AdaLogo

Turtle Programming

Tran, Minh Cuong
tran@de.ibm.com

Pervasive Verification

November 8, 2010

Outline

1. Motivation
2. Language
3. Student Exercises
4. GUI
5. Interpreter and Debugger
6. Success Story

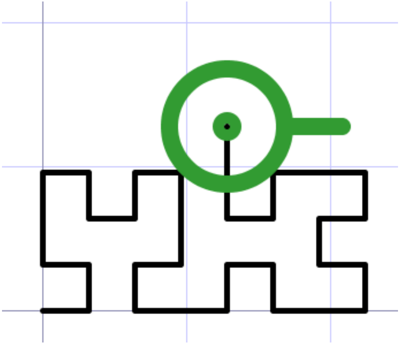
Motivation

- Learning Ada as your **first** programming language.
- Reducing the high learning curve at the first 2-3 weeks.
- Don't care about compiler and editor.
- You want to "see" and understand what your program is doing.

- Let's do it in AdaLogo!

Language

Reducing Ada to AdaLogo as a Small Subset

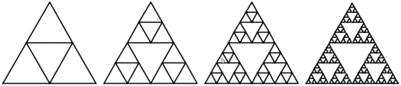


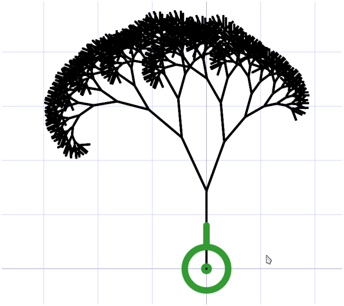
- Turtle action code:
  - forward(100); move_to(200,200); turn(90); turn_to(225); turtle_reset;
  - pen_up; pen_down;
  - put_line(“hallo“); put(100); new_line;
- Language
  - control flow: if else, exit, for loop, while loop
  - data types: integer and boolean
  - procedure with call-by-value

Student Exercises

Examples from Dr. Lewandowski

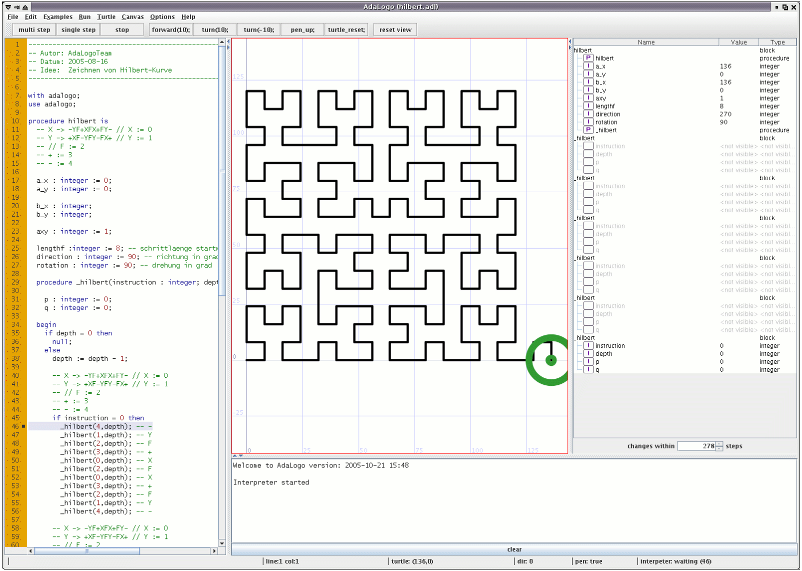
- Drawing simple and complex pictures
  - circle, triangle, square or spiral
  - π
- Recursion
  - Sierpinski triangle
  - Lime tree
  - Tower of Hanoi
- Analyzing unknown programs with debugger (program protocol)





