Event Driven Languages

Purpose: interactive systems

Examples: Visual Basic, JavaScript, Java Swing, Tcl/Tk, Qt, GTK, . . .

NB: most of these are event-driven libraries used by existing languages

Notable features: based on the idea of having code executed as a consequence of something (an event) happening, rather than in some pre-specified order

Event Driven Languages

Feet

- Visual Basic: You do a Google search on how to shoot yourself in the foot. You find seventeen completely different ways to do it, none of which are properly structured. You paste the first example into the IDE and compile. It brushes your teeth

Event Driven Languages

Perhaps more of a style of programming, rather than a family of languages

Event Driven Languages

Perhaps more of a style of programming, rather than a family of languages

Lots of general purpose languages can be used in the event driven style, though there are a few languages specifically designed for this, e.g., Simula

Event Driven Languages

Perhaps more of a style of programming, rather than a family of languages

Lots of general purpose languages can be used in the event driven style, though there are a few languages specifically designed for this, e.g., Simula

Widely used to support GUIs and other interfaces and control systems, e.g., embedded controllers

Event Driven Languages

Typically the code contains a *main loop* that waits for events (key presses, mouse clicks, temperature limits reached, data packets arriving, clock timeouts, etc.) which then chooses which chunk of code (*event handler*) to run in response

Event Driven Languages

Typically the code contains a *main loop* that waits for events (key presses, mouse clicks, temperature limits reached, data packets arriving, clock timeouts, etc.) which then chooses which chunk of code (*event handler*) to run in response

```
while (FAMNextEvent(&fc, &fe)) {
    if (fe.code == FAMExists || fe.code == FAMEndExist)
      continue;
    t = time(NULL);
    tm = localtime(&t);
    strftime(buf, 32, "%H:%M:%S", tm);
    printf("%s %s: %s\n", buf, trim(fe.filename),
                          event[fe.code]);
}
```

Event Driven Languages

The style is very much like interrupt processing

Event Driven Languages

The style is very much like interrupt processing

Code has to be written with the understanding that you don't know in what order the parts of the code will be executed

Event Driven Languages

The style is very much like interrupt processing

Code has to be written with the understanding that you don’t know in what order the parts of the code will be executed

Widely used in interactive applications

Event Driven Languages

The style is very much like interrupt processing

Code has to be written with the understanding that you don’t know in what order the parts of the code will be executed

Widely used in interactive applications

Also very important in *simulation*

Simulation

This is where you simulate some situation, e.g., molecules in a gas, tanks on a battlefield, to discover its properties

Simulation

This is where you simulate some situation, e.g., molecules in a gas, tanks on a battlefield, to discover its properties

Objects interact by events, e.g., a molecule has hit another, a tank has fired a missile

Simulation

This is where you simulate some situation, e.g., molecules in a gas, tanks on a battlefield, to discover its properties

Objects interact by events, e.g., a molecule has hit another, a tank has fired a missile

These events trigger some behaviour, e.g., molecules change direction of travel, tanks explode

Simulation

This is where you simulate some situation, e.g., molecules in a gas, tanks on a battlefield, to discover its properties

Objects interact by events, e.g., a molecule has hit another, a tank has fired a missile

These events trigger some behaviour, e.g., molecules change direction of travel, tanks explode

Widely used in a huge variety of situations

Simulation

This is where you simulate some situation, e.g., molecules in a gas, tanks on a battlefield, to discover its properties

Objects interact by events, e.g., a molecule has hit another, a tank has fired a missile

These events trigger some behaviour, e.g., molecules change direction of travel, tanks explode

Widely used in a huge variety of situations

So there are lots of simulation specific languages, e.g., Simula, SPICE, and so on

Markup Languages

Purpose: description of objects

Examples: HTML, XML, SGML, CSS, nroff, LaTeX, . . .

