CS123

Programming Your Personal Robot

Part 2: Event Driven Behavior

You Survived !

Smooth Sailing …

Topics

- 2.1 Event Driven Programming
  - Programming Paradigms and Paradigm Shift
  - Event Driven Programming Concept
    - Tkinter – as a simple example
  - More on threads
  - Implementation of a simple event driven behavior for Hamster
- 2.2 Finite State Machine
  - Concept of FSM
  - Implementation details (a simple FSM for Hamster)
  - FSM driven by an event queue
- 2.3 Related Topics and Discussion
  - Concept of HFSM and BT

  (if time allows, not needed for projects)

Class Structure

- Class 1: Basic Concept of Event Driven Programming
- Class 2: Implementation of Event Driven Programming
- Class 3: Concept and Implementation of FSM
- Class 4: Discussion of Related Topics
  - Hierarchical Finite State Machine and Behavior Tree (if time allows)

Objectives

- Learn Event Driven Programming
  - For develop behavior for Hamster
- Learn “How To Learn” A New Topic
  - You can find everything online
  - Ability to “Parse” large amount of information
  - Ability to “Synthesize”

2.0 Standard Hamster API

Standard Hamster API

- After homework is handed-in, a Stanford Hamster API will be given
- With this API, you can use both Mac and PC
- For PC users, you need to get a dongle
  - With some software installation, instruction available

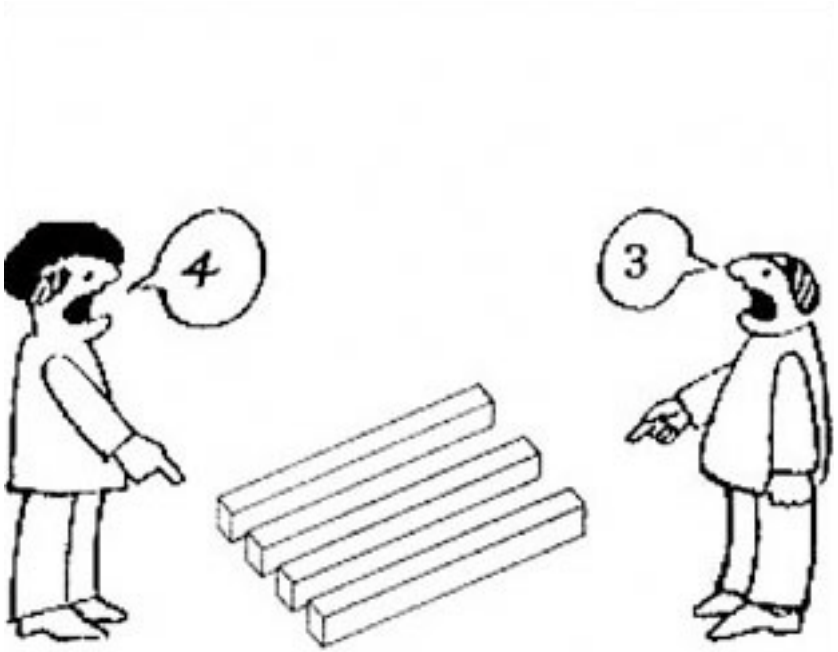
Standard Hamster API

- Set
  - Wheel
  - LED
  - Buzzer/musical_node
- Get
  - Proximity
  - Line
  - Light
  - Acceleration
  - ….

2.1 Event Driven Programming

What is Paradigm?

Paradigms: Ways of organizing thoughts



Different ways to look at things (the world)

What are Programming Paradigms

Various Programming Paradigms

- Procedural Programming
  - Von Neumann
  - Program flow: step-by-step specified
- Functional Programming
  - Mathematical foundation (return value, not side-effect)
- Object-Oriented Programming
  - Encapsulation
  - Manage complexity
- Imperative Programming
  - Specify what you want, not how to do it
- Declarative (Logic) Programming
  - Inference (Search)
  - Typical languages: Prolog/SQL
- Event-Driven Programming
  - Program flow driven by events
  - Typical examples: UI, Game, robots

