Ada as a Foundation Programming Language: Starting Off on the Right Foot

Michael B. Feldman
Chair, ACM SIGAda Education Working Group
Professor, Department of Computer Science
School of Engineering and Applied Science
The George Washington University
Washington, DC 20052
(202) 994-5919 (voice)
(202) 994-4875 (fax)
mfeldman@seas.gwu.edu (Internet)
http://www.seas.gwu.edu/~mfeldman (WWW)

What's a Foundation Programming Language?

- Introduced in first or second course in undergrad curriculum ("CS1" or "CS2", in educator shorthand)
- (Usually) the first language students learn
- Introduced early enough to serve as a foundation for the rest of the curriculum
- A good foundation language helps student "start off on the right foot"

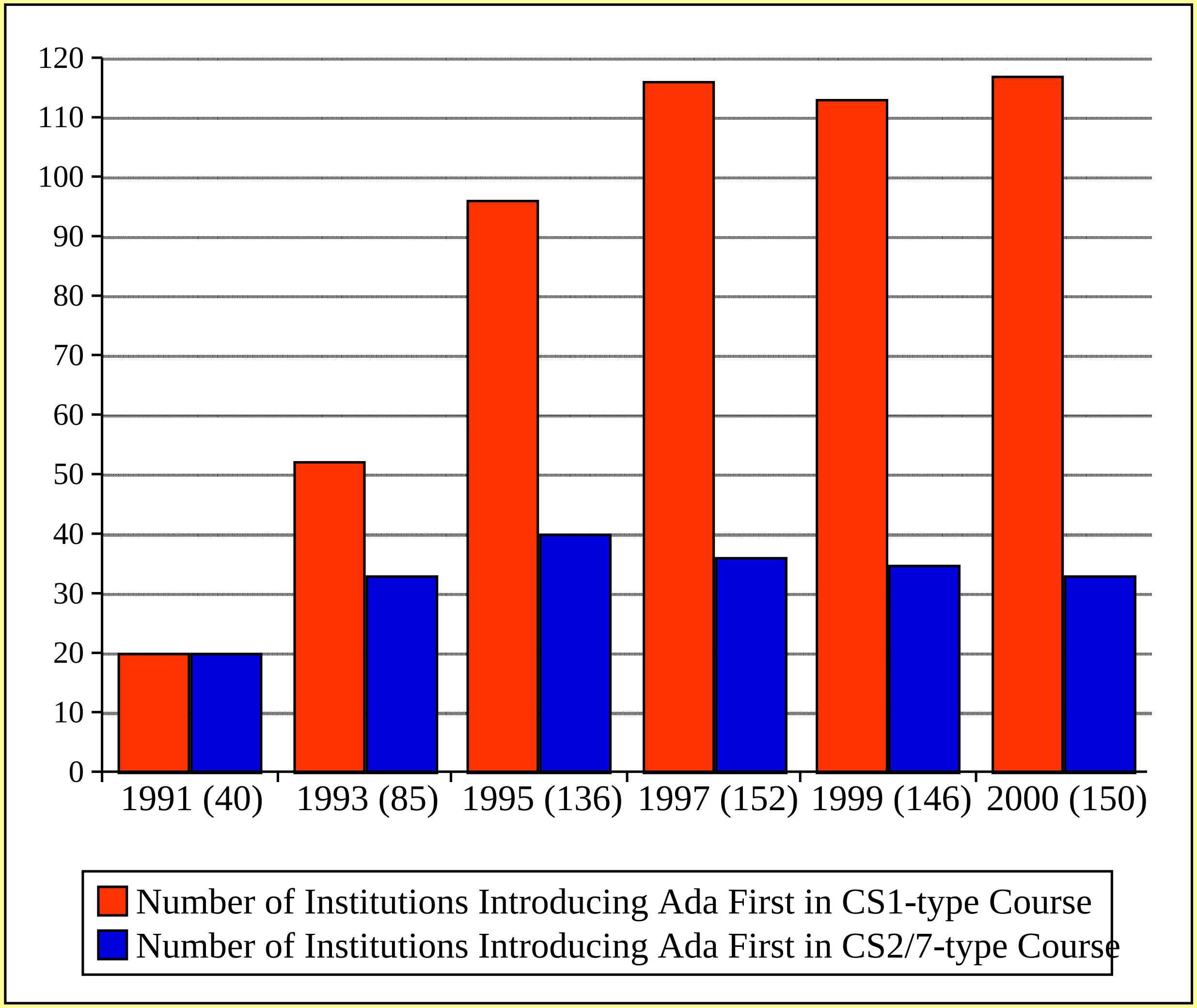
Why Ada as the Foundation Language?

- the disciplined control and data structures of Pascal
- modeling of scalar data (scalar subtypes)
- system composition structures (private types, packages, exceptions, generics, concurrency) derived from Simula, Clu, CSP
- classification structures in Ada 95 are an easily understood extension of the existing type and package structures
- a standard that is taken seriously—provides portability, and sets a good example of professionalism and maturity in the software field
- GNAT is a *very* friendly compiler!

How Does the Foundation Language Get Chosen?

- Faculties usually *vote* on curriculum decisions.
- There are always fads, bandwagons, factions, speculations, etc., just like any other political process.
- Smaller departments are easier to work with, simply because fewer “yes” votes are required to pass something.
- As in Congress, in a faculty it is difficult to make something happen but easy to make it *not* happen.

