin [2] surmise that programming skills are essentially needed to survive in the information age in a global society.

Further, it is observed that visual programming is more user-friendly and effective than textual programming. Students find visual programming using Scratch more interesting, motivated, less bored and away from the syntax of textual programming languages. Realising the powerful dimensions of Scratch software like being a networked and media-rich programming environment, [11]Maloney et. al. hypothesised that if learners work on meaningful Scratch projects, such as animated stories, games, and interactive art, they will develop technological fluency, mathematical and problem solving skills. Scratch is being used in Harvard University as an introduction to programming for university students [9]. Statistics from the Scratch web site indicates there is a wide age range of users for this tool [15].

Historically, a program has been viewed as a logical procedure that takes input data, processes it, and produces output data. However, the real time application programming challenge was seen as how to write the logic, not how to define the data. This need lead to Object-Oriented programming(OOP) that focuses on the objects and how to manipulate it. Learning OOPs was an exigent task for the undergraduate students [4]. But it could be taught in a more interactive and integrated environment with tools like: 'Blue', 'BlueJ', 'Greenfoot', that makes easy to teach, learn and practice OOPs concepts particularly for novices [1]. It is fun learning OOPs through games and animations [12, 13].

Greenfoot is an integrated tool, which aims at learning and teaching programming to young novices. It helps users to design and visualize important concepts of object-oriented programming. Further, the tool allows teachers to introduce the most important and fundamental concepts of object orientation in an easily understandable manner [5].

Thus, the educational benefits of Scratch and Greenfoot environment for students are also obvious and teachers should consider these potentials in their courses.

3. Scratch and GreenFoot

Scratch and Greenfoot are integrated educational software development environments aimed at learning and teaching programming to novices.

Few guiding principles of both Scratch and GreenFoot are:

- To "lower the floor" for programming, so that novice can get started earlier.
- Desirable to be high level enough to do something useful and interesting, but low-level enough to allow flexibility, and to allow composing user's own actions. Thus, a programmer is not only a user of a language, but can also be a designer. This power of abstraction is one of the most fascinating things in computing.
- Aim at removing or hiding accidental complexities, letting users get on with the job, and making them work with fundamental constructs.
- Designer team has worked hard to find right balance between scope of functionality and flexibility on one hand, and simplicity on the other.
- To value simplicity very highly and not being sucked into feature escalation.
- Engaging and empowering the user is the most important common theme for both [7]. They have strived to engage the user by allowing them to write programs about things that connect with their interests (stories, games, simulations, etc.) in contrast to more conventional programming instruction where examples might be things like generating prime numbers or sorting a list of numbers [7].

These tools are intended as a way to introduce programming before beginning with a formal programming language.

A. Scratch

Scratch is a tool, which has taken inspiration from Logo [14]. It is a visual-based tool and users are encouraged to create programs by simply tagging together blocks provided to create their own program. Figure 1 shows screen shot of the opening main window of Scratch. Scratch sprites are objects that own state and behavior, but have no classes or inheritance.

The Scratch grammar is based on a collection of graphical "programming blocks" that can be snapped together to create programs as shown in Figure 2. One can start by simply selecting few blocks, snapping them together in different sequences and combinations to see what happens. There is no violation of syntax of traditional programming languages. The time to get started is low, but provides ample opportunities to build increasingly complex projects.

