Comparing Alice, Greenfoot & Scratch

Sally Fincher (moderator)  
School of Computing  
University of Kent  
Canterbury, Kent, UK  
+44 (0)1227 824061  
S.A.Fincher@kent.ac.uk

Stephen Cooper  
Computer Graphics  
Technology  
Purdue University  
West Lafayette, IN, USA  
coopers@acm.org

Michael Kölling  
School of Computing  
University of Kent  
Canterbury, Kent, UK  
+44 (0)1227 827570  
M.Kolling@kent.ac.uk

John Maloney  
Media Laboratory  
Massachusetts Institute of Technology  
Cambridge, MA, USA  
jmaloney@media.mit.edu

SUMMARY
This panel will showcase and compare three leading Initial Learning Environments (ILE): Alice, Greenfoot and Scratch.

Categories and Subject Descriptors
D.1.7 [Programming Techniques]: Visual Programming  
D.2.6 [Programming Environments]: Interactive Environments  
K.3.2 [Computer and Information Science Education]

General Terms: Design, Languages

Keywords
ILE, Alice, Greenfoot, Scratch.

1. INTRODUCTION
Although designed at different times and in different contexts, these three environments—Alice, Greenfoot and Scratch—can be classified together as sharing similar characteristics. All are visual, all aim to foster immediate engagement in an attractive activity, and all aim to introduce pre-University students to programming. We describe these as “Initial Learning Environments”.

As well as being used by individuals and in face-to-face group situations, these environments can also support remote community in the form of “galleries”, webpages where students can post and share their materials. Characterized as the fridge door approach (“at the end of a day at kindergarten, youngsters like to have a painting to take home to show the family and to have it displayed on the door of the refrigerator” [1]), the Scratch team describe this as “deep shareability” [2].

In this session, we will give a short demonstration of each ILE. We will then pose a series of the same questions to each ILE, to expose the particular strengths of each.

The purposes of this panel are threefold:

• Firstly, we seek to showcase ILE software, demonstrating the maturity of this type of environment.

• Secondly, by including the three major ILEs, we show the breadth of approach in the area.

• Thirdly, by interrogating each ILE in turn, we allow comparison of their features and encourage educators to select the resource most suited to their needs and their situation.

Each ILE is represented by a panelist who is central to the design team for their environment and so may authoritatively speak to features, goals, desires and wish-lists. They can not only demonstrate features of the environment, but also can speak about why they were put there. Equally, they are knowledgeable as to the directions that each environment might develop in the future.

2. ALICE
http://www.alice.org

Alice is a 3-dimensional interactive animation program visualization environment. Novice programmers build animated 3-D movies and author games as they learn introductory object-oriented programming concepts.

Alice provides a drag-and-drop development environment, to prevent students from making syntax errors. The software (and the source code, for the more adventurous) is freely available from the Alice website, and instructional/curricular materials are freely available to teachers at www.aliceprogramming.net.

Alice was originally designed as a tool to improve at-risk computing undergraduate students' ability to succeed in CS1 and beyond (see [3] for greater detail). It is currently being used in hundreds of colleges and secondary schools, in pre-CS1, CS1, pre-AP, and non-majors courses.

The Alice development team is currently building the next version of Alice, Alice 3. Alice 3 will enable the typing of Java code to build Alice worlds (as well as an ability to move between Alice's IDE and a more traditional one, such as NetBeans), as well as including characters from Electronic Arts' popular game, The Sims.

3. GREENFOOT
http://www.greenfoot.org

Greenfoot is an educational development environment highly specialized for the development of interactive, graphical applications. It is based on the Java programming language. Using Greenfoot, students from high school age can develop engaging and interesting programs, such as games and simulations, quickly and easily while learning fundamental programming concepts.

The use of text based programming based on Java makes very sophisticated applications possible, matching students' high
expectations, and prepares them for progression to more general programming environments.

4. **SCRATCH**

http://scratch.mit.edu/

Scratch is a media-rich, networked environment originally designed for use in “computer clubhouses”, a network of afterschool learning centers for youth from economically-disadvantaged communities. Scratch’s focus on media springs from this initial situation, as clubhouse participants were observed to be “creating and manipulating graphics, animations, videos, and music (and often integrating multiple media)”. Thus “Scratch emphasizes media manipulation and supports programming activities that resonate with the interests of youth, such as creating animated stories, games, and interactive presentations”.

A Scratch project is built from a background and a number of movable sprites. Programming is based on a metaphor of building blocks, and, influenced by previous systems such as LegoBlocks, is accomplished by dragging blocks from a palette and assembling them, like puzzle pieces, to create “stacks” of blocks. These “stacks” then determine various behaviors of the objects.

5. **REFERENCES**

1. Fincher, S. What are we doing when we teach programming? In Frontiers in Education ’99, pages 12a41-5. IEEE, November 1999