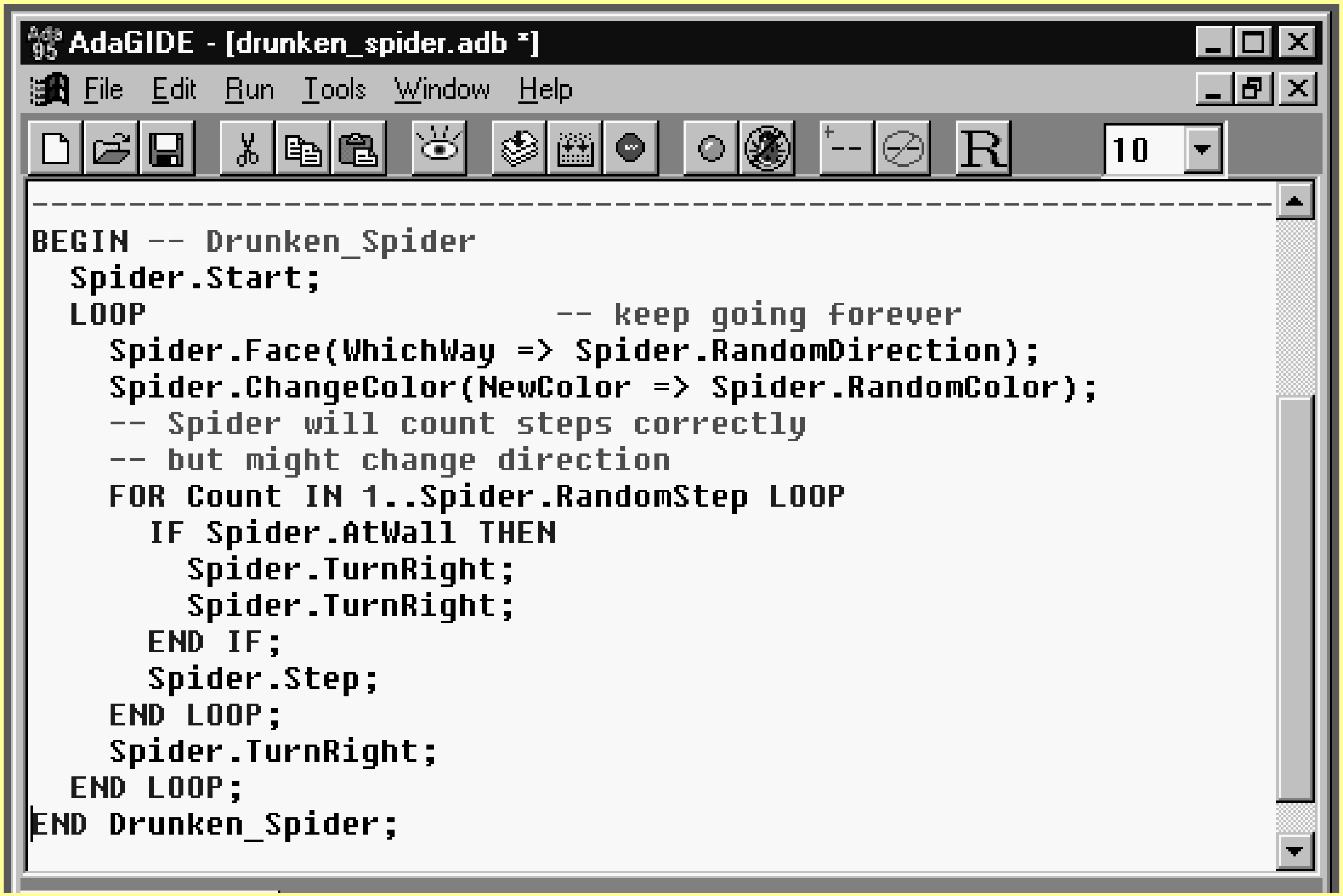
Ada as a Foundation Programming Language



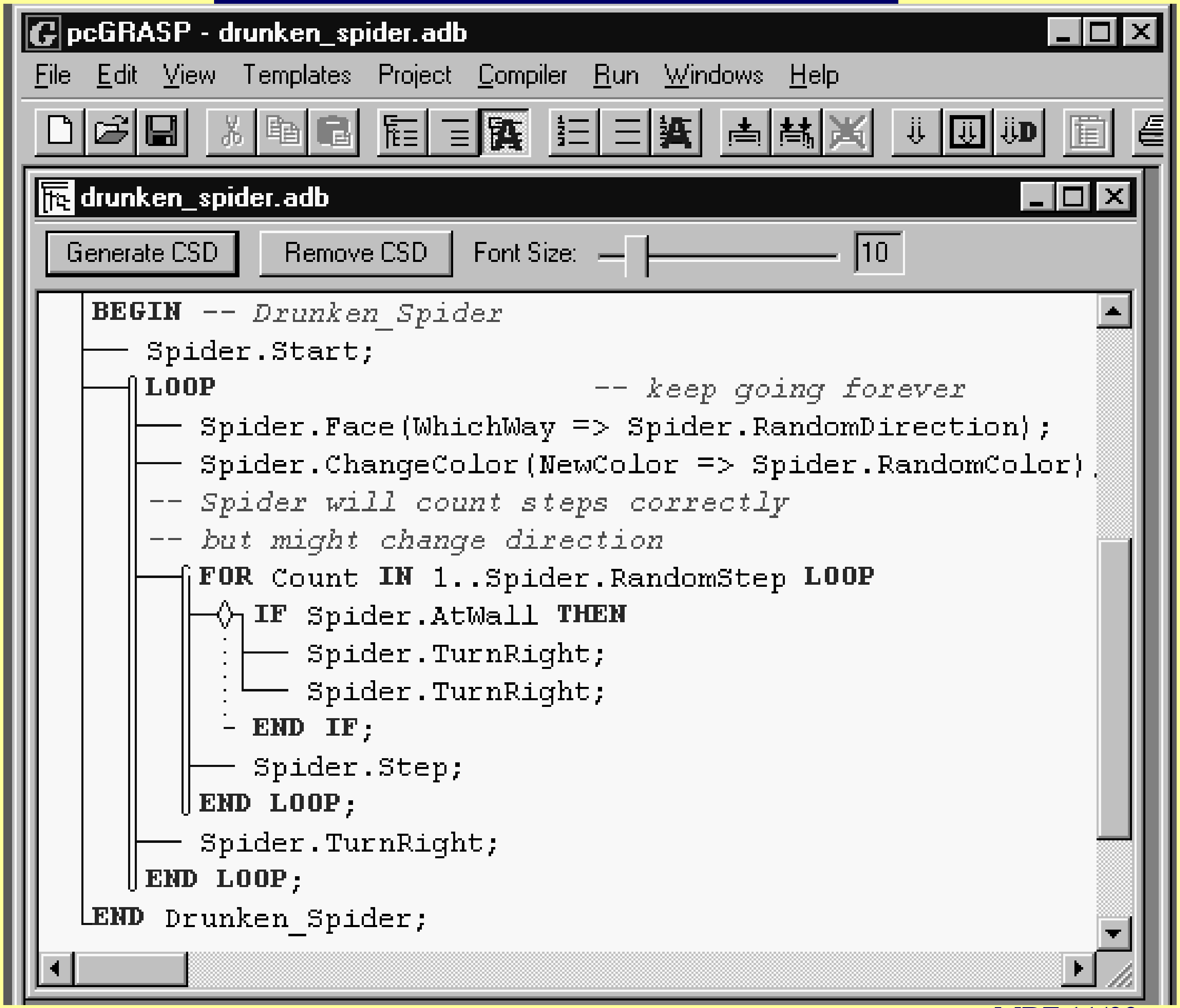
Continuing Innovation in Ada Foundation Courses

