Abstract

This paper presents results of a study to compare Java and C# programming languages features in terms of portability, functional programming and execution time. This comparison permits to evaluate both programming languages to know which one has better performance in the image processing area.

1. Introduction

Image Processing involves the manipulation of the pixels of one image to form another image. Those techniques have been used for years to improve the visual appearance of images. Image processing operations can be divided into three major categories: Image Compression, Image Enhancement and Restoration, and Image Measurement Extraction. An image can be obtained from some devices such as: scanners, sensors, etc.

Many applications to solve the image processing discipline have been developed for years. Programmers are using different kinds of programming languages to create new tools that improve quality of images to facilitate its study. Programmers use high level language such as C, Java and MATLAB to create these environments. Some of these high level languages, such as MATLAB, offer a rich functionality with a large group of images routines, but hides many important aspects of the algorithms [1]. For that reason, is important to create new environment to know the performance of languages and compare them.

C# is one of the most recent programming languages created by Microsoft and Java is a programming language very popular to create environment to solve the image processing discipline. This paper deals with a basic comparison between Java and C# programming languages focus in image processing programming area. The goal of this paper is find out is discover which language facilitates manage of image processing algorithms.

In next section a comparison of both languages will be done in terms of: portability, functional programming and execution time. The last section will provide the conclusion and future work.

2. Java and C# Portability

Portability refers to the flexibility to run a program in different architectures and operating systems. Java offers three kinds of portability: source code portability, CPU architecture portability and Operating System. A Java code can be run at any machine independently of
the operating system and its CPU architecture. Java uses a Java Virtual Machine (JVM) that is an abstract computing machine. It has an instruction set and manipulates various memory areas at run time. That means that a user can do a Java program in Windows platform and the same program can be run in a computer using Linux platform without doing any change to that code. Figure 1 shows some of the platforms supported by JVM.

![Figure 1. Some platforms supported by JVM.](image)

On the other hand, a program written in C# can be executable just on Windows operating system. C# is a language build specifically to program the .NET Framework. The .NET Framework consist of a runtime environment called (CLR) Common Language Runtime, and a set of base class libraries, which provide a development platform that can be exploited by a variety of languages and tools. The CLR greatly improves runtime interactivity between programs, security, development simplicity, and cross-language integration. Whereas a Java virtual machine only runs Java programs, the CLR has been designed to run programs written in any language. Microsoft has developed .NET compilers for C++, C#, JScript, and a new version of Visual Basic named VB.NET. At the same time, other companies are developing .NET-compatible compilers for many other languages. Figure 2 shows the CLR portability.

![Figure 2. CLR portability.](image)

3. Functional Programming in Java and C#

Functional programming consists in that the main program is written as a function which receives the program’s input as its argument and delivers the program’s output as its result. Typically the main function is defined in terms of other functions, which in turn are defined in terms of still more functions, until at the bottom level the functions are language primitives[2].

In this case, the function to be developed will be functions that will operate on images. Those functions will receive an image as a parameter, then it will perform changes to the image and it will return another image as a result. Figure 3 shows the functional programming diagram to manipulate images.

![Figure 3. Functional Programming for Image Processing.](image)
4. Executable Speed in Java and C#

To execute a source code made in Java, the machine must have a JVM. JVM interprets the class file format. A class file contains Java virtual machine instructions and a symbol table, as well as other ancillary information [3]. After that, JVM executes that program using the requirements to determine the operating system. Every time a Java code is executed, the JVM has to interpret the class files. On the other side, when the user runs a source code in C#, the computer must have a CLR environment for hosting managed applications. If the user runs a managed executable, the CLR loads the module containing the executable and execute the code inside it [4].

An important factor to study in image processing programming is the execution speed of Java and C# programs respectively. When a programming language is going to manipulate any images, the images are converted into a matrix where every element of the matrix represents a pixel of the image.