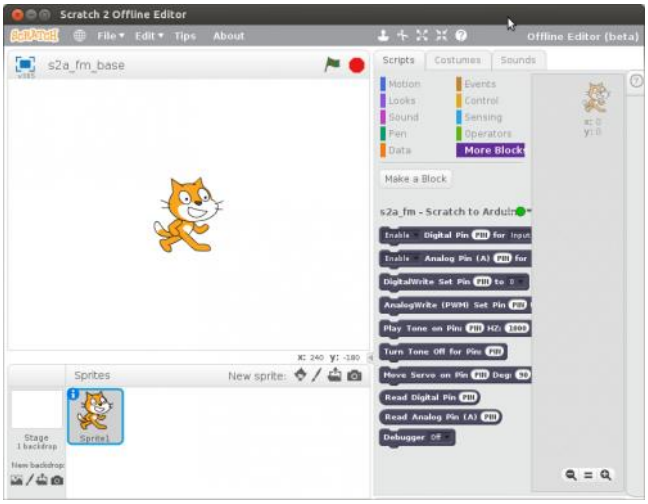

Figure 1

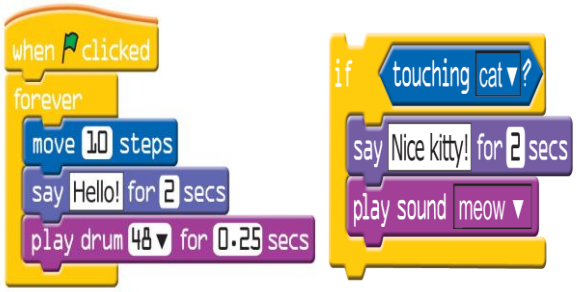

Figure 2

Scratch blocks are shaped to fit together only in ways that make syntactic sense. Control structures (like repeat and forever) are C-shaped to suggest that blocks should be placed inside them. Blocks that output values are shaped according to the types of values they return: ovals for

numbers and hexagons for booleans. Conditional blocks (like if and repeat-until) have hexagon-shaped voids, indicating that a boolean is required.

The Scratch web site has turned out to be a vibrant online community, where people share, discuss and remix one's project with another.

B. GreenFoot

Greenfoot is an educational integrated development environment aimed at learning and teaching OOPs. Greenfoot combines graphical, interactive output with programming in Java, a standard text-based object-oriented programming language.

Figure 3 shows Greenfoot's main window, with a scenario, which is Greenfoot's term for a project. The main part of the window shows the Greenfoot world, the area where the program executes. On the right, class diagram is shown that visualizes the classes used in this scenario and their inheritance relations. Once the object has been created, it can be placed into the world.

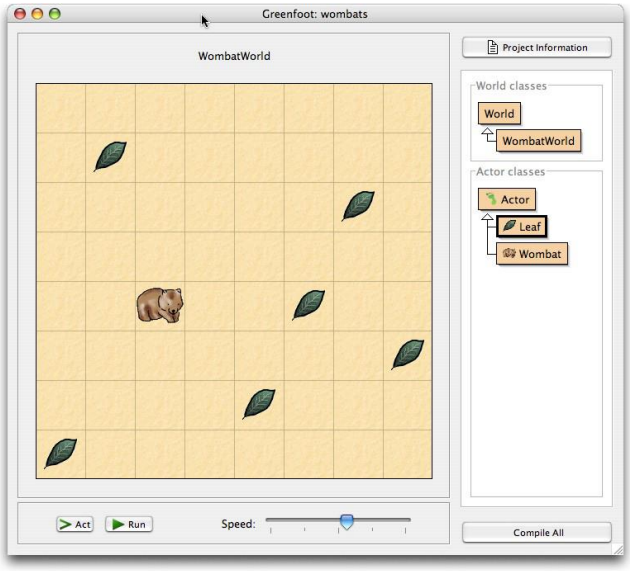

Figure 3

When an object is in the world, a right-click shows all the object's public methods for all available actions it can perform. Figure 4 illustrates a sample of an object's public methods. Once a method has been selected, it executes and the effect is visible onscreen immediately.

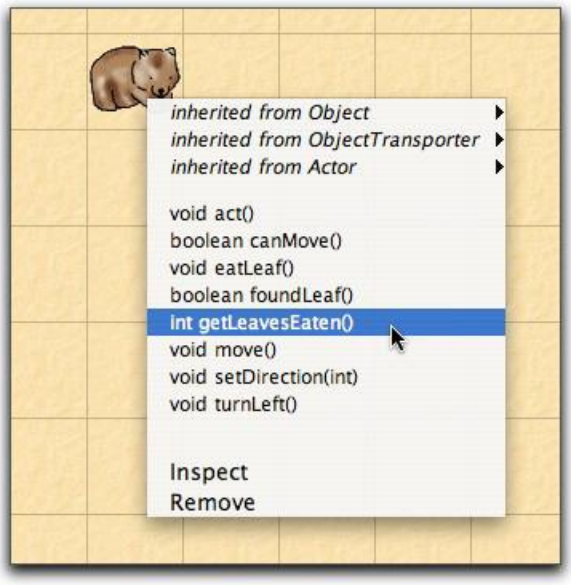

Figure 4

The language used to program is standard Java. Greenfoot internally uses the standard Java compiler and the standard virtual machine (JVM) to ensure full conformance with current Java specifications. GreenFoot embeds Java language in a graphical environment. Instances of actor classes can be simply dropped into world scenario and instances can be easily controlled using available methods or user can write his own methods.

