Informatics = Basic Subject
Challenge and Chance

Walter Gander

ETH Zurich and BU Hong Kong

CEDI 2013

18 September 2013, Madrid

George Forsythe wrote 1963:

Founder of CS-Dept. Stanford
A Father of Silicon Valley

Machine-held strings of binary digits can simulate a great many kinds of things, of which numbers are just one kind. For example, they can simulate automobiles on a freeway, chess pieces, electrons in a box, musical notes, Russian words, patterns on a paper, human cells, colors, electrical circuits, and so on. To think of a computer as made up essentially of numbers is simply a carryover from the successful use of mathematical analysis in studying models . . . Enough is known already of the diverse applications of computing for us to recognize the birth of a coherent body of technique, which I call computer science. [a]

[a] Educational implications of the computer revolution. Applications of Digital Computers, W. F. Freiberger and William Prager (eds.), Ginn, Boston, 1963, pp. 166-178.

50 years later. . . Computers Determine Our Life
We Indeed Live in a Digital World

Communication: e-mail, cell-phone, sms, social networks: facebook, twitter, LinkedIn . . .

Writing: text-processing, spreadsheets, presentation tools

Reading: Google eBooks, e-Reader: Kindle, iPad, Sony Reader, Digital Book Index provides links to more than 165,000 full-text digital books
`http://www.digitalbookindex.com/about.htm`

Music: iTune, e-music, MP3

Radio and Television: digital, Internet

Photography: software has replaced chemically processed films

Search for Information: libraries, archives available on-line, Wikipedia

many more examples . . .

What About Computer Education in Schools?

Switzerland as Example:

- 1986: first attempt to introduce CS in Gymnasium
  - Rename "Descriptive Geometrie" → "Applied Mathematics"
  - CS part of mathematics. Contents: installation and use of PC, Programming (often PASCAL). Very few applications available.
  - Many teachers unable to cope, no curriculum, frequent system changes, poor reliability, student freaks with more knowlege
- 1995: Curriculum Reform "CS is used everywhere" → ICT
  - No need of programming anymore, many applications, Internet.
  - ECDL, Microsoft Office → good students are bored (