Comparing Different Paradigms

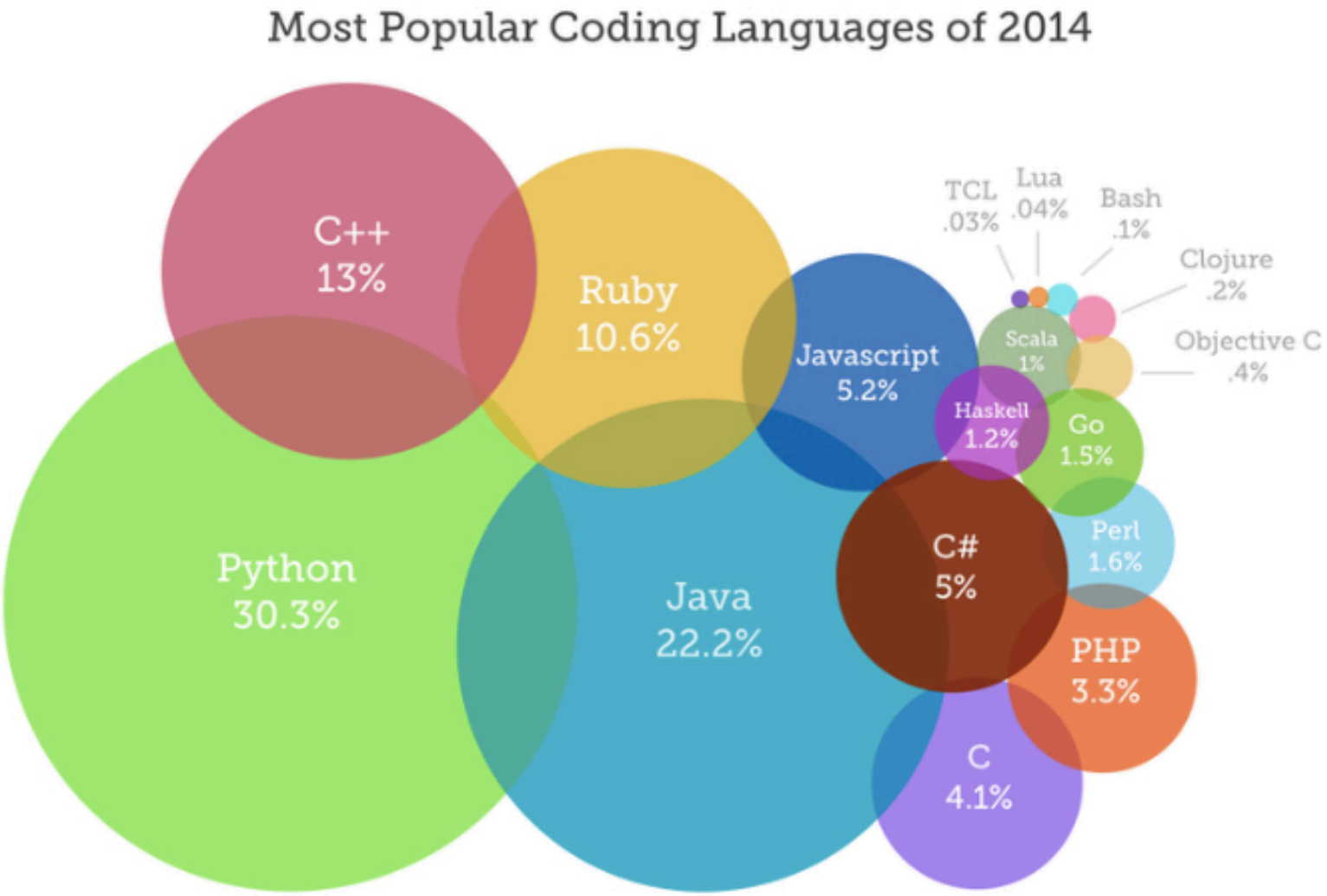
Different “axis” to organize/compare these paradigms

- Declarative vs. Imperative
  - What you want vs. How to do it
- Procedural vs. Event-Driven
  - Step-by-step vs. Event driven