One of the goals of Greenfoot is a design that explicitly visualizes important concepts of object-oriented programming. Greenfoot has the Java object model and introduces OOP. It would be good to hide language baggage like – edit, compile, run and provide more interactivity so as to engage students and make learning experience more enjoyable. All these desirable features are provided in GreenFoot, which can be used before progressing to Java or an equivalent.

4. Our Approach

The methodology we adopted to introduce procedure and object oriented programming languages to undergraduate students are described in this section.

A. Introducing Procedure Oriented Programming language

Although students in third semester would have completed first programming course in previous semester, good number of students would not have acquired skills to carry out reasonably complex programming. Hence, it was decided to introduce "Scratch" software to them and make them carry out a small interesting project.

Games-based learning (GBL) is the use of computer games for learning and educationalists hope to utilize the benefits of it within the class [8]. Computer games/animations are

an exceptionally popular medium across all age groups.

A brief introduction to Scratch was given in regular laboratory class. Students got really excited about Scratch software and its capability to develop simple animations and games with ease. Students who still had not understood the logic of coding and use of control structures also could better understand and get a feel of the same. This gave them confidence that they can also do programming. Further, it even helped them to think creatively.

The best way to learn is by getting hands dirty. Hence, all students were asked to develop a simple game or animation of their choice. The faculty coordinator kept track of the progress and ensured that all students carry out the task assigned.

The demonstration of their developed game/animation was held in the presence of few final year students and all faculty members and. The final year students played the role of judges, giving appreciations for the efforts and few suggestions. This increased the self confidence and morale of all students. Now, most students are confident of their programming skill and have confidence to develop good projects in future semesters.

B. **Introducing Object Oriented Programming Language**

Object-oriented programming is the mainstream in computing industry and Java is one of the most widely adopted languages in software industry. However, Java has steep learning curve. To circumvent this problem, it was planned to adopt a GUI-centered approach to teach object-orientation concepts and Java coding. Before letting students to create any simple application, it is important to ensure that students truly understand the basic concepts of object-oriented notions and have ability to think in the object-oriented way. An assistive software environment called Greenfoot was used to achieve this.

Java is taught to second year students and GreenFoot was briefly introduced to them in a laboratoryclass. A quick demonstration of a simple game development and required java coding was shown, which gave enough confidence for all students to play with GreenFoot. They were also excited to understand how a game and an animation is developed. To ensure that students pick up the basic concepts of developing game using object-oriented concepts, they were instructed to develop a game of their choice.

A demonstration of games developed by all students was arranged. Faculty members and final year students were invited to participate in the session and give valuable remarks for each game developed. Most students enjoyed and experienced good learning in the process of game development. They got acquainted with object oriented concepts and programming. This made their journey into actual Java coding more easy and interesting.

5. Reflections

The idea of introducing programming using graphical tools was successful. All students enjoyed the process of learning and developing a simple game as the outcome of learning. The feedback from both students and teachers are briefed here.

A. Students' Views

Students had very exciting hands-on experience while learning Scratch and Greenfoot tools. Scratch not only helped them to feel programming better but also improved their logical thinking and creativity ideas. It changed their way of understanding programming concepts. Students found OOPs concept simpler and adoptable with Greenfoot. In addition, it gave them enough confidence take up good projects in their coming semesters.

At the end of the students' project demonstration; they expressed their learning, suggestions and their views towards Scratch and Greenfoot tools in a questionnaire. The questionnaire reflected their general opinion and attitude to Scratch and Greenfoot. Since, Scratch was their new experience with the world of programming, the anticipated results were highly optimistic. Compiling the students' feedback on Scratch and Greenfoot programming tools, some of the opinions of the students were :

It really made coding easy and helped us to develop small games in an interesting way

Could visualize the OOPs concepts using Greenfoot software and learn faster

Able to create something innovative on our own.