- New textbook editions:
  Feldman/Koffman (1999); Dale/Weems/McCormick (2000).
- AdaGIDE and GRASP GUIs both contain “beginner-friendly” features.
- University of Northern Iowa: switched *from* C++ *to* Ada. Local industry really likes the recent graduates, even for C++ projects.
- British Columbia Institute of Technology adopted Ada this year as foundation language.
- U.S. Air Force Academy is starting to use Ada in all-freshmen course to program LEGO Mindstorm robots.
- George Washington University research in introducing concurrent programming into a CS1-level course. (2 doctoral dissertations so far.)

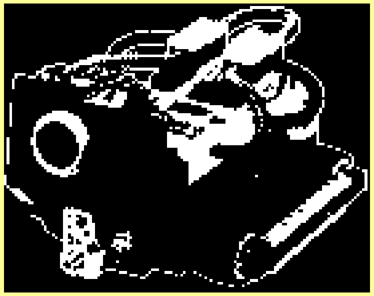
AdaGIDE (U.S. Air Force Academy)

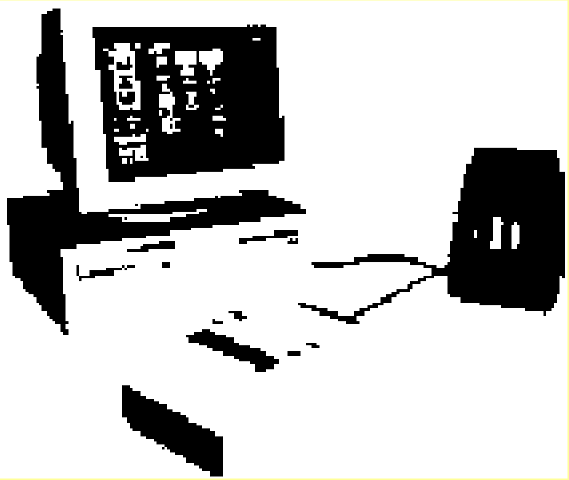


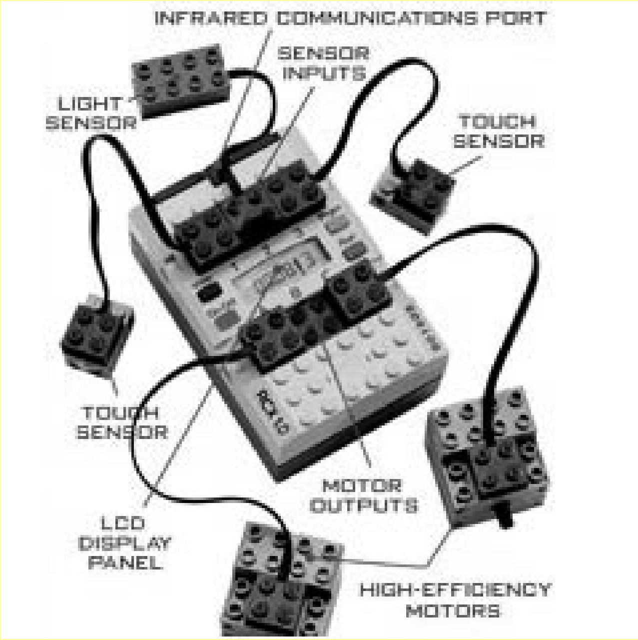
GRASP (Auburn University)



LEGO Mindstorms at U.S. Air Force Academy







GW Computer Science Education Research: Should We Teach Concurrency in Intro Courses?

- Language sophistication is making it possible to do—concurrency primitives are no longer “low level” and OS-dependent
- Some educators opine that because the world is concurrent, concurrency is not hard for beginners to understand
- Some opine that if students learn concurrency early, they won’t be stuck in a sequential “von Neumann bottleneck”
- A possible downside: “conservation of curriculum”; i.e., making room for concurrency means we may have to give less attention to other introductory subjects
- Caution and controlled experimentation is called for, to try to ensure that introducing a major new subject does not degrade student learning of traditional subjects

GW's Introductory Software Development Course for CS Majors

`http://www.seas.gwu.edu/~csci51`

- Csci 51, Introduction to Software Development
- Catalog Description:

  Introduction to the solution of problems on a digital computer using Ada. Structured programming concepts; proper documentation techniques; efficiency of programs; design of test data. Writing, debugging, and running programs in an interactive computing environment.

