



# CS10 The Beauty and Joy of Computing

## Lecture #5 : Programming Paradigms

2010-09-15

UC Berkeley EECS  
Lecturer SOE  
Dan Garcia

### RENDERING ON CLOUD...

Intel has shown demos of the game Wolfenstein whose visual images weren't generated by your graphics card, but by a ton of servers running remotely.



[www.technologyreview.com/blog/editors/25751](http://www.technologyreview.com/blog/editors/25751)



UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (5)



Garcia, Fall 2010

[en.wikipedia.org/wiki/Programming\\_paradigm](http://en.wikipedia.org/wiki/Programming_paradigm)

## What are Programming Paradigms?

- "The concepts and abstractions used to represent the elements of a program (e.g., objects, functions, variables, constraints, etc.) and the steps that compose a computation (assignment, evaluation, continuations, data flows, etc.)."
- Or, a way to classify the style of programming.



UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (6)



Garcia, Fall 2010

## Most Languages Are Hybrids!

- This makes it hard to teach to students, because most languages have facets of several paradigms!
  - Called "Multi-paradigm" languages
  - Scratch too!
- It's like giving someone a juice drink (with many fruit in it) and asking to taste just one fruit!



UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (7)



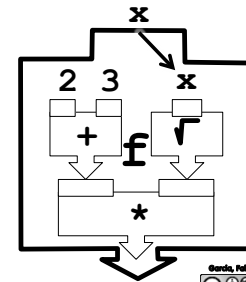
Garcia, Fall 2010

[en.wikipedia.org/wiki/Functional\\_programming](http://en.wikipedia.org/wiki/Functional_programming)

## Functional Programming (review)

- Computation is the evaluation of functions
  - Plugging pipes together
  - Each pipe, or function, has exactly 1 output
  - Functions can be input!
- Features
  - No state
    - E.g., variable assignments
  - No mutation
    - E.g., changing variable values
  - No side effects
- Examples
  - Scheme, Scratch BYOB

$$f(x) = (2+3) * \sqrt{x}$$



UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (8)

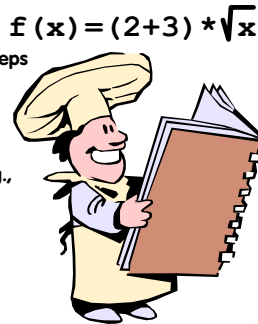


Garcia, Fall 2010

[en.wikipedia.org/wiki/Imperative\\_programming](http://en.wikipedia.org/wiki/Imperative_programming)

## Imperative Programming

- AKA "Sequential" Programming
- Computation a series of steps
  - Assignment allowed
    - Setting variables
  - Mutation allowed
    - Changing variables
- Like following a recipe. E.g.,
- Procedure  $f(x)$ 
  - ans = x
  - ans =  $\sqrt{\text{ans}}$
  - ans =  $(2+3) * \text{ans}$
  - return ans



$$f(x) = (2+3) * \sqrt{x}$$



UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (9)

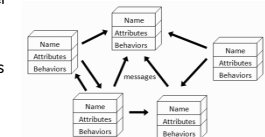


Garcia, Fall 2010

[en.wikipedia.org/wiki/Object-oriented\\_programming](http://en.wikipedia.org/wiki/Object-oriented_programming)

## Object-Oriented Programming (OOP)

- Objects as data structures
  - With methods you ask of them
    - These are the behaviors
  - With local state, to remember
    - These are the attributes
- Classes & Instances
  - Instance an example of class
  - E.g., Fluffy is instance of Dog
- Inheritance saves code
  - Hierarchical classes
  - E.g., pianist special case of musician, a special case of performer
- Examples: Java, C++



An object-oriented program consists of many well-encapsulated objects and interacting with each other by sending messages  
[www3.ntu.edu.sg/home/ahchua/programming/java/images/OOP-Objects.gif](http://www3.ntu.edu.sg/home/ahchua/programming/java/images/OOP-Objects.gif)



UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (10)




Garcia, Fall 2010

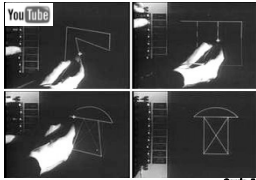
en.wikipedia.org/wiki/Sketchpad

## OOP Example : SketchPad

- Dr. Ivan Sutherland
  - "Father of Computer Graphics"
  - 1988 Turing Award ("Nobel prize" for CS)
  - Wrote Sketchpad for his foundational 1963 thesis
- The most impressive software ever written
- First...
  - Object-oriented system
  - Graphical user interface
  - non-procedural language



Spent the past few years doing research @ Berkeley in EECS dept!




UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (9)

en.wikipedia.org/wiki/Declarative\_programming

## Declarative Programming

- Express **what** computation desired **without specifying how** it carries it out
  - Often a series of assertions and queries
  - Feels like magic!
- Sub-categories
  - Logic
  - Constraint
    - We saw in Sketchpad!
- Examples: Prolog




Anders Hejlsberg  
"The Future of C#" @ PDC2008  
channel9.msdn.com/pdc2008/TL16/

UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (10)

## Declarative Programming Example

- Five schoolgirls sat for an examination. Their parents – so they thought – showed an undue degree of interest in the result. They therefore agreed that, in writing home about the examination, each girl should make one true statement and one untrue one. The following are the relevant passages from their letters:


<ul style="list-style-type: none"> <li><b>Betty</b> <ul style="list-style-type: none"> <li>Kitty was 2<sup>nd</sup></li> <li>I was 3<sup>rd</sup></li> </ul> </li> <li><b>Ethel</b> <ul style="list-style-type: none"> <li>I was on top</li> <li>Joan was 2<sup>nd</sup></li> </ul> </li> <li><b>Joan</b> <ul style="list-style-type: none"> <li>I was 3<sup>rd</sup></li> <li>Ethel was last</li> </ul> </li> <li><b>Kitty</b> <ul style="list-style-type: none"> <li>I came out 2<sup>nd</sup></li> <li>Mary was only 4<sup>th</sup></li> </ul> </li> <li><b>Mary</b> <ul style="list-style-type: none"> <li>I was 4<sup>th</sup></li> <li>Betty was 1<sup>st</sup></li> </ul> </li> </ul>	
--	--

UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (11)

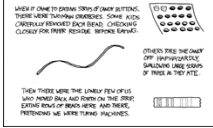
en.wikipedia.org/wiki/Turing\_completeness

## Turing Completeness

- A **Turing Machine** has an infinite tape of 1s and 0s and instructions that say whether to move the tape left, right, read, or write it
  - Can simulate any computer algorithm!
- A **Universal Turing Machine** is one that can simulate a Turing machine on any input
- A language is considered **Turing Complete** if it can simulate a **Universal Turing Machine**
  - A way to decide that one programming language or paradigm is just as powerful as another



Turing Machine by Tom Dunne



Xkcd comic "Candy Button Paper"

UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (14)

en.wikipedia.org/wiki/Programming\_paradigm

## Ways to Remember the Paradigms

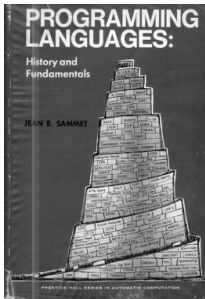
<ul style="list-style-type: none"> <li><b>Functional</b> <ul style="list-style-type: none"> <li>Evaluate an expression and use the resulting value for something</li> </ul> </li> <li><b>Imperative</b> <ul style="list-style-type: none"> <li>First <i>do this</i> and next <i>do that</i></li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><b>Object-oriented</b> <ul style="list-style-type: none"> <li>Send messages between objects to simulate the temporal evolution of a set of real world phenomena</li> </ul> </li> <li><b>Declarative</b> <ul style="list-style-type: none"> <li>Answer a question via search for a solution</li> </ul> </li> </ul>
--	--

[www.cs.aau.dk/~normark/prog3-03/html/notes/paradigms\\_themes-paradigm-overview-section.html](http://www.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html)

UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (15)

## Summary

- Each paradigm has its **unique benefits**
  - If a language is Turing complete, it is equally powerful
  - Paradigms vary in efficiency, scalability, overhead, fun, "how" vs "what" to specify, etc.
- Modern languages usually take the best from all**
  - E.g., Scratch
    - Can be functional
    - Can be imperative
    - Can be object-oriented
    - Can be declarative



UC Berkeley CS10 "The Beauty and Joy of Computing" : Programming Paradigms (16)