A benchmark has been implemented in both languages in order to find out which language has better performance. The benchmark multiplies a vector by a squared matrix. The tests were realized in a Dell Laptop Inspiron 4150, Pentium 4 of 1.8 GHz, RAM 384MB, 30GB Ultra ATA Hard Drive. The operating system used was Windows XP Professional. The Java version is Java™ 2 Runtime Environment, Standard Edition (build 1.4.0-b92). The .NET Edition used was Microsoft Development Environment 2002 version 7.0. Table 1 shows resulting executing time in both environments.

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>Matrix Size</th>
<th>C# Time(ms)</th>
<th>Java Time(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>512 x 512</td>
<td>31</td>
<td>10</td>
</tr>
<tr>
<td>1024</td>
<td>1024 x 1024</td>
<td>150</td>
<td>30</td>
</tr>
<tr>
<td>2048</td>
<td>2048 x 2048</td>
<td>610</td>
<td>590</td>
</tr>
<tr>
<td>4096</td>
<td>4096 x 4096</td>
<td>2544</td>
<td>4357</td>
</tr>
<tr>
<td>8192</td>
<td>8192 x 8192</td>
<td>9975</td>
<td>18697</td>
</tr>
<tr>
<td>16834</td>
<td>16834 x 16834</td>
<td>2947168</td>
<td>N/A</td>
</tr>
</tbody>
</table>

The benchmark measures the execution time that takes the multiplication of the vector and matrix in both languages. Java provides a function to take time in milliseconds. In contrast, C# has not a function to take time in milliseconds. For that reason, the time was taken in hours, minutes, seconds and milliseconds to then convert all to milliseconds. Figure 4 shows Java and C# code to take time system of the multiplication of the vector and matrix.

**Java Code**

```java
Start = System.currentTimeMillis();
//matrix and vector multiply
for (int i=0; i<size; i++)
  for (int j=0; j<size; j++)
    re[i]=a1[j]*matrix[j][i]+re[i];
end = System.currentTimeMillis();
```

**C# Code**

```csharp
hrbf   = DateTime.Now.Hour;
mnbf = DateTime.Now.Minute;
scbf   = DateTime.Now.Second;
msbf  = DateTime.Now.Millisecond;
//matrix and vector multiply
for (int i=0; i<size; i++)
  for (int j=0; j<size; j++)
    re[i]=a1[j]*matrix[j,i]+re[i];
msaf = DateTime.Now.Millisecond;
```

**Figure 4.** Java and C# code for vector and matrix multiply.
The result shows that Java application executes in less time than C# when matrix and vector is less than 1024 size. For matrices and vectors of 2048 size, the execution time is similar for both languages. When matrix and vector sizes are more than 4096, C# execute in less time than Java. In the environment that was used for the test, Java was unavailable to execute a multiplication matrix vector with a 16834 size. Therefore, C# language can perform the matrix and vector of 16384, it takes a while. Figure 5 shows that the execution time grows in exponential form when matrix and vector grows.

![Execution Time in C# and Java languages](image)

**Figure 5.** Execution Time in Java and C#.

A hypothetical response to explain this is that C# manages better the system components or that it uses the hard disk to solve this. Others studies will be done to know clearly what can be the causes.

**6. Conclusion and Future Work**

In terms of portability, Java is portable in source code, hardware and operating system. However, C# is portable in terms of programming languages. Both languages are Object Oriented language that allows reuse the code and offers the advantages of a functional programming that permit convert main program in terms of functions that receive an argument as input and produce another argument as an output. Java and C# are similar in the construction of classes. When a Java program is compiled, it generates a class file that contains a set of statements that JVM must interpret to execute it at any platform. C# just creates an executable file from the source code.

In the experiment to measure the execution time in Java and C#, Java was faster than C# when the vector and matrix sizes were small. For the other side, C# is faster than Java when matrix and vector were greater than 4096. Another factor to observe in this experiment is that Java cannot solve vector and matrix with size of 16834 and C# does. As a hypothetical conclusion to explain is that C# manages better the system components or use hard disk to solve it. Others studies will be done to know clearly what can be the causes.

Future work includes develop a C# environment with image processing algorithms in order to compare it with other Java applications.

**References**