- 3 credits: 2.5 clock hours lecture; 1.5 clock hours laboratory per week

CSci 51 Demographics, Spring 2000

- Number of students: 79
  Students claiming previous programming experience: 31 (39%)
- Students indicating they were CS majors: 51 (64%)
  CS majors claiming previous programming experience: 22 (43%)
- Number of females: 19 (25% of population)
  Females with previous experience: 9 (47% of females)
  Female CS majors: 12 (23% of CS majors)
  Female CS majors with previous experience: 4 (33%)
- In most semesters, a student's position on the final grade curve is essentially independent of major or previous experience
- In most semesters, the withdrawal rate is no more than about 25% (this is low for typical CS1 courses)

GW's Introductory Software Development Course for CS Majors

- Course was taught with Pascal 1980-92, Ada 83 1992-95, Ada 95 1995-present
- Students complete a small lab exercise in each lab
- Students complete (typically) 8 software development projects, 1-2 weeks each
- "Spider"—simple turtle-graphics-type alogithm animation—used at start of course to teach algorithms, control structures, and exception handling
- GNU Ada 95 (GNAT) compiler used; default platform is Solaris but students can install on Mac, Linux, Windows
- Default Spider is simple 24 x 80 monochrome, but high-resolution color provided for Windows, Mac

A Spider Program

```
WITH Spider;
PROCEDURE Spiral IS
BEGIN -- Spiral
  Spider.Start;
  Spider.Face(WhichWay => Spider.RandomDirection);

  -- draw ten lines, starting in a random direction
  FOR Line IN 1..10 LOOP
    Spider.ChangeColor(NewColor => Spider.RandomColor);

    -- inner loop takes its bound from outer count
    FOR Count IN 1..Line LOOP
      Spider.Step;
    END LOOP;

    Spider.TurnRight;
  END LOOP;
  Spider.Quit;
END Spiral;
```

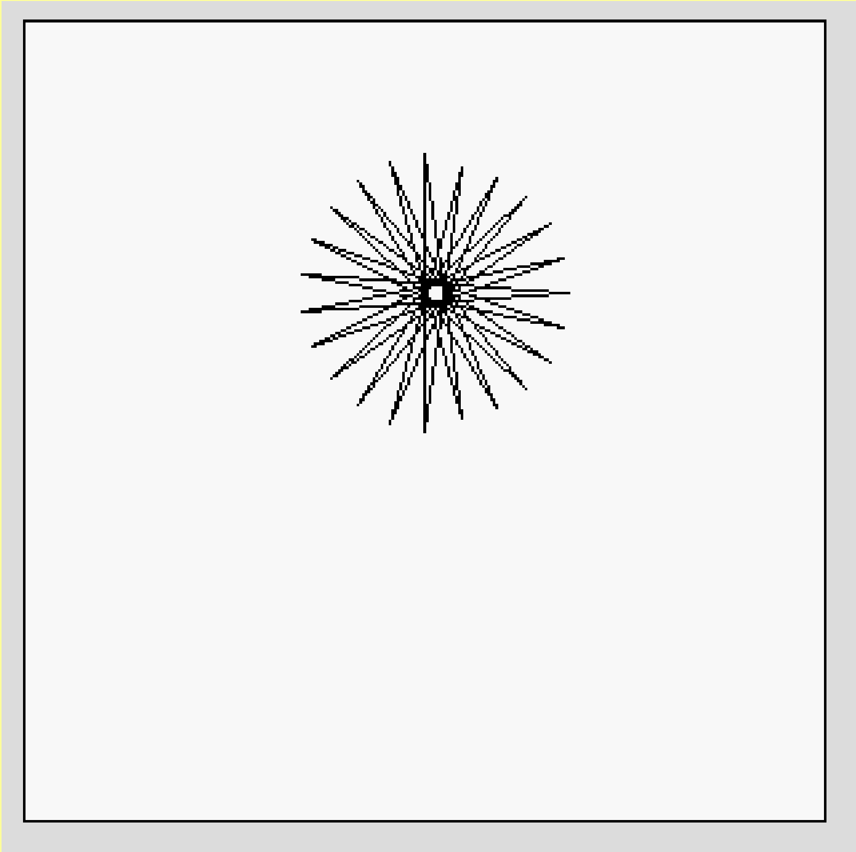
Spider Program Running

```
 ---          ---------------------------------------
| v |        |. . . . . . . . . . . . . . . . . . . .|
| . |        |. . . . . . . . . . . . . . . . . . . .|
 ---         |. . . . . . . . . . . . . . . . . . . .|
             |. . . . . . . . . . . . . . . . . . . .|
             |. . . . . . . . . . . . . . . . . . . .|
             |. . . . . . . . . . . . . . . * . . . .|
             |. . . . . B . . . . . . . . . . . . . .|
             |. . . . . B . . . . . . . G . . . . . .|
             |. . . . . B . K . . . . . G . . . . . .|
             |. . . . . B . K . K K B . G . . . . . .|
             |. . . . . B . K . G . B . G . . . . . .|
             |. . . . . B . K . . . B . G . . . . . .|
             |. . . . . B . K . . . . . G . . . . . .|
             |. . . . . B . . . . . . . G . . . . . .|
             |. . . . . B R R R R R R R R . . . . . .|
             |. . . . . . . . . . . . . . . . . . . .|
             |. . . . . . . . . . . . . . . . . . . .|
             |. . . . . . . . . . . . . . . . . . . .|
             |. . . . . . . . . . . . . . . . . . . .|
             |. . . . . . . . . . . . . . . . . . . .|
              ---------------------------------------
```