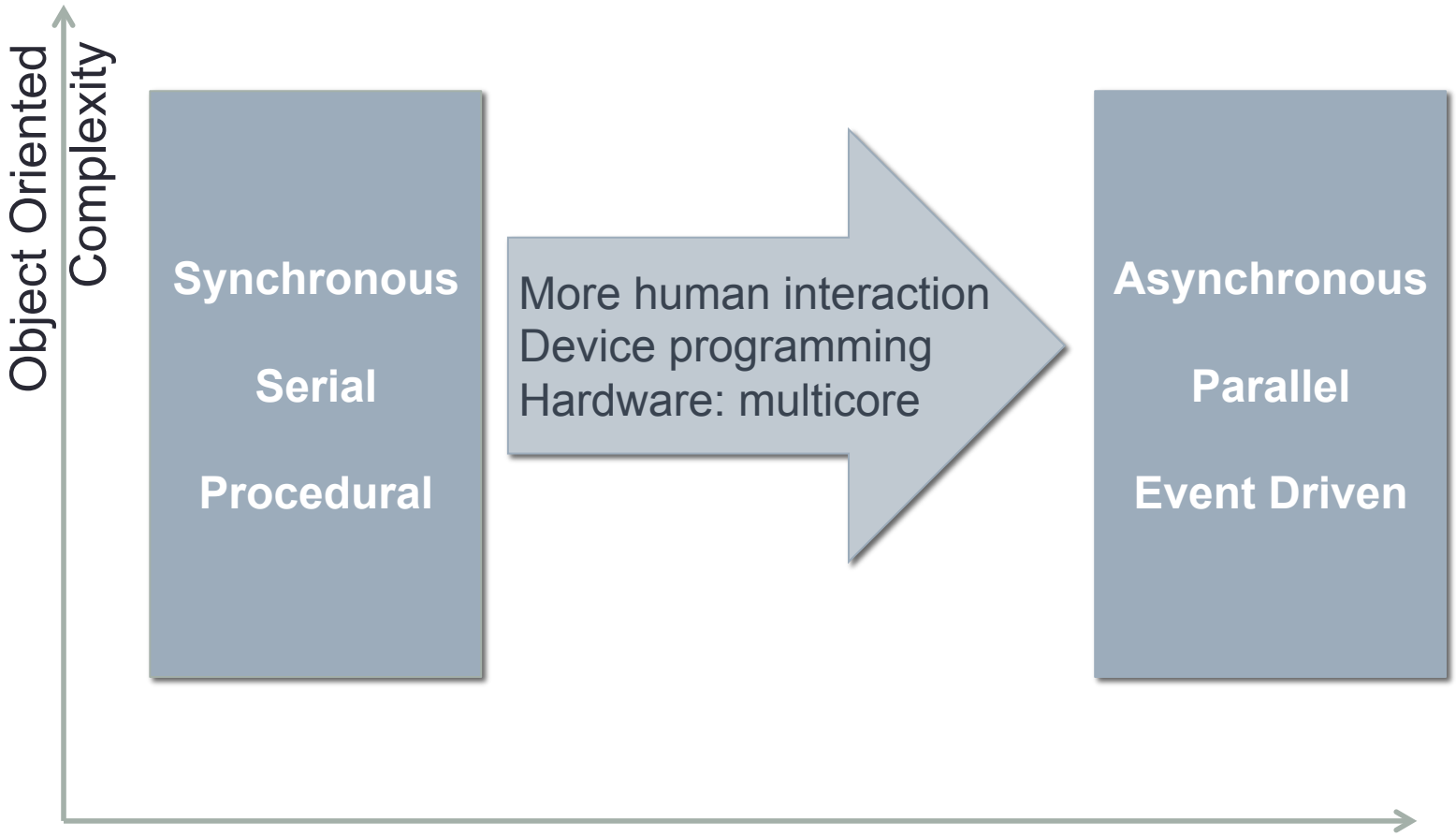
Programming Paradigms vs. Languages

- Not 1-to-1 Mapping
- Earlier languages are more “pure”
  - Pascal/C – Procedural/Imperative
  - Prolog/LISP – Declarative
- Newer languages: supports multiple paradigms
- Language trend: higher level of abstraction, less for the machine, more for the programmers
  - Earlier languages are closer to hardware
  - Driven by complexity of tasks
  - Enabled by increase hardware power

Programming Languages



Programming Paradigm “Shift”



Choosing A Paradigm: What to Consider?