Notable features: use of notation, usually within a document, to describe elements of the document (often, but not exclusively, visual); generally not “executed” in the usual sense

Markup Languages

- HTML: HyperText Markup Language
- XML: Extensible Markup Language
- SGML: Standard Generalized Markup Language
- CSS: Cascading Style Sheets
- nroff: new roff (roff: runoff)
- LaTeX: Lamport's TeX (TeX: from "technology")

Markup Languages

Feet

- HTML: You cut a bullethole in your foot with nothing more than a small penknife, but you realize that to make it look convincing, you need to be using Dreamweaver

Markup Languages

Feet

- HTML: You cut a bullethole in your foot with nothing more than a small penknife, but you realize that to make it look convincing, you need to be using Dreamweaver
- XML: You can’t actually shoot yourself in the foot; all you can do is describe the gun in painful detail

Markup Languages

Feet

- HTML: You cut a bullethole in your foot with nothing more than a small penknife, but you realize that to make it look convincing, you need to be using Dreamweaver
- XML: You can't actually shoot yourself in the foot; all you can do is describe the gun in painful detail
- nroff:

```
troff -ms -Hdrwp | lpr -Pwp2 & .*place
bullet in footer .B .NR FT +3i .in 4 .bu Shoot!
.br .sp .in -4 .br .bp NR HD -2i .*
```

Markup Languages

Feet

- HTML: You cut a bullethole in your foot with nothing more than a small penknife, but you realize that to make it look convincing, you need to be using Dreamweaver
- XML: You can't actually shoot yourself in the foot; all you can do is describe the gun in painful detail
- nroff:

  ```
  troff -ms -Hdrwp | lpr -Pwp2 & .*place
  bullet in footer .B .NR FT +3i .in 4 .bu Shoot!
  .br .sp .in -4 .br .bp NR HD -2i .*
  ```
- CSS: Everyone can now shoot themselves in the foot, but all their feet come out looking identical and attached to their ears

Markup Languages

Very widely used

Markup Languages

Very widely used

- HTML was originally used to describe the content and appearance of Web pages. Usually poorly

Markup Languages

Very widely used

- HTML was originally used to describe the content and appearance of Web pages. Usually poorly
- These days the accepted approach is to use HTML to describe the content, and use CSS to describe the appearance

Markup Languages

Very widely used

- HTML was originally used to describe the content and appearance of Web pages. Usually poorly
- These days the accepted approach is to use HTML to describe the content, and use CSS to describe the appearance
- XML is used to markup the *meaning* of text. Currently seen as the cure to all “Web 2.0” scenarios. Usually incorrectly

Markup Languages

HTML

```
<html>
<head>
<title>CM20214/221: Programming II</title>
<link rel="stylesheet" type="text/css" href="notes.css">
</head>
<body>
<h2>CM20214/221: Programming II</h2>

<h4>Some texts</h4>
Books on Functional Languages
<p>
Lisp has been about since 1957,
```

Markup Languages

CSS

```
body {
  font-family: Arial;
  background: white url("bg.png") repeat-y;
}

tt {
  font-size: larger;
}

.warn {
  color: red;
}
```

Markup Languages

XML

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle=
     "http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
      <m:OrderItemResponse xmlns:m="Some-URI">
      <OrderNumber>561381</OrderNumber>
      </m:OrderItemResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP is a standard data encoding for transfer of data between Web services that uses XML

Markup Languages

HTML

HTML and XML are both derivatives of a more general language, SGML

Markup Languages

HTML

HTML and XML are both derivatives of a more general language, SGML

HTML is ubiquitous

Markup Languages

HTML

HTML and XML are both derivatives of a more general language, SGML

HTML is ubiquitous

Unfortunately, its design ignored a lot of useful earlier work on hypertexts

Markup Languages

HTML

HTML and XML are both derivatives of a more general language, SGML

HTML is ubiquitous

Unfortunately, its design ignored a lot of useful earlier work on hypertexts

HTML/CSS is about *display* of documents

Markup Languages

XML

XML is about identifying information within documents

Markup Languages

XML

XML is about identifying information within documents

XML is widely used in Web 2.0 and Web services as data communication protocols

Markup Languages

XML

XML is about identifying information within documents

XML is widely used in Web 2.0 and Web services as data communication protocols

Specifications exists for:

