CS22510Assignment 2

C++, C and Java Programming Paradigms

Author	Owen Garland(owg1)
	130072557
Date Published	April 16, 2015
Assessed By	Fred Labrosse, Neal Snooke
Department	Computer Science
Address	Aberystwyth University
	Penglais Campas
	Ceredigion
	SY23 3DB

1 Introduction

The ladybirds and aphids simulation assignment was written in C++, this was the first time I had used C++, so it meant that I had to spend a lot of time learning the language as well as trying to create a good solution to the problem. Once the project was completed it became clear that I had tried to use more of the C style of coding than the object orientated approach of Java and C++. The issue with this is that it makes the project less maintainable and harder to extend and add new features; however I did find it simpler to develop as I had to worry less about using inheritance.

My application did meet all of the requirements, but due to not treating each creature in the simulation as a unique object, but rather a representation that can be deleted and created elsewhere (loosing its properties), it would be hard to add in additional features such as a life span for each unique creature.

The reason I did it this way was that it was simpler to implement in the short run as I didn't have to worry about passing objects around, or keeping ordered lists of the creatures in each cell. Instead I could just destroy them and create a new one in its place. This is most evident with the mating requirement, instead of having each unique object find another unique partner to mate with, I could just divide the number of creatures in the cell by two and then create that many offspring based on the probabilities of them reproducing.

2 C

C was developed in 1972 by Dennis Ritchie, and was used to create the Unix operating system. Since then it has continued to be used to this day for a variety of purposes, generally anywhere that software needs to interact directly with hardware, such as operating systems, hardware drivers and embedded devices. The Linux Kernel for example is written almost entirely in C.[1] The reason for this is that C allows a lot better access to low level memory than other languages. The programmer can directly manipulate the memory of the machine by using pointers to access variables memory locations; due to this it is seen as a more difficult language to learn and use as the programmer needs to manually manage their memory without any kind of garbage collection.

2.1 Suitability for this project

C is historic, but interwoven with all Unix-like operating systems, as well as being a prime choice of language for systems that require a precise amount of performance; whether that be due to limited resources on an embedded system, or in an environment where performance is crucial.

For this project though, neither of those factors apply as this software is not using any Unix features, nor does it need to be efficient or fast performing since it will be ran on desktop machines with more than ample resources. The project also states that the simulation is inherently object orientated[2] which makes using a non object orientated language clearly a bad choice.

Despite this it would still be possible to create the application in this way, however C++ offers the power of C with a large amount of extra features and supports full object orientation. This means that the choice between C and C++ is clear, C++ is more suitable for this project by far. Because of this there doesn't seem to be much need to compare C and Java, as C++ shares the majority of the pros and cons that Java does so it makes more sense to compare the two object orientated languages directly as they are naturally more suited to this task than C.

If this project needed to be made to work on a large Unix system and support parallel processing for performance reasons, then it may make a lot more sense to write this in C as it could be made a lot more efficient, and utilising the resources and features that such a system may have to offer could be a lot easier. However this again is probably true of C++.

3 C++

C++ was developed in 1983 by Bjarne Stroustrup, its core concept is to bring object orientation to the C language. Object Orientation is a very powerful paradigm that focuses on using objects to abstract and encapsulate code; making it easier to break down a problem into constituent parts to work on. I found learning C++ to have quite a daunting learning curve despite being familiar with C and Java. In my personal experience this is mostly due to the online resources that were available being sub standard, it was quite hard to Google for solutions when compared to other languages. I think this just reflects the difficulty of the language, the people that use it regularly don't tend to need StackOverflow to find solutions to their problems.

That aside C++ is a very powerful language, the efficiency and speed of C as well as the power of object orientation makes it the go to language for large applications where speed matters, such a browsers and video games to name a few.

3.1 Suitability for this project

As mentioned earlier C++ is a better candidate for this project than C, simply because object orientation is a core feature of the language, and is a key requirement for this project. Some of the other advantages over C are having a String class, that makes life a lot easier, especially when it comes to formatting output, which was very important for this assignment as we were creating an animation with text. Another very useful feature is Exceptions, which make error handling a lot easier when compared to C.

One issue I encountered with C++ was the tendencies to program like I was using C, this was frustrating in small ways, such as forgetting to put the class name before a method definition and declaring global variables in both the source file and the header file. The main problem though, was that I did not make the most of the objects that I was creating, instead of having each creature be a unique object with its own properties I was simply using the object in a way similar to a struct.