GUI Design and Concepts

Editor, Canvas, Debugger and Terminal



```
------------------------------------------------------------
-- Autor: AdaLogoTeam
-- Datum: 2005-08-16
-- Idee:  Zeichnen von Hilbert-Kurve
------------------------------------------------------------

with adalogo;
use adalogo;

procedure hilbert is
  -- X -> -YF+XFX+FY- // X := 0
  -- Y -> +XF-YFY-FX+ // Y := 1
  -- // F := 2
  -- + := 3
  -- - := 4

  a_x : integer := 0;
  a_y : integer := 0;

  b_x : integer;
  b_y : integer;

  axy : integer := 1;

  lengthf :integer := 8; -- schrittlaenge startw
  direction : integer := 90; -- richtung in grad
  rotation : integer := 90; -- drehung in grad

  procedure _hilbert(instruction : integer; dept

    p : integer := 0;
    q : integer := 0;

  begin
    if depth = 0 then
      null;
    else
      depth := depth - 1;

      -- X -> -YF+XFX+FY- // X := 0
      -- Y -> +XF-YFY-FX+ // Y := 1
      -- // F := 2
      -- + := 3
      -- - := 4
      if instruction = 0 then
        _hilbert(4,depth); -- -
        _hilbert(1,depth); -- Y
        _hilbert(2,depth); -- F
        _hilbert(3,depth); -- +
        _hilbert(0,depth); -- X
        _hilbert(2,depth); -- F
        _hilbert(0,depth); -- X
        _hilbert(3,depth); -- +
        _hilbert(2,depth); -- F
        _hilbert(1,depth); -- Y
        _hilbert(4,depth); -- -

      -- X -> -YF+XFX+FY- // X := 0
      -- Y -> +XF-YFY-FX+ // Y := 1
      -- // F := 2
```

Name	Value	Type
hilbert		block
P hilbert		procedure
I a_x	136	integer
I a_y	0	integer
I b_x	136	integer
I b_y	0	integer
I axy	1	integer
I lengthf	8	integer
I direction	270	integer
I rotation	90	integer
P _hilbert		procedure
_hilbert		block
instruction	<not visible>	<not visibl...
depth	<not visible>	<not visibl...
p	<not visible>	<not visibl...
q	<not visible>	<not visibl...
_hilbert		block
instruction	<not visible>	<not visibl...
depth	<not visible>	<not visibl...
p	<not visible>	<not visibl...
q	<not visible>	<not visibl...
_hilbert		block
instruction	<not visible>	<not visibl...
depth	<not visible>	<not visibl...
p	<not visible>	<not visibl...
q	<not visible>	<not visibl...
_hilbert		block
instruction	<not visible>	<not visibl...
depth	<not visible>	<not visibl...
p	<not visible>	<not visibl...
q	<not visible>	<not visibl...
_hilbert		block
instruction	<not visible>	<not visibl...
depth	<not visible>	<not visibl...
p	<not visible>	<not visibl...
q	<not visible>	<not visibl...
_hilbert		block
I instruction	0	integer
I depth	0	integer
I p	0	integer
I q	0	integer

```
Welcome to AdaLogo version: 2005-10-21 15:48

Interpreter started
```

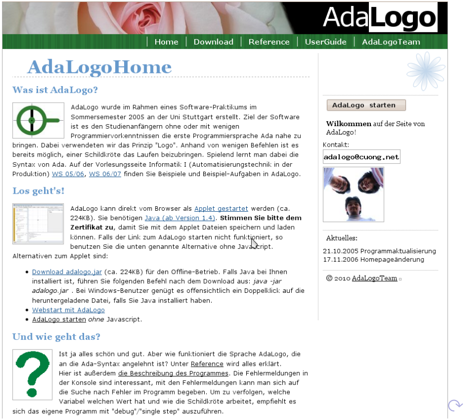
line:1 col:1 | turtle: (136,0) | dir: 0 | pen: true | interpeter: waiting (46)

GUI Design and Concepts

As Easy As Possible

- Editor
  - Syntax highlighting and breakpoint setting
- Canvas
  - Zoom and navigation
- Debugger (@see later)
- Terminal
  - Syntax and semantic errors
  - AST and user output
- Useful buttons
  - Debug with single step or breakpoint
  - Built-in examples
  - Current state of turtle

- Web page without installation: http://adalogo.cuong.net
- Reference and user guide



AdaLogoHome

Was ist AdaLogo?

AdaLogo wurde im Rahmen eines Software-Praktikums im Sommersemester 2005 an der Uni Stuttgart erstellt. Ziel der Software ist es den Studienanfängern ohne oder mit wenigen Programmiervorkenntnissen die erste Programmiersprache Ada nahe zu bringen. Dabei verwendeten wir das Prinzip "Logo". Anhand von wenigen Befehlen ist es bereits möglich, einer Schildkröte das Laufen beizubringen. Spielend lernt man dabei die Syntax von Ada. Auf der Vorlesungsseite Informatik I (Automatisierungstechnik in der Produktion) WS 05/06, WS 06/07 finden Sie Beispiele und Beispiel-Aufgaben in AdaLogo.

Los geht's!