- Suitable for problem formulation
- Ease of implementation
  - clarity
  - debugging
- Scalability
- Efficiency

An Example: Declarative Programming

```
Here are the resultant clauses:
-------------------------------

  male(james1).
  male(charles1).
  male(charles2).
  male(james2).
  male(george1).

  female(catherine).
  female(elizabeth).
  female(sophia).

  parent(charles1, james1).
  parent(elizabeth, james1).
  parent(charles2, charles1).
  parent(catherine, charles1).
  parent(james2, charles1).
  parent(sophia, elizabeth).
  parent(george1, sophia).
```

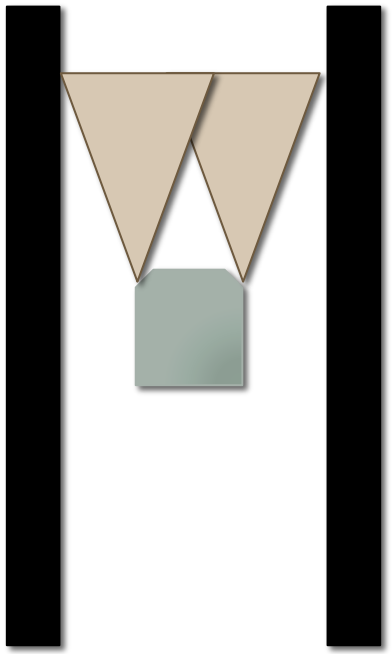
```
Here is how you would formulate the following queries:

     Was George I the parent of Charles I?
              Query: parent(charles1, george1).
     Who was Charles I's parent?
              Query: parent(charles1,X).
     Who were the children of Charles I?
              Query: parent(X,charles1).
```

How to Characterize Robot Programming

How to Characterize Robot Programming

- Open-loop Control
  - Execute robot actions without feedbacks
- Closed-loop Control
  - Adjust robot actions (motion) base on sensor feedbacks, thus compensate for errors

Closed-loop Control



Hallway following

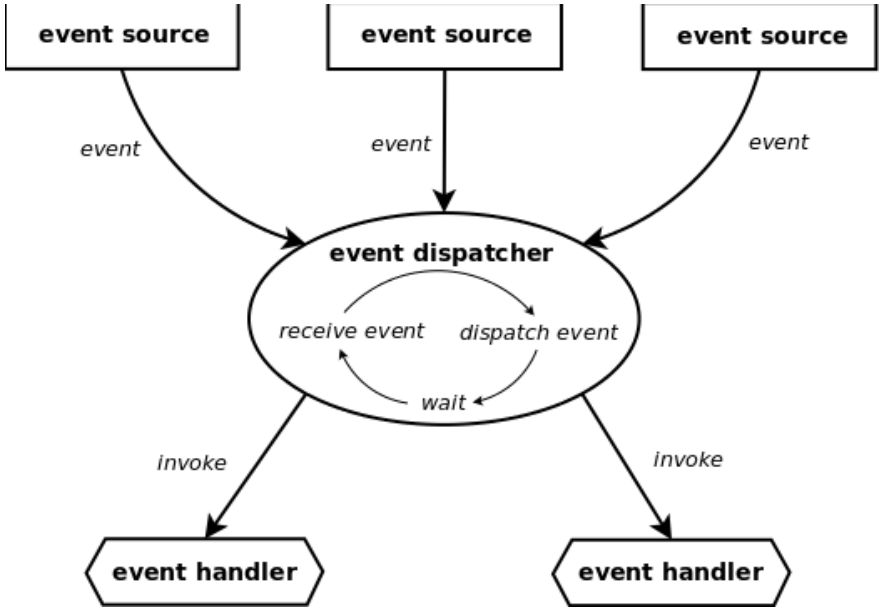
- Adjust robot actions (motion) base on sensor feedbacks, thus compensate for errors
- Necessary because of incomplete and imperfect model of the world, and because of control uncertainty

