Event Based Programming

S. Hansen
UW - Parkside

T.V. Fossum
SUNY - Potsdam

Kenosha WI, May 23, 2010

Contents

Preface

Introduction

This is a book about event based programming. There are dozens of computer science books that have 'event' or 'event based' in their titles. Almost all of these are about some particular language or system that uses events. This book is different. Our goal is to introduce you to event based programming – and, more generally, event based systems – as a computer science paradigm that focuses on the fundamental ideas relating to understanding, designing, implementing, and testing loosely coupled systems.

One of our objectives is to introduce you to approaches used in event based programming and to illustrate them with carefully crafted examples. Programming event based systems is different from procedural programming or object-oriented programming. If the solution is to be event based, programmers think about the problem differently. Its important that you understand why an event based approach is more appropriate for some problems than for others.

Event based programming also has its own set of challenges, principally because almost all event based systems are nondeterministic and are inherently difficult to test and debug. Another of our objectives is to show you how to use conceptual and design tools to create event based systems that behave correctly.

What you will learn from this book is applicable to any event based language or library. Since this is a book about programming, you will encounter many programs as you read it. We have chosen Java as our principal implementation language. Java is available almost universally for hardware platforms and operating systems, and Java has native support for events. However, this book is not about making you an expert in Java. Instead, you will gain an understanding of how to develop event based software, with Java serving as a particular implementation language. In the end, you will be able to develop reasonably sized event based applications in Java, and you will be able to take the principles and ideas you learn and apply them to new and different event based languages as needed.

Computing is continually changing. Computing professionals are always playing catch-up, trying to keep their technical knowledge current with regard to new languages, operating systems, and application versions. On the other hand, programming paradigms change very little. Procedural programming – with its while loops, if statements, and procedure calls – has remained largely unchanged since the 1950s. Similarly, the concepts behind object-oriented programming have been relatively stable for over 20 years. Procedural programming and object-oriented programming are *paradigms*. They aren't about any particular language. Instead, they give the programmer problem solving techniques and methodologies that are applicable in a variety of languages.

In recent years, event based programming has emerged as its own distinct paradigm. Its notions of runtime association of sources and handlers, and minimal timing assumptions apply, and will continue to apply, no matter how many new event based languages are released. The fundamental ideas behind the paradigm are stable. By emphasizing an understanding of the paradigm, we will lay a foundation that will let you easily pick up new event based languages as they emerge.

Events are incredibly important in modern computing. They occur in graphical user interfaces (GUIs), operating systems, discrete event simulation, database management systems and many other computing-related fields. They are also integral to the operation of user interfaces in modern electronic devices such as cellular phones and television sets. Because events are so ubiquitous, it is reasonable to expect that you, as a student of computing, should understand them in detail. Unfortunately, you are more likely to learn first about event based programming when you are asked to develop GUIs – and then you would typically be exposed only to what you need to make the GUI operate. As you will see, events have many more applications than implementing GUIs.

Event based systems are commonplace, but they have distinct properties and pose unique challenges for developers. Event based programming deserves a broad–based comprehensive treatment in the computer science curriculum. This book is an effort to provide that treatment.

Audience

You should be comfortable approaching the material in this book if you have a background equivalent to a two-semester undergraduate-level course in programming with some experience using an object-oriented language. Knowing Java is a plus, but you should be able to understand the programming examples without it.

You should be comfortable dealing with elementary data structures such as arrays and lists. We may refer occasionally to more advanced data structures, in which case we will provide background discussion and pointers to resources where you can study relevant material. You should also be acquainted with classes, objects, inheritance, and polymorphism, though again we may take the opportunity to refresh your knowledge of these ideas when they play a particularly important role in our discussions or in example code. We do not expect you to have studied mathematics beyond elementary calculus and/or discrete math.

Resources

While Java is our primary implementation language in this book, other languages (C++, Python, and C#, for example) – along with appropriate library support – can serve equally well. We will post sample code on our course website (`http://cs.uwp.edu/Events`) to supplement the code in this book as it becomes available.