High-Resolution Polystars

```
WITH Spider_Hires; USE  Spider_Hires;
PROCEDURE Polystars IS
  PROCEDURE Polystar(Length: IN Steps; Sides: IN Positive) IS
  BEGIN
    FOR Side IN 1..Sides LOOP
      Step(HowMany => Length);
      TurnRight(HowFar => 180.0 - 180.0/Float(Sides));
    END LOOP;
  END Polystar;
BEGIN -- Polystars
  SetSpeed(Slow);
  FOR Count IN 3 .. 25 LOOP
    IF Count REM 2 = 1 THEN
      Start;
      Polystar(Length => 100, Sides => Count);
      Wait;
    END IF;
  END LOOP;
  Quit;
END Polystars;
```

High Resolution Polystars



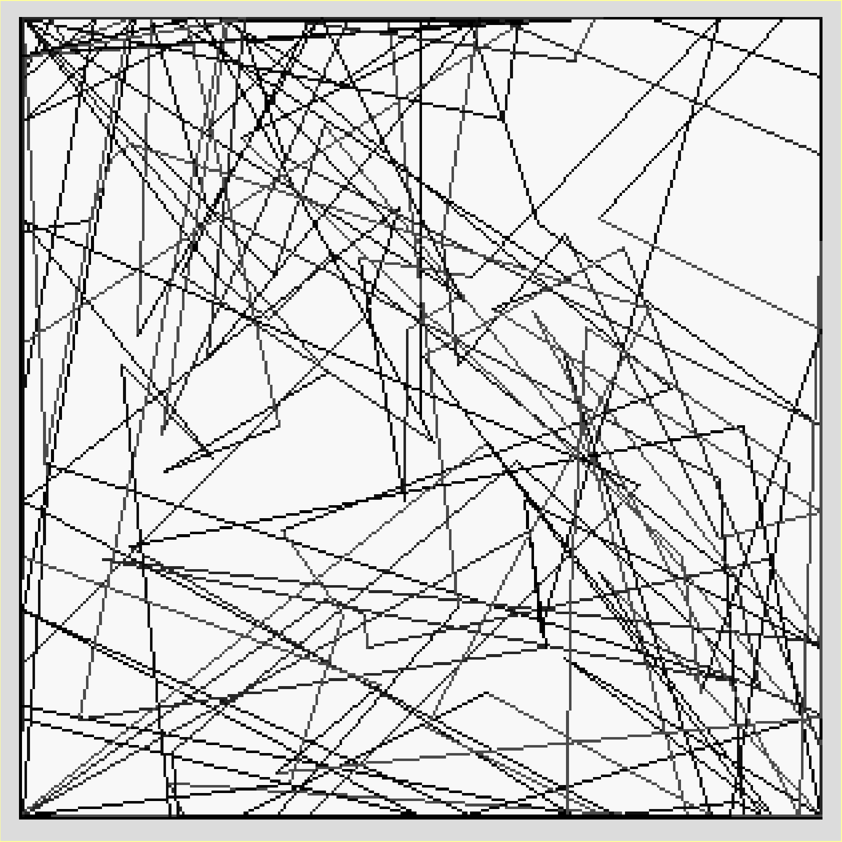
Drunken Spider

```
WITH Spider;
PROCEDURE Drunken_Spider IS
BEGIN -- Drunken_Spider
  Spider.Start;
  LOOP                     -- keep going forever
    Spider.Face(WhichWay => Spider.RandomDirection);
    Spider.ChangeColor(NewColor => Spider.RandomColor);
    -- Spider will count steps correctly
    -- but might change direction
    FOR Count IN 1..Spider.RandomStep LOOP
      IF Spider.AtWall THEN
        Spider.TurnRight;
        Spider.TurnRight;
      END IF;
      Spider.Step;
    END LOOP;
    Spider.TurnRight;
  END LOOP;
END Drunken_Spider;
```

Drunken Spider

```
  ---              ---------------------------------------
| v |             |. . . . . . . . . . . . . . . . . . . .|
| G |             |. . . . . . . . . . . . G . . . . . . .|
  ---             |. . . . . . . . . . . . G . . . . . . .|
                  |. . . . . . . . . . . . G . . . . . . .|
                  |. . . . . . . . . . . . G . . . . . . .|
                  |G G G G G G G G G G G G * . . . . . . .|
                  |. . . . . . . . G . . . . . . . . . . .|
                  |. . . . . . . . G . . . . . . . . . . .|
                  |. . . . . . . . G . . . . . . . . . . .|
                  |G G G G G G G G G G G G G G G G G G G .|
                  |. . G . . . . . G G . . . . . . . . . .|
                  |. . G . . . . . G G . . . . . . . . . .|
                  |. . G . . . . . G G . . . . G . . . . .|
                  |G G G G G G G G G G G G G G G . . . . .|
                  |. . . . . . . . G G G G G G G G G G G G|
                  |. . . . . . . . G G . . . . G . . . . .|
                  |. . . . . . . . G G . . . . G . . . . .|
                  |. . . . . . . . G G . . . . G . . . . .|
                  |. . . . . . . . G G . . . . G . . . . .|
                  |. . . . . . . . G G . . . . G . . . . .|
                    ---------------------------------------
```

Drunken Spider, High Resolution



Research Studies on Concurrent Programming in Intro Courses

- Bachus, D.Sc.Dissertation 1996. Can we teach concurrent programming (with Ada) to novices? Is concurrency necessarily an advanced topic?
- METHOD: non-credit GW summer course with voluntary students; soldiers at U.S. Army IT training school
- RESULT: Bachus showed that students without much programming experience can understand some fairly sophisticated examples of concurrency

Research Studies on Concurrent Programming in Intro Courses

- Lund, D.Sc. Dissertation 1999. Can we introduce concurrency into a CS1-level course? Will we lose anything by doing so?
- METHOD:

  teach Csci 51 with no concurrency content

  teach it again (twice) with unchanged original content but add concurrency

  measure student outcomes (projects, exams) on both nonconcurrency and concurrency content
- RESULT: Lund showed that we can introduce concurrency with no substantial degradation of outcomes

Typical Final Projects Using Concurrency in Csci 51 (and Csci 131, Algorithms and Data Structures I)

