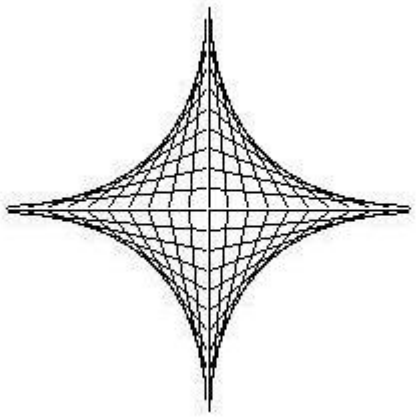
BATS:

A Geometric Figure Drawing Language

Behrooz Badii(Team Leader) Aleksandr Borovinskiy
Tanya Shtemberg Sui Sum Wong

September 23, 2003

Background:



That's an example of something we want to make with BATS. Using the JAVA graphics library through BATS, we can and will be able to create something like the picture above. BATS will be an enjoyable language, since its output is enjoyable, satisfying, and visually gratifying. This creates a language that people can use and that people want to use. What's better than seeing your code come out as a beautiful graphic image or geometrical figure?

In case of curiosity, the title of BATS came from taking the first letter of the first name of each person in the group (**B**ehrooz, **A**lex, **T**anya, **S**ui).

Introduction:

BATS is a geometric figure drawing language using JAVA code to draw simple to complex shapes and figures on a canvas. BATS is a reliable, understandable, efficient, usable, testable, portable, interpreted, high-performance, robust, internet compatible, and visually beautiful programming language.

Reliable:

"There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult."
(C.A.R. Hoare)

Since BATS's compiler was based off JAVA, the reliability of the language is relative to that of JAVA. Programmers can still make errors where their own drawings are not correct (for example creating a triangle instead of a rectangle), but due to the simplicity of the language, these errors would occur quite seldom. Also, if the JAVA

Virtual Machine does not have deprecated methods in the Graphics section of the JAVA library later on, BATS can stay reliable.

Understandable:

"Increasingly, people seem to misinterpret complexity as sophistication, which is baffling - the incomprehensible should cause suspicion rather than admiration."
(Niklaus Wirth)

Due to the grammar of the language, and the immediate graphic results, the code is understandable. The language's naming structure gives the programmer the freedom to make reasonable names, bringing an understanding to what each line of code, each variable, and each shape is and does.

Efficient:

"A language that doesn't have everything is actually easier to program in than some that do."
(Dennis M. Ritchie)

BATS gives several possibilities to create a shape or line to draw on the screen. It's also simple to use, so a simple-to-use language creates efficiency. Since the compiler of the language is object oriented (the compiler is code in JAVA), the compiler is efficient due to its simplicity to understand. So, a compiler that can be understood or debugged easily will create efficient code.

Usable:

"I consider it the obligation of scientists and intellectuals to ensure that their ideas are made accessible and thus useful to society instead of being mere playthings for specialists."
(Bjarne Stroustrup)

Programmers and users need a very limited background of understanding graphs and points to be able to use BATS. If programmers know how to make complex graphical images using algorithms, then there is no bound to what they can make. The usability of the BATS language has a direct relation with the knowledge of the user in creating graphical images and complex two-dimensional geometric figures.

Testable:

"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."
(Brian Kernighan)

BATS can be programmed and debugged for syntax errors. For semantic errors of incorrect drawing, the error can be immediately seen and corrected. If the programmer puts a circle where it shouldn't be, then they can change it very quickly after seeing the incorrect image he or she created.

Portable:

Most papers in computer science describe how their author learned what someone else already knew. (paraphrase)
(Peter Landin)

BATS is portable because it is based off JAVA. Since JAVA is portable, BATS is portable. Furthermore, the graphics portion of the JAVA library has been relatively unchanged throughout different releases and editions of JAVA. This makes the BATS compiler backwards compatible with releases of JAVA.

Interpreted:

"The best writing is rewriting."
(E. B. White)

BATS is interpreted into JAVA, and the interpreted program will be used to make a graphical image on an applet.

High-performance:

"C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do, it blows away your whole leg."
(Bjarne Stroustrup)

Once again, the language BATS is interpreted into JAVA, giving BATS a great asset: high performance. Since JAVA is a high-performance language, BATS becomes a high-performance language. JAVA's Just In Time compiler is high-performance, so that is a boon for BATS.

Robust:

"Any sufficiently advanced bug is indistinguishable from a feature."
(Rich Kulawiec)

Due to BATS's simplicity, it is robust. Since there are simple ways to display complex figures, BATS becomes robust. In terms of drawing, it can do many things. It

can draw anything that is two-dimensional. The more the programmer puts into his or her code, the more the power of BATS is realized.

Internet Compatible:

"Languages shouldn't hinder progress by outliving their usefulness."
(Alan Kay)

BATS's graphical images will be displayed on an applet. Since an applet is used to show BATS's output, it is therefore Internet compatible. Internet-compatible JAVA interprets the code, and its output is shown an applet, a feature of JAVA seen very often on websites.

Visually Beautiful:

"Do what you think is interesting, do something that you think is fun and worthwhile, because otherwise you won't do it well anyway."
(Brian Kernighan)

The output of BATS is what makes it stand out. Dealing with graphics, one would assume, and assume correctly, that BATS is visually beautiful. Beauty is in the hands of the programmer. If they can program a beautiful image, then it will come into existence.

Example of Code:

The most important and distinguishing bit of the language are the shape object, which can take an array of points that will be connected to each other, and the draw() function, which takes the shape object as a parameter and produces the drawing of that shape on the applet. The following is the coding needed to draw the above picture in the Background section:

```
Shape Xshape
Shape Yshape
Shape lineshape
Point[] temp
Point[] x
Point[] y
int w
int z
w = 100
z = 10
x = {[-100,0],[100,0]}
y = {[0,-100],[0,100]}          //making x and y axes
Xshape.sarray = x
```

```
Yshape.sarray = y
draw(Xshape)
draw(Yshape)                    //drawing x and y axes
while (w > 0) {
        temp = {[z,0],[0,w]}
        lineshape.sarray = temp
        draw(temp)              //draw top right quadrant
        temp = {[z,0],[0,-w]}
        lineshape.sarray = temp
        draw(temp)              //draw bottom right quadrant
        temp = {[-z,0],[0,w]}
        lineshape.sarray = temp
        draw(temp)              //draw top left quadrant
        temp = {[-z,0],[0,-w]}
        lineshape.sarray = temp
        draw(temp)              //draw bottom left quadrant
        w = w -10
        z = z + 10
}
```