Programming Paradigms Lecture

- What are they?
  - Most are Hybrids!
- The Four Primary ones
  - Functional
  - Imperative
  - Object-Oriented
    - OOP Example: Sketchpad
  - Declarative
- Turing Completeness
- Summary

Most Languages Are Hybrids!

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

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.

Functional Programming (review)

- Computation is the evaluation of math 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

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 fix
  - ans = x
  - ans = \sqrt{ans}
  - ans = (2+3) * ans
  - return ans

Examples: Pascal, C
Object-Oriented Programming (OOP)

- Objects as data structures
  - With methods you ask of them
  - These are the instances
  - 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 is special case of musician, a special case of performer
- Examples: Java, C++

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

Turing Completeness

- A Turing Machine has an infinite tape of 0s and 1s 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 is decide that one programming language or paradigm is just as powerful as another

Ways to Remember the Paradigms

- Functional
  - Evaluate an expression and use the resulting value for something
- Imperative
  - First do this and next do that
- Object-oriented
  - Send messages between objects to simulate the temporal evolution of a set of real world phenomena
- Declarative
  - Answer a question via search for a solution

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

- 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

- Garciau (Fall 2009)
  - "The Beauty and Joy of Computing: Programming Paradigms"
  - "Programming Languages: History and Fundamentals"