AdaLogo kann direkt vom Browser als Applet gestartet werden (ca. 224KB). Sie benötigen Java (ab Version 1.4). **Stimmen Sie bitte dem Zertifikat zu**, damit Sie mit dem Applet Dateien speichern und laden können. Falls der Link zum AdaLogo starten nicht funktioniert, so benutzen Sie die unten genannte Alternative ohne Javascript.
Alternativen zum Applet sind:

- Download adalogo.jar (ca. 224KB) für den Offline-Betrieb. Falls Java bei Ihnen installiert ist, führen Sie folgenden Befehl nach dem Download aus: *java -jar adalogo.jar* . Bei Windows-Benutzer genügt es offensichtlich ein Doppelklick auf die heruntergeladene Datei, falls Sie Java installiert haben.
- Webstart mit AdaLogo
- AdaLogo starten *ohne* Javascript.

Und wie geht das?

Ist ja alles schön und gut. Aber wie funktioniert die Sprache AdaLogo, die an die Ada-Syntax angelehnt ist? Unter Reference wird alles erklärt.
Hier ist außerdem die Beschreibung des Programmes. Die Fehlermeldungen in der Konsole sind interessant, mit den Fehlermeldungen kann man sich auf die Suche nach Fehler im Programm begeben. Um zu verfolgen, welche Variabel welchen Wert hat und wie die Schildkröte arbeitet, empfiehlt es sich das eigene Programm mit "debug"/"single step" auszuführen.

Wilkommen auf der Seite von AdaLogo!

Kontakt:

adalogo@cuong.net

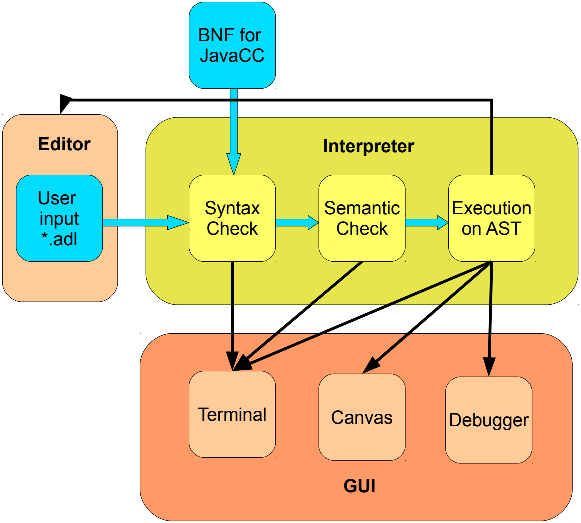
Aktuelles:

21.10.2005 Programmaktualisierung
17.11.2006 Homepageänderung

© 2010 AdaLogoTeam

Interpreter

Architecture Overview



Interpreter
Architecture Overview

Java Compiler Compiler

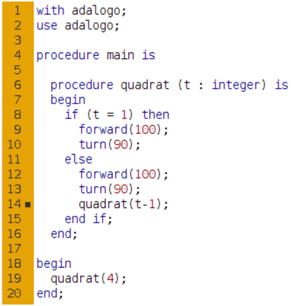
- LL(k) parser generation
- Grammar, self-written or reuse?
- 53 rules vs. 160 rules

Interpreter

- 3 Phase Checking and Interpreting
- Different interactions with Editor and other GUI components

Debugger

Program Example, Callstack and AST with Semicolon as Breakpoint

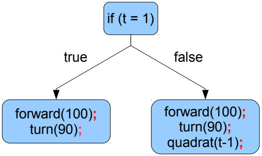


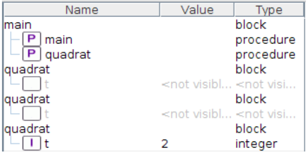
```
with adalogo;
use adalogo;

procedure main is

  procedure quadrat (t : integer) is
  begin
    if (t = 1) then
      forward(100);
      turn(90);
    else
      forward(100);
      turn(90);
      quadrat(t-1);
    end if;
  end;

begin
  quadrat(4);
end;
```





Name	Value	Type
main		block
P main		procedure
P quadrat		procedure
quadrat		block
t	<not visibl...	<not visi...
quadrat		block
t	<not visibl...	<not visi...
quadrat		block
I t	2	integer

Success Story

- Team: Hailang Thai, Minh Cuong Tran, Lesmana Zimmer
- Usage in ”Einfuehrung in die Informatik (autip)“ from Dr. Stefan Lewandowski at University Stuttgart since 2005.
- Web page usage peak at the beginning of every semester.
- Feel free to visit http://adalogo.cuong.net