The application that is being built doesn't benefit from the performance gains of C++, but development does get slowed down by having to worry about garbage collection, which is something that the programmer doesn't have to worry about with Java. C++ gives the programmer more power, but also more things to worry about, since this application doesn't require the features provided by C++, Java does look like a simpler option.

Pro's

- Object Orientation + efficiency
- Operator overload
- Strings
- Native binaries
- Multiple Inheritance
- Exceptions

Con's

- Complex OO
- Online documentation / support is hard to find
- Hard to read / write
- Compiler dependant behaviour
- Debugging segfaults

4 Java

Java was developed in 1995 by James Gosling for Sun Microsystems. It was designed to be almost entirely object orientated, and most interestingly to be compiled once and ran anywhere. This was achieved by having the code compile down to Java byte code which could then be ran on any Java Virtual Machine (JVM) on any platform. While this does work, it does make it a requirement that systems have Java installed first.

I personally don't enjoy developing in Java that much, for simple applications that don't require multiple classes, everything having to be an object in Java becomes quite tiresome. Creating and compiling a distributable jar is also an additional step that you don't have to worry about with C/C++, this isn't a massive disadvantage but it does slow development time down slightly, especially for people who choose not to use an IDE and prefer to use a text editor like Vim.

4.1 Suitability for this project

Java is a good choice of language for this project, there is a clear need for object orientation as mentioned earlier, and Java covers that aspect very well since object orientation is so built into the language. One of the clear differences between the languages from a programmers perspective is the inclusion of garbage collection. Not having to worry about memory management makes it a lot easier to develop applications as you can allow the JVM to do some of the hard parts for you, at the cost of less efficient code.

This is a definite advantage over C++ with regards to this application. Whilst developing my C++ application I did have to worry about memory leaks and spend quite a long time debugging them with valgrind to make sure that there was no memory leaks. If the project had been done in Java I wouldn't have had to worry about this issue, which would have sped up development greatly.

From a technical standpoint this is where Java comes ahead of C++ in terms of viability for this project. The additional power (and complexity) of C++ simply aren't needed for a simple project such as this. If the project was specifically for a single platform, that would nullify Java's advantage of cross platforms binaries; if the project was also centered around performance (Running many thousands of simulations as fast as possible) then it may make sense to do the project in C++ as there would be some speed gains from using native binaries instead of ones written for the Java Virtual Machine. Sadly though that isn't the case so Java wins out here.

One clear downside of Java is that the end users have to have Java installed on their machine. Although most machines do have this software installed not all do and this can provide an issue for inexperienced users. There is also a small moral issue of Java being owned by Oracle and being proprietary and not open source. However this is easily avoided by using the OpenJDK alternative which is open source and freely available.

Pro's

- Platform independent binary
- Easy Object Orientation
- Android support
- Fixed definitions
- Garbage collection
- Exceptions

Con's

- Everything is OO
- Slower than C/C++
- Java is required to be installed
- Security
- Eclipse

5 Language choice for this project

To summarise C is best harnessed in applications that require efficiency, either due to limited resources or for performance reasons. C++ has the benefits of C but with the overhead of having to manage objects, which slightly reduces performance, however it allows for object orientated programming which is a very strong plus when it comes to dealing with large systems. Java doesn't have the speed advantages, and access to low level memory that C/C++ does, however object orientation is at its core of the design philosophy so developing OO applications is very natural in Java.

The task in hand matches the use case of Java the closest, an object orientated approach is crucial, but performance and efficiency aren't required. This makes a C or C++ approach to the problem more complex than what is needed, as you have to deal with memory management and no garbage collections.

From the comparisons I have made, it appears that the most sensible choice of language for this application would be Java. This is mainly as C doesn't support true object orientation, and while C++ does support it, the application has no need for the more complex features of C++, nor is performance an issue. Using Java would also have the benefit of making the binary available to be ran on all platforms, this would save time porting the application to other platforms if that was required.

References

[1] 95% of the Linux Kernel is written in C `https://github.com/torvalds/linux`

[2] "The simulation is inherently object oriented and almost everything can be represented as a class" Assignment 1 brief, Fred Labrosse, Neal Snooke, 17/02/2015.