Chapter 3.2
C++, Java, and Scripting Languages

"The major programming languages used in game development."

C++

- C used to be the most popular language for games
- Today, C++ is the language of choice for game development
    - C#, Objective C, ARM-C

C++: Strengths

- Performance
  - Control over low-level functionality (memory management, etc)
  - Can switch to assembly or C whenever necessary
  - Good interface with OS, hardware, and other languages

C++: Strengths

- High-level, object-oriented
  - High-level language features are essential for making today's complex games
  - Has inheritance, polymorphism, templates, and exceptions
  - Strongly typed, so it has improved reliability

C++: Strengths

- C Heritage
  - C++ is the only high-level language that is backwards-compatible with C
  - Has APIs and compiler support in all platforms
  - Easier transition for experienced programmers

C++: Strengths

- Libraries
  - STL (Standard Template Library)
    - Comprehensive set of standard libraries
  - Boost: widely used library with wide variety of functionality
  - Most commercial C++ libraries also available (Open GL, DirectX)

C++: Weaknesses

- Too low-level
  - Still forces programmers to deal with low-level issues (memory allocation)
  - Too error-prone
  - Attention to low-level details is overkill for high-level features or tools
  - If 90% of code is NOT performance critical, is it worth writing it at such a low level

C++: Weaknesses

- Too complicated
  - Because of its C heritage, C++ is very complicated
  - “Historical baggage” left behind in languages like Java, C#, Objective C.
  - Long learning curve to become competent with the language

C++: Weaknesses

- Lacking features
  - No reflection or introspection features
    - Object: What size are you?
  - No method of object serialization
    - Reading from media state of an object.
  - No native support for message passing
    - Not the same as “function calls”.
    - Custom buffer solution.

C++: Weaknesses

- Slow iteration
  - C++ is fully compiled into binary format from source code
  - Compiling large numbers of files can be very very slow
  - This will only become more of a problem as games become more complex
    - Workarounds exist (precompiled headers, dynamic linking, etc.)

C++: When to Use It?

- When performance is crucial (core features may include assembly, register)
- If your current code base is mostly C and C++
- If you have a lot of in-house expertise in C++
- Avoid using it for high-level code, such as tools

Java for Game Development

- Why use Java?
  - It's a high-level OO language that simplifies many C++ features
  - Adds several useful high-level features
  - Easy to develop for multiple platforms because of intermediate bytecode
  - Good library support
  - Lower learning curve

Java for Game Development

- Performance
  - Has typically been Java's weak point
    - Has improved in the last few years
    - Still not up to C++ level, but getting close
  - Uses Just-In-Time compiling and HotSpot optimizations
  - Also has access to native functionality
    - Now has high-performance libraries
    - JIN (Java Native Interface)

Java for Game Development

- Platforms
  - Well suited to downloadable and browser-based games
  - Strong player in mobile and handheld platforms
  - Possible to use in full PC games
    - More likely to be embedded into a game
  - Not currently used in consoles

Java in Game Development

- Commercial games using Java
  - Downloadable games like those from PopCap Games: Bejeweled, etc.
  - Online card/table/casino games
  - PC games using Java as a scripting language: *Star Wars Galaxies, DDO*
  - PC games fully written in Java: *You Don't Know Jack*, *Who Wants to Be a Millionaire*

Scripting Languages

- Why use scripting languages?
  - Ease and speed of development
    - Typing, memory management
    - May be simple enough for designers
  - Short iteration time
    - No need to compile
  - Code becomes a separate game asset (AI)
  - Offer additional features and are customizable
    - Reflection, serialization

Scripting Languages

- Drawbacks
  - Slow performance
  - Limited tool support (debugger)
  - Dynamic typing makes it difficult to catch errors
  - Can be awkward to interface with the game
  - Difficult to implement well

Scripting Languages

- Popular scripting languages
  - Python
  - Lua
  - Other off-the-shelf options such as Ruby, Perl, Javascript
  - Custom scripting languages
    - UnrealScript, QuakeC, NWNScript

Scripting Languages

- How to choose a scripting language
  - Consider whether you need one at all
  - What features do you need?
  - What kind of performance do you need?
  - What debugging facilities does the language have?
  - On what platforms does it need to run?
  - What resources and expertise are available?