Markup Languages

XML

XML is about identifying information within documents

XML is widely used in Web 2.0 and Web services as data communication protocols

Specifications exists for:

- MathML: mathematics

Markup Languages

XML

XML is about identifying information within documents

XML is widely used in Web 2.0 and Web services as data communication protocols

Specifications exists for:

- MathML: mathematics
- OFX: Open Financial Exchange, financial data

Markup Languages

XML

XML is about identifying information within documents

XML is widely used in Web 2.0 and Web services as data communication protocols

Specifications exists for:

- MathML: mathematics
- OFX: Open Financial Exchange, financial data
- XUL: XML User-interface Language, a language for describing user interfaces

Markup Languages

XML

XML is about identifying information within documents

XML is widely used in Web 2.0 and Web services as data communication protocols

Specifications exists for:

- MathML: mathematics
- OFX: Open Financial Exchange, financial data
- XUL: XML User-interface Language, a language for describing user interfaces
- AML: Astronomical Markup Language, for controlling astronomical instruments.

Markup Languages

XML

- RSS: Really Simple Syndication
- WML: Wireless Markup Language
- SVG: Scalable Vector Graphics
- MusicXML: music notation
- VoiceXML: Voice Extensible Markup Language
- PDML: Product Data Markup Language
- ODF: Open Document Format
- SMIL: Synchronized Multimedia Integration Language
- Gastro Intestinal Markup Language
- And hundreds of others

Markup Languages

Originally designed to be text based and therefore easily debugged by sight, common usage of HTML and XML is so complicated this is no longer possible

Markup Languages

Originally designed to be text based and therefore easily debugged by sight, common usage of HTML and XML is so complicated this is no longer possible

XML is being adopted everywhere for Web applications, often without proper consideration of the alternatives, such as JSON

Markup Languages

Originally designed to be text based and therefore easily debugged by sight, common usage of HTML and XML is so complicated this is no longer possible

XML is being adopted everywhere for Web applications, often without proper consideration of the alternatives, such as JSON

Also, increasingly it is being used to *store* information, which it is *very* bad at

Markup Languages

Originally designed to be text based and therefore easily debugged by sight, common usage of HTML and XML is so complicated this is no longer possible

XML is being adopted everywhere for Web applications, often without proper consideration of the alternatives, such as JSON

Also, increasingly it is being used to *store* information, which it is *very* bad at

Use a database to store information!

Markup Languages

Originally designed to be text based and therefore easily debugged by sight, common usage of HTML and XML is so complicated this is no longer possible

XML is being adopted everywhere for Web applications, often without proper consideration of the alternatives, such as JSON

Also, increasingly it is being used to *store* information, which it is *very* bad at

Use a database to store information!

If you ever have a project that uses an “XML database”, walk away in disgust

Object Oriented Languages

Purpose: general programming

Examples: Java, C++, Objective C, Lisp, Perl, JavaScript, Scala, . . .

Notable features: use of objects as a means to control complexity

Object Oriented Languages

Feet

- C++: You accidentally create a dozen clones of yourself and shoot them all in the foot. Emergency medical assistance is impossible since you can’t tell which are bitwise copies and which are just pointing at others and saying, “That’s me, over there.”

Object Oriented Languages

Feet

- C++: You accidentally create a dozen clones of yourself and shoot them all in the foot. Emergency medical assistance is impossible since you can't tell which are bitwise copies and which are just pointing at others and saying, "That's me, over there."
- C++ (2): "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)

Object Oriented Languages

Feet

- C++: You accidentally create a dozen clones of yourself and shoot them all in the foot. Emergency medical assistance is impossible since you can’t tell which are bitwise copies and which are just pointing at others and saying, “That’s me, over there.”
- C++ (2): “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)
- “C++ is history repeated as tragedy. Java is history repeated as farce” (Scott McKay)

Object Oriented Languages

Feet

- Objective C: You write a protocol for shooting yourself in the foot so that all people can get shot in their feet

Other Classifications

We are going to look at OO in depth shortly, but there is more to be said about language families before we move on