- Simulation of Highway Speed Monitor
- Multiple Drunken Spiders
- Dining Philosophers
- Simulation of a Bank in Operation

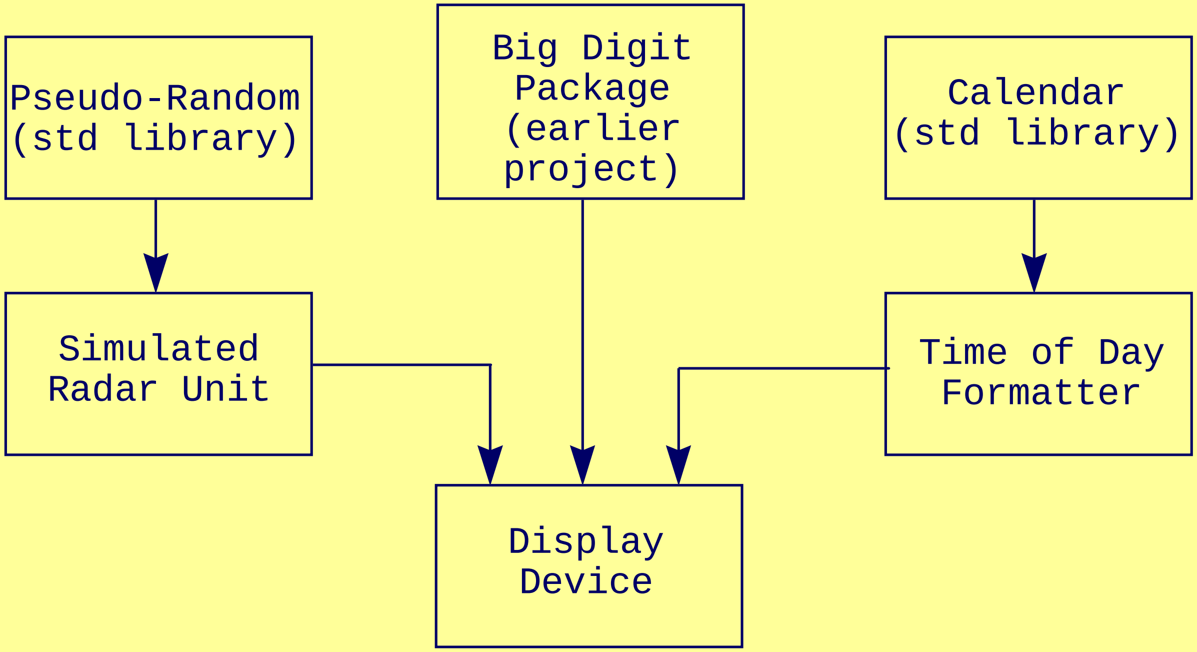
Highway Speed Monitor Simulation

```
                              Current time 16:32:05


              Y O U R   S P E E D   I S


                     XX          XXXXX
                XX   XX         XXXXXXX
                XX   XX        XX      XX
                XX   XX        XX      XX
                XX   XX        XX      XX
                XXXXXXXXX       XXXXXXXX
                XXXXXXXXX        XXXXXXX
                     XX                XX
                     XX        XX      XX
                     XX         XXXXXXX
                     XX          XXXXX
```

