History of Object-Oriented Programming

Chapter 1

Squeak: Object-oriented design with multimedia applications
Start of the Story: Late 60's and Early 70's

- Windows are made of glass, mice are undesirable rodents
- Good programming = Structured programming
  - Verb-oriented
Structured Programming

- Define tasks to be performed
- Break tasks into smaller and smaller pieces
  - Until you reach an implementable size
- Define the data structures to be manipulated
- Design how functions interact
  - What's the input
  - What's the output
- Group functions into components ("units" or "classes")
- Write the code
Object-oriented programming

- First goal: Model the objects of the world
  - Noun-oriented
  - Focus on the domain of the program

- Phases
  - Object-oriented analysis: Understand the domain
    - Define an object-based model of it
  - Object-oriented design: Define an implementation
    - Design the solution
  - Object-oriented programming: Build it
How’d we get from there to here?

How did we move from structured to object-oriented?

- Key ideas
  - Master-drawings in Sketchpad
  - Simulation “objects” in Simula

- Alan Kay and a desire to make software better
  - More robust, more maintainable, more scalable
Birth of Objects, 1 of 2

- Ivan Sutherland's Sketchpad, 1963
Sketchpad

- First object-oriented drawing program
- Master and instance drawings
  - Draw a house
  - Make two instances
  - Add a chimney to the master
  - Poof! The instances grow a chimney
- Other interesting features
  - 1/3 Mile Square Canvas
  - Invention of “rubber band” lines
  - Simple animations
Birth of Objects, 2 of 2

- **Simula**
  - Simulation programming language from Norway, 1966
  - Define an *activity* which can be instantiated as *processes*
  - Each process has its own data and behavior
    - In real world, objects don't mess with each others' internals directly
  - (Simulated) Multi-processing
    - No Universal Scheduler in the Real World
Alan Kay

- U. Utah PhD student in 1966
  - Read Sketchpad, Ported Simula
- Saw “objects” as the future of computer science
- His dissertation: Flex, an object-oriented personal computer
  - A personal computer was a radical idea then
Kay’s Insights

- “Computer” as collection of Networked Computers
- All software is simulating the real world
- Biology as model for objects
  - Bacterium has 120M of info, 1/500th of a Cell, and we have $10^{13}$ of these in us
  - What man-made things can scale like that?
    - Stick a million dog houses together to get the World Trade Center?
    - Internet does, but how can we make that the norm?
Birth of Objects

- Objects as models of real world entities
- Objects as Cells
  - Independent, indivisible, interacting -- in standard ways
- Scales well
  - Complexity: Distributed responsibility
  - Robustness: Independent
  - Supporting growth: Same mechanism everywhere
  - Reuse: Provide services, just like in real world
Features of Objects

- Encapsulation: Can't mess with the innards
- Inheritance: Objects can get structure (data) and behavior (methods) from another
- Aggregation: Objects can be created over-and-over and combined within other objects
"A Personal Computer for Children of All Ages"

- Flex, an object-oriented *personal* computer
Flex

- Enabled by Moore's Law
  - Imagining personal computing in 1969
- Logo, Sketchpad, and Simula
  - Learning representations and knowledge through programming them
  - Keyboard and drawing tablet
- Computer as meta-medium
  - The first medium to encompass other media
Xerox PARC: Learning Research Group: Smalltalk-72

- Alan Kay, Dan Ingalls, Adele Goldberg, Ted Kaehler
Goal: Dynabooks

- Small, handheld
- Wireless networking
- A Personal Computer for Children of All Ages
Smalltalk Inventions

- WIMP interface
  - overlapping Windows
  - Icons, even iconic programming
  - Pop-up menus
  - Mouse as Pointing device

- Object-oriented programming

- Multimedia authoring environment: Drawing, music, animations
How Smalltalk was Implemented

- Bytecode compiler
  - Machine language for a make-believe computer
- *Virtual machine* to create the make-believe computer
  - Invented years earlier by Burroughs
  - Used in UCSD Pascal, Java, Python, etc.
- Four files needed for this implementation:
  - VM
  - Image file (in bytecode)
  - Sources file (all sources always came along)
  - Changes file (added sources by user)
Virtual Machines

Native language of processor X

xyzzy xy xy

Real Processor X

The VM interprets the ab language to run on X.