Scratch was like a simplified C like language where you can drag-n-drop elements instead of writing them.

Gained confidence in programming

B. Teachers' Views

Teachers expressed positive feeling about the use of Scratch and Greenfoot tools to introduce programming to students. They strongly opined that most students could better learn programming using these tools. The pleasant surprise was the good and creative simple games developed by all students. They appreciated the interest, involvement and committement shown by students in doing the game development. Teachers expressed their gladness that they too were benefitted and had great time working with Scratch and Greenfoot. Few reflections of teachers are:

Students learnt programming aspects much faster than other programming language with Scratch and Greenfoot

Greenfoot is a brilliant instruction tool that helped teachers to explain programming and OOPs concepts in a much easier way than before to their students.

Students who were academically weak were also doing really well with Scratch.

Students were found to be excited about the work they are doing and even worked harder than ever.

Conclusion

Scratch is rich graphical programming tool, which enabled students to design and develop programs by using drag-and-drop facility. Greenfoot supported blocks-oriented visual programming helping students to learn OOPs concepts in a much easier way. Scratch and Greenfoot keeps students interested and reinforces the programming principles. Students with little or no programming background were able to learn programming with less difficulty. Familiarization with Scratch and Greenfoot not only enhanced students' experiences to programming, but also decreased the anxiety to take up projects in their forthcoming semesters. Thus, our proposed approach provides a constructive and encouraging atmosphere in which students can gain knowledge of the object-oriented concepts and programming. Further, the approach of introducing programming using Scratch and Greenfoot was highly positive in achieving higher levels of Bloom's Taxonomy: apply, design and create.

References

1. Kölling, Michael, ' Lessons from the Design of Three Educational Programming Environments: Blue, BlueJ and Greenfoot.', International Journal of People-Oriented Programming, 4 (1). pp. 5-32,2016.
2. Filiz KALELIOĞLU1, Yasemin GÜLBAHAR2, 'The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners', Perspective, Informatics in Education, Vol. 13, No. 1, 33–50, © 2014
3. Papadopoulos, Y., Tegos, S. Using microworlds to introduce programming to novices. In: Proceeding PCI '12: Proceedings of the 16th Panhellenic Conference on Informatics, Piraeus, Greece, 180–185,2012
4. M. Sivasakthi and R. Rajendran, 'Learning difficulties of 'object-oriented programming paradigm using Java': students' perspective', Indian Journal of Science and Technology Vol. 4 No. 8 ,Aug 2011.
5. Michael Kölling King's College London, 'The Greenfoot Programming Environment", Article in ACM Transactions on Computing Education · November 2010
6. Calder, N. Using Scratch: An integrated problem-solving approach to mathematical thinking. Australian Primary Mathematics Classroom, 15(4), 9–14,2010
7. Programming Environment', Article in ACM Transactions on Computing Education , November 2010.
8. Cooper, S., K¨olling, M., Maloney, J., and Resnick, M. Alice, 'Greenfoot and Scratch – A discussion', ACM Trans. Comput. Educ. 10, 4, Article 17 , November 2010.
9. Groff, J., Howells, C. and Cranmer, S. The impact of console games in the classroom: Evidence from schools in Scotland, 2010.
10. Malan, D.J. Reinventing CS50. In Proceedings of the 41st ACM technical symposium on Computer science education. ACM, pp. 152–156,2010.
11. Nelson, J. Celebrating Scratch in libraries: creation software helps young people develop 21st-century literacy skills. School Library Journal, 20–21,2009.
12. Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., Resnick, " Scratch: a sneak preview" . In: Second International Conference on Creating, Connecting, and Collaborating through Computing. Kyoto, Japan, 104–109, 2004.
13. Micheal Kolling, 'Introduction to Programming with Greenfoot_ Object-Oriented Programming in Java with Games and Simulations' - text book
14. Lu Yan, 'Teaching Object-Oriented Programming with Games', Sixth International Conference on Information Technology: New Generations,2009.
15. Papert, S. Mindstorms: children, computers, and powerful ideas, Basic Books, Inc, 1980.
16. MIT, Statistics on scratch users , 2011. http://stats.scratch.mit.edu/