Event Driven Programming

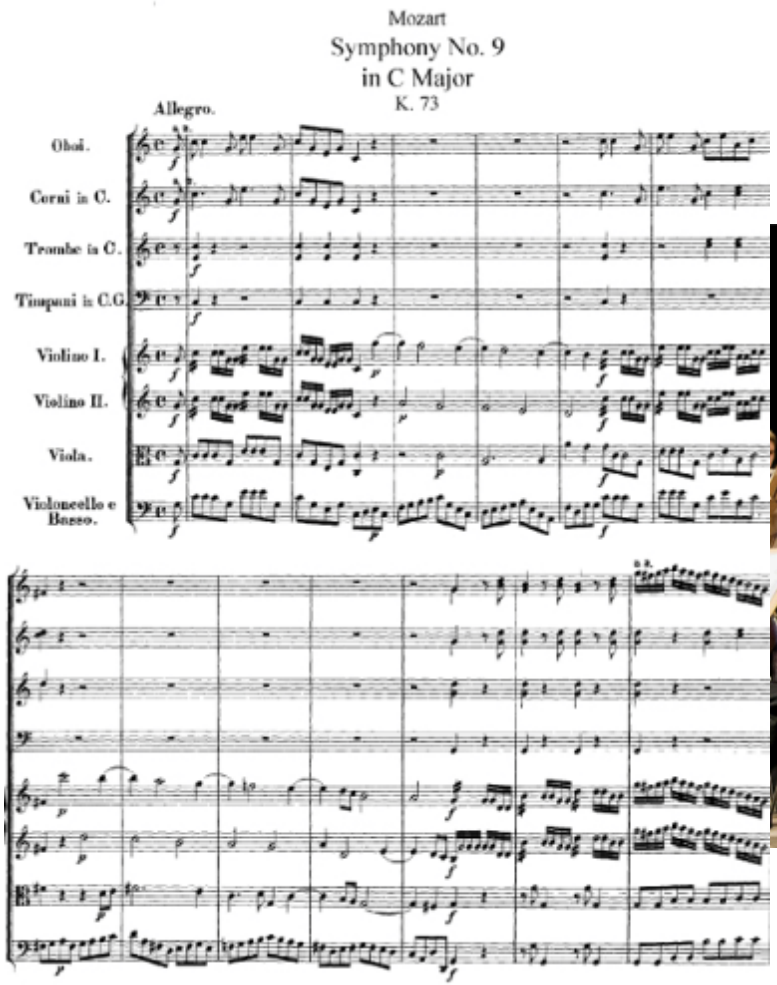
- Event Driven (Event-based) Programming is a programming paradigm is which the flow of the program is determined by events
- Common examples:
  - Games
  - Web UI
  - Robot

Event Driven Programming

- Event Dispatcher
  - Monitor events and “dispatch” to handlers
- Event Handlers
  - Program waits for events
  - When certain events happen, the program responds and does something (or decides to do nothing)



Or Like a Symphony

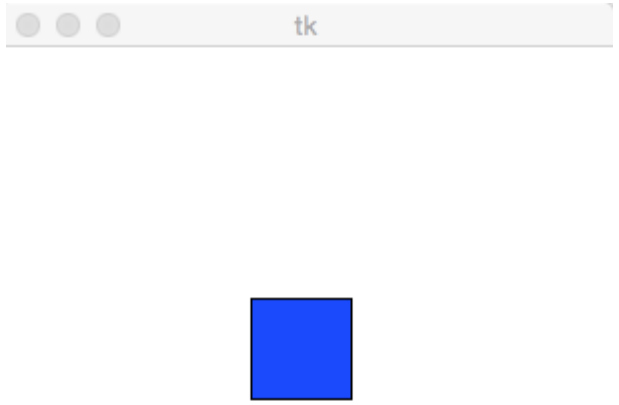


A Simple Example Using Tkinter

Tkinter is the de facto standard GUI package for Python

Two Reasons For Introducing **Tkinter**

- Yet another very simple example of event driven programming
  - Register an event with a callback function
- Very useful for Part 3
  - Visualizing Hamster's "world"



A Basic Tkinter Program

- create a root window
- create widgets within the root window
- customize widgets
- layout the widgets
- bind event handlers to widget events
- start the event loop

(see sample)

2.2 Event Driven Programming Implementation

2.3 Finite State Machine (FSM)

2.4 Topics and Discussion