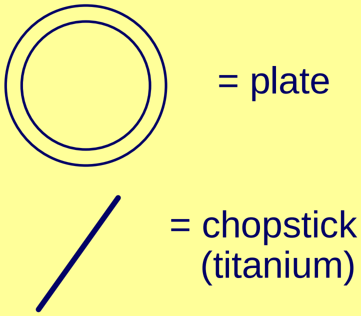
Highway Speed Monitor Simulation

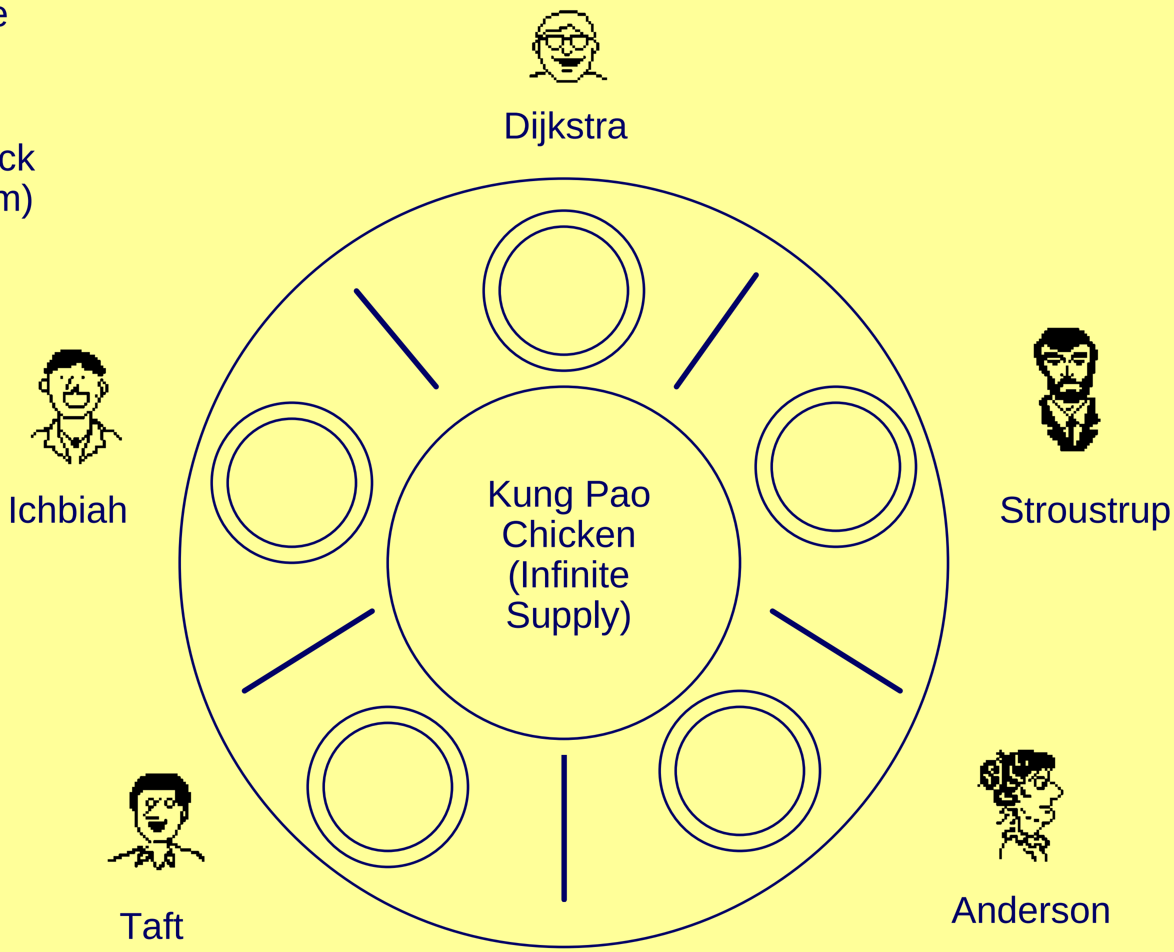


Drunken Spiders Family

```
 ---------------------------------------
|. . . . . . . . * . . . . . . . . . . .|
|. . . . . . . . . . . . . . . . . . . .|
|. . . . . . . . . . . . . . . . . . . .|
|. . . . . . . . . . . . * . . . . . . .|
|. . . . . . . . . . . . . . . . . . . .|
|. . . . . . . . . . . . . . . . . . . .|
|. . . . . . . . . . . . . . . . . . . .|
|. . . . . . . . . . . . . . . . . . . .|
|. . . . . . . . . . . . . . . . * . . .|
|. . . * . . . . . . * . . . . . . . . .|
|. . . . . . . . . . . . . . . . . . . .|
|* . . . . . . . . . . . . . . . . . . .|
|. . . . . . . . . . . . . . . . . * . .|
|. . . . . . . . . . . . . . . . . . . .|
|. . . . . . . . . . * . . . . . . . . .|
|. . . . . . . . . . . . . . . . . . . .|
|. . . . . . . * . . . . . . . . . . . .|
|. . . . . . . . . . . . . . . . . . . .|
|. . . . . . . . . . . . . * . . . . . .|
|. . . . . . . . . . . . . . . . . . . .|
 ---------------------------------------
```

Dijkstra’s Famous Dining Philosophers (1971)





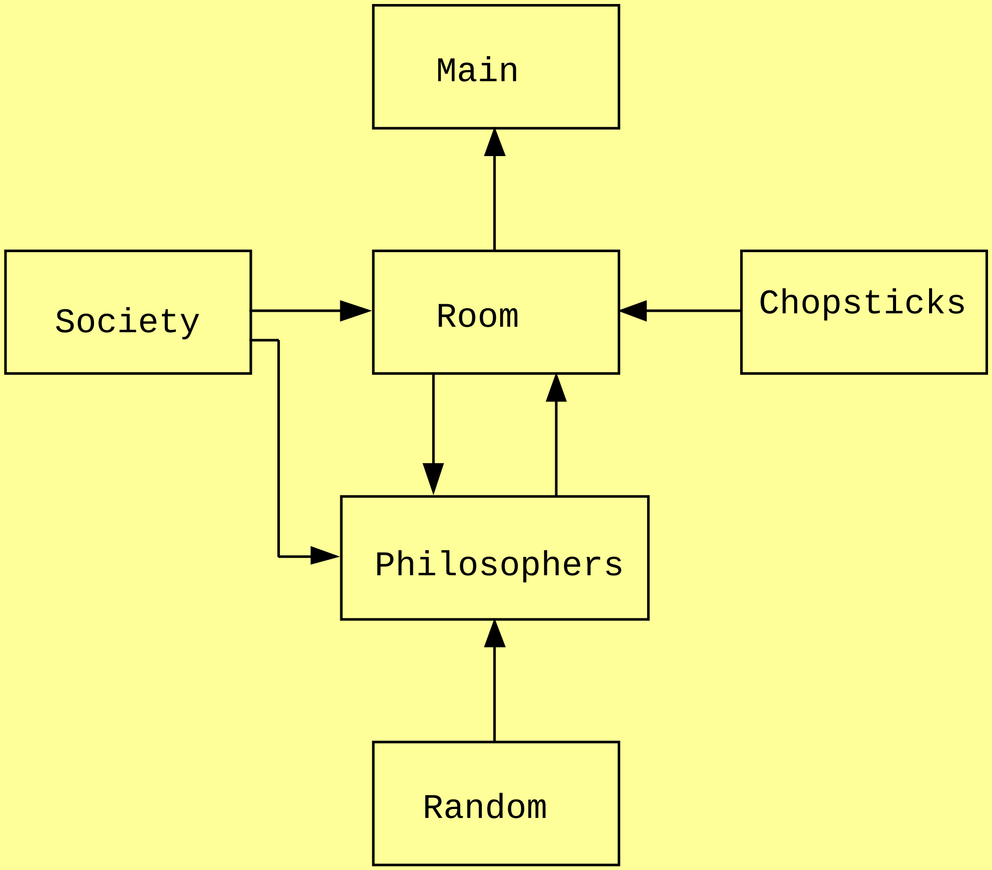
Dining Philosophers

```
                    +----------------------------+
                    |Edsger Dijkstra             |
                    |----------------------------|
                    |T = 13 Meal 1, 5 seconds.   |
                    |T = 18 Yum-yum (burp)       |
                    |T = 18 Thinking 8 seconds.  |
                    +----------------------------+

+----------------------------+             +----------------------------+
|Jean Ichbiah                |             |Bjarne Stroustrup           |
|----------------------------|             |----------------------------|
|T = 14 First chopstick 3    |             |T = 19 Meal 1, 9 seconds.   |
|T = 19 Second chopstick 4   |             |T = 18 First chopstick 1    |
|T = 19 Meal 2, 10 seconds.  |             |T = 19 Second chopstick 5   |
+----------------------------+             +----------------------------+

     +----------------------------+    +----------------------------+
     |Tucker Taft                 |    |Chris Anderson              |
     |----------------------------|    |----------------------------|
     |T = 19 Yum-yum (burp)       |    |T = 18 First chopstick 2    |
     |T = 19 Thinking 4 seconds.  |    |T = 13 Yum-yum (burp)       |
     |T = 9 Meal 2, 10 seconds.   |    |T = 13 Thinking 4 seconds.  |
     +----------------------------+    +----------------------------+
```

