Sneak Peek: Alice 3.0

A new way to teach programming

Concepts

Approaching programming with tools designed for use by professional engineers presents a steep learning curve: mainstream languages present a complex syntax designed to maximize productivity rather than for clarity. On the other end, didactic languages are often very limited in scope. Both are unattractive for students who are looking for practical applications and quick reward.

Alice is an innovative authoring system in which you can create stories by animating characters in a 3D world. Targeted at middle and high-school students, Alice offers a powerful interaction model based on objects and verbs.

We have tested the yet-to-be-released preview version of Alice 3.0, an ongoing ambitious rewrite of the already well established version 2.0.

The Art of Computer Programming

Computer science is often presented to students as an abstract discipline, very close to mathematics. While this may be scientifically more rigorous, idealized computers that only exist on paper tend to look aseptic and ultimately boring to the average learner.

The *practice* of programming is quite a different discipline from its underlying theoretical scaffolding. Software engineering studies the technical, organizational and social challenges of writing large-scale programs.

There is yet another way to look at it: some say that creating programs is *primarily a work of art*. As controversial as this statement may sound, the all-time best selling book about computer science is titled "*The Art Of Computer Programming*".

Be it a science, a branch of engineering or a modern expressive art, in the Internet era there seems to be some a practical value in learning the basics of how computers "think".

Visual programming

Alice is an Open Source project of Carnegie Mellon University written in Java, a modern programming language which is popular both in education and enterprise. This educational tool offers an innovative "low floor, high ceiling" approach that fits a wide spectrum of computer science topics.

The basic idea is to engage students in storytelling within a three-dimensional world that can be filled with objects and characters of all kinds. The students become the directors of a movie, by controlling the camera and making characters perform actions and interact with each other. There's a vast library of ready-to-use objects, creatures and human characters suitable for modern, historic

or fantasy worlds. It is also possible to create new objects by customizing one of these templates, or starting anew from a very sophisticated object editor.

Any object in the world can be controlled by means of command verbs, such as *move* or *turn*. Verbs are often used in conjunction with a certain number of parameters, such as “move” “forward” “10 feet”, for which reasonable defaults are provided. Any object belongs to a category, such as Dog or Human which may know how to perform additional actions, such as *bark* or *handshake*. Students can teach their characters to perform new actions by assembling simpler actions. For example, the action “go to X” could be assembled by “turn to X”, “walk forward <distance from X>”. Specific subcategories may change the meaning of common verbs such as *walk*: bipeds and insects walk in very different ways, but once the work of teaching a new verb is done, the director can just tell any object to walk 10 feet forward without being concerned by such details.

Underlying programming paradigm

This method of interaction has a very close relationship with the underlying Object-Oriented Programming paradigm used by Java. In OOP jargon, objects are called *instances,* object categories are *classes*, and the verbs used to perform actions are *methods* or *member functions*. Needless to say, students of CS require time to grasp these abstractions. Mapping them to real-world concepts helps clear the confusion. This is a transposition, not oversimplified analogy which detracts from correctness. All the other advanced aspects of modern Object Oriented Programming are also present: the ability to define new verbs in categories is called *specialization*, redefining existing verbs unleashes the concepts of *type-based polymorphism*, a technique used for *information-hiding*. Applying all this to knights and dragons helps students contextualize these abstractions and learn how to correctly apply them to solve problems.

Building the scene

On a higher scale, the process of creating a story involves placing command verbs for the various actors and for the camera in a meaningful sequence.

You can even place multiple actions in a “do together” block to perform them in parallel. This technique offers a wonderful opportunity to gently introduce students to some of the hardest topics of computer science: *concurrent execution, synchronization* and *inter-process communication.*

“Competition”

The idea of making young programmers control a character has its roots in the ancient educational programming language LOGO created in 1967. With just one fixed actor to play with, a turtle capable of drawing lines on its trail, LOGO would let students create geometrical figures on a 2-dimensional world.

Modern reinterpretations of the original LOGO concept include Turtle Art and Scratch, both of which offer a simple visual language based on snap-together blocks, although the interaction happens in a simpler, 2-dimensional world. Scratch also offers multiple actors with custom skins.

Reception

The website claims that Alice is being used in 10 to 15% of high education institutions, as well as a number of middle and high-schools, but fails to specify the details of how these suspiciously high figures were obtained.

Nevertheless, a strong online community seems to be inhabiting the forums, offering advice and publishing materials of all sorts. A much missed “web gallery” application would make these jewels easier to discover.

Several printed books are available, including *Learning to Program with Alice*, which is accompanied by abundant online teaching materials and exercises, whose redistribution is unfortunately forbidden and restricted by an instructor password.

Giving the software out for free and selling the book was probably part of Alice's business model for version 2.0. That is, before Sun came into play last year offering direct financial support for the development of version 3.0. Sun, evidently has a direct interest in fostering the adoption of Java grow within educational institutions, which by itself should guarantee Alice's future growth and prosperity.

Pros and Cons

With such a huge increase in complexity with respect to traditional “third person” programming environments, one would expect Alice to go far beyond mere teaching of computer science. Sadly, it falls short of being a tool that anyone but a aspiring programmer would want to use. Artists, choreographers and film directors would find the work-flow excessively “object oriented” rather than “expression oriented”. Alice's impressive library contains large quantities of unimaginative, assembly-line constructed object sets. The Alice mascot rendered in modern-manga style was supposed to look smart, but results instead somewhat pathetic. A quick glance at the web site is enough to get the impression of a struggle to boost the “coolness” factor of a product that was clearly built from the ground up by engineers.

Another frustrating aspect is slowness of the rendering engine, which may in part be justified by the beta status of the version we have evaluated. On the other hand, the overall stability was very satisfactory, which is a manifestation of the high-quality engineering process typical of Java shops.

Conclusions

All considered, Alice is a sophisticated tool with a rich feature set. One rarely sees a product of this depth and vastness in the small pond of free educational tools.

If you view computation as a tool accessory to teaching other subjects, and especially for teaching about story telling, Alice is probably an elephant in the classroom. But if you're looking for an environment to help introduce middle and high-school students to programming, Alice is certainly worth adopting.

About the author

Bernie Innocenti

Infrastructure coordinator, Sugar Labs

bernie@codewiz.org