Other Classifications

There are many other classifications that cut across the families we have described

Other Classifications

There are many other classifications that cut across the families we have described

Some more important than others

Other Classifications

There are many other classifications that cut across the families we have described

Some more important than others

- Declarative and Imperative
- Parallel, Distributed
- GC and non-GC
- Strongly typed, weakly typed, statically typed, dynamically typed and untyped
- Area of application: numeric, symbolic, business process, graphical, database, . . .
- Interpreted and Compiled (byte code interpreted etc.)
- and so on

Declarative and Imperative

Imperative: the program is a list of the actions to be taken

Examples: C, Java, Lisp, Fortran, . . .

Declarative and Imperative

Declarative: the program is a description of the results we want

Examples: Prolog, ASP, Haskell (pattern matching), Mathematica (pattern matching), SQL (the SQL engine must find the best way of finding records that fit the query), . . .

Declarative and Imperative

- ASP: Answer Set Programming
- SQL: Structured Query Language

Declarative and Imperative

Feet

- Mathematica: You try to shoot yourself in the foot and then have to figure out why it didn't work

Declarative and Imperative

Feet

- Mathematica: You try to shoot yourself in the foot and then have to figure out why it didn’t work
- Mathematica (2): Your code to shoot yourself in the foot actually shoots someone else in the foot, but you think it works because you still feel pain

Declarative and Imperative

Feet

- Mathematica: You try to shoot yourself in the foot and then have to figure out why it didn't work
- Mathematica (2): Your code to shoot yourself in the foot actually shoots someone else in the foot, but you think it works because you still feel pain
- SQL: You cut your foot off, send it out to a service bureau and when it returns, it has a hole in it but will no longer fit the attachment at the end of your leg

Declarative and Imperative

Imperative languages are widely used

Declarative and Imperative

Imperative languages are widely used

Purely declarative are fairly rare and specialist

Declarative and Imperative

Imperative languages are widely used

Purely declarative are fairly rare and specialist

SQL is hugely widely used (it’s in your browser; it’s in your ’phone!)

Declarative and Imperative

Imperative languages are widely used

Purely declarative are fairly rare and specialist

SQL is hugely widely used (it’s in your browser; it’s in your ’phone!)

At a stretch, markup languages can be though of as declarative

Declarative and Imperative

Imperative languages are widely used

Purely declarative are fairly rare and specialist

SQL is hugely widely used (it’s in your browser; it’s in your ’phone!)

At a stretch, markup languages can be though of as declarative

Programmers have difficulty thinking in a declarative way, unless the problem is already declarative

Declarative and Imperative

Imperative languages are widely used

Purely declarative are fairly rare and specialist

SQL is hugely widely used (it’s in your browser; it’s in your ’phone!)

At a stretch, markup languages can be though of as declarative

Programmers have difficulty thinking in a declarative way, unless the problem is already declarative

But declarative languages are naturally parallel as they don’t describe sequences of operations

Parallel

Parallel computers are becoming ever more important

Parallel

Parallel computers are becoming ever more important

Most programming languages were designed in a uniprocessor world

Parallel

Parallel computers are becoming ever more important

Most programming languages were designed in a uniprocessor world

Some languages families are naturally parallel

- Declarative
- Functional

Parallel

Parallel computers are becoming ever more important

Most programming languages were designed in a uniprocessor world

Some languages families are naturally parallel

- Declarative
- Functional

As declarative languages don't specify how to do something, the system is free to do things in the most efficient way it can: and this includes in parallel

Parallel

The nature of functional languages is such that it is relatively easy to make parts run in parallel due to them having no global state, so no possibility of unexpected interference between different parts of the program

Parallel

The nature of functional languages is such that it is relatively easy to make parts run in parallel due to them having no global state, so no possibility of unexpected interference between different parts of the program

Unexpected interference is a major problem in parallel programs

Parallel

The nature of functional languages is such that it is relatively easy to make parts run in parallel due to them having no global state, so no possibility of unexpected interference between different parts of the program