VM

The VM is written in terms of xyz

Native language of processor A

aab ab ababa ba

Imaginary Processor A
Tradeoffs of Virtual Machines

- Advantages:
  - Can be easily ported
    - If VM is well-designed, small amount of code needs to be written for each platform
  - The VM can be very small
    - PowerMac 308K, Win32 328K, Linux-Intel 276K, SGI Irix 609K, Solaris 816K
  - Binary compatibility across platforms

- Distadvantages
  - Hard to make it efficient
    - But Moore's law makes efficient-enough easier all the time
1981: Xerox releases Smalltalk-80

- To prove portability, sends tapes to IBM, Sun, Apple, H-P, Tektronix
  - Smalltalk research starts up at all these places
  - Some Tektronix oscilloscopes have Smalltalk inside of them
- Spins off ParcPlace to market Smalltalk
  - Adele Goldberg goes to run the new company
  - Smalltalk-80 -> ObjectWorks -> VisualWorks
  - ParcPlace -> ObjectShare + Neometron and then Cincom
- Other Smalltalks: Digitalk's Smalltalk/V and Quasar's SmalltalkAgents
Back to the Future: Birth of Squeak

1995: Alan Kay, Dan Ingalls, Ted Kaehler are all at Apple
- Still want "A development environment in which to build educational software that could be used—and even programmed—by non-technical people and by children"
- Build on Open Source Software strengths
  - Use the distributed power of Internet-based programmers
Squeak Team

- Include John Maloney, Scott Wallace, and Kim Rose
  - Maloney from Self: O-O at nearly C speeds
  - Wallace: End-user programming, programming frameworks
  - Rose: Education practice and study
- Wanted a new Smalltalk, but didn't have to build from scratch
  - Apple had the original Smalltalk-80 still!
"Build everything in Smalltalk"

- "We determined that implementation in C would be key to portability but none of us wanted to write in C."

- Make the Apple Smalltalk portable again
- Write a new VM all in Smalltalk
- Write a Smalltalk-to-C translator
- Spit out the new VM on the new Smalltalk
- Whole process: 16 weeks
Squeak

- Even from the beginning, powerful implementation
  - 16 voice music synthesis, all in Smalltalk

- Released to the net, and ported to Windows and UNIX within five weeks
  - Apple license allows commercial apps, but system fixes must be posted

- Squeak Team moves to Disney
  - In the end, it's about media.

- Squeak today
  - Media: 3-D graphics, MIDI, Flash, sound recording
  - Network: Web, POP/SMTP, zip compression/decompress
  - Beyond Smalltalk-80: Exceptions, namespaces
Squeak for CS Education

Everything is written in Smalltalk!
Common Ancestry of Other O-O Languages

- **C++**
  - Bjarne Stroustrup wanted highly-efficient Simula
  - At Bell Labs, developed C-With-Classes
    - In 1984, C++ was born

- **Java**
  - 1991, Sun starts project to produce a language for embedded computing
    - James Gosling's Oak
  - On the Web, every client looks like a different kind of programmable toaster oven
  - In 1995, Oak became Java
C++ compared to Smalltalk

C++ is...
- Mixed paradigm programming: Objects, but can have functions, too
- Compiled to native code (Recall: Main goal is efficiency)
- Based on traditional functions and stack-based scoping
- Strongly typed
- Storage is controlled by the programmer

Smalltalk is...
- Dynamic (feels like an interpreter)
- Byte-code compiled
- Persistent objects
- Not at all typed
- Garbage collection: Storage is managed by the system
Java compared to the others

- Java looks like C++, but...
  - Mostly objects (no functions, but some primitive types)
  - Uses a VM
  - Objects are more like C++'s than Smalltalks
  - Even more strongly typed
  - Storage is handled by garbage collection