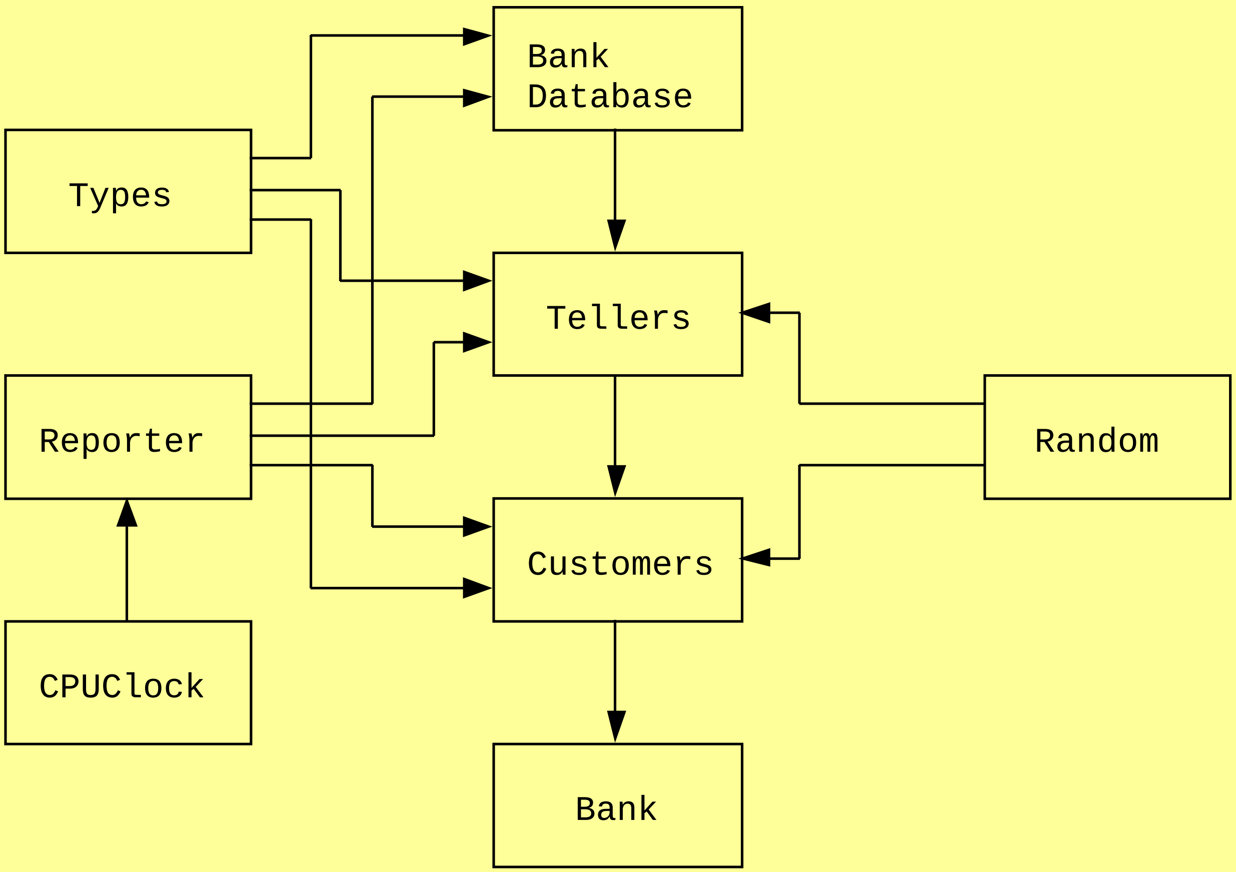
Module Dependencies for Dining Philosophers



Bank Simulation

```
T = 5   Teller B: Transaction will take 4 sec
T = 7   Teller A: Acct 2 - Balance is 0
T = 7   Teller A: Transaction will take 3 sec
T = 7 Account 2 alive.
T = 7 Account 2 will return after 3 sec
T = 9   Teller B: Acct 1 - Balance is 266
T = 10 Account 1 will return after 5 sec
T = 10   Teller A: Acct 3 - Balance is 0
T = 10   Teller A: Transaction will take 4 sec
T = 10 Account 3 alive.
T = 10 Account 3 will return after 3 sec
T = 10 Account 2 depositing 266 with Teller B
T = 10   Teller B: Transaction will take 1 sec
T = 11   Teller B: Acct 2 - Balance is 266
T = 12 Account 2 will return after 5 sec
T = 13 Account 3 depositing 266 with Teller B
```

Module Dependencies for Bank Simulation



Conclusions

- Material in foundation courses must be handled with care; students must complete the course well-prepared for the rest of the courses in their program
- GW *seniors* study Ada, Java, C, C++, and lots of other stuff in their four years. They tell us they are glad they began their program with Ada.
- The jury is still out on concurrency in intro courses but we are starting to understand the issues
- Computer Science Education research is challenging but rewarding