Unexpected interference is a major problem in parallel programs

We all have had the experience of working on something with other people where someone else changes something while you are not looking, thereby messing up what you were trying to do

Parallel

The nature of functional languages is such that it is relatively easy to make parts run in parallel due to them having no global state, so no possibility of unexpected interference between different parts of the program

Unexpected interference is a major problem in parallel programs

We all have had the experience of working on something with other people where someone else changes something while you are not looking, thereby messing up what you were trying to do

Parallel programming is this a trillion times faster

Parallel

The nature of functional languages is such that it is relatively easy to make parts run in parallel due to them having no global state, so no possibility of unexpected interference between different parts of the program

Unexpected interference is a major problem in parallel programs

We all have had the experience of working on something with other people where someone else changes something while you are not looking, thereby messing up what you were trying to do

Parallel programming is this a trillion times faster

Plus a lot of other less obvious problems

Parallel

There have been many attempt to take sequential languages and tweak them to support parallelism

Parallel

There have been many attempt to take sequential languages and tweak them to support parallelism

OpenMP is C with a few hints added, e.g., “do this block in parallel”

Parallel

There have been many attempt to take sequential languages and tweak them to support parallelism

OpenMP is C with a few hints added, e.g., “do this block in parallel”

CUDA is C lightly modified to make programming of GPU (graphics) cards possible

Parallel

There have been many attempt to take sequential languages and tweak them to support parallelism

OpenMP is C with a few hints added, e.g., “do this block in parallel”

CUDA is C lightly modified to make programming of GPU (graphics) cards possible

OpenCL is C with a parallel library

Parallel

There have been many attempt to take sequential languages and tweak them to support parallelism

OpenMP is C with a few hints added, e.g., “do this block in parallel”

CUDA is C lightly modified to make programming of GPU (graphics) cards possible

OpenCL is C with a parallel library

All of these take advantage of the programmer’s familiarity with the legacy language (often C)

Parallel

There have been many attempt to take sequential languages and tweak them to support parallelism

OpenMP is C with a few hints added, e.g., “do this block in parallel”

CUDA is C lightly modified to make programming of GPU (graphics) cards possible

OpenCL is C with a parallel library

All of these take advantage of the programmer’s familiarity with the legacy language (often C)

Which may lure them into a false sense that they understand what they are doing

Parallel

Some languages are designed to be parallel from scratch

Parallel

Some languages are designed to be parallel from scratch

Occam (1993): derived from a mathematical notation for parallel processes. Was going to be big, but the hardware of the time couldn't cope

Parallel

Some languages are designed to be parallel from scratch

Occam (1993): derived from a mathematical notation for parallel processes. Was going to be big, but the hardware of the time couldn't cope

Strand: declarative, derived from Prolog

Parallel

Some languages are designed to be parallel from scratch

Occam (1993): derived from a mathematical notation for parallel processes. Was going to be big, but the hardware of the time couldn't cope

Strand: declarative, derived from Prolog

Erlang: functional. Used in real applications

Parallel

Some languages are designed to be parallel from scratch

Occam (1993): derived from a mathematical notation for parallel processes. Was going to be big, but the hardware of the time couldn't cope

Strand: declarative, derived from Prolog

Erlang: functional. Used in real applications

Go: intended to be a modern version of C, designed by the designers of C. Also called *Golang* to aid search on the Web

Parallel

Some languages are designed to be parallel from scratch

Occam (1993): derived from a mathematical notation for parallel processes. Was going to be big, but the hardware of the time couldn't cope

Strand: declarative, derived from Prolog

Erlang: functional. Used in real applications

Go: intended to be a modern version of C, designed by the designers of C. Also called *Golang* to aid search on the Web

Rust: designed to be a memory-safe replacement for C

Parallel

Feet

- Occam: You shoot both your feet with several guns at once
- Go: To shoot yourself in the foot you must first import the `unsafe` package
- Rust: you try to shoot yourself in the foot, but you can’t as the gun has immutably borrowed your foot

Parallel

And much more. There is a whole final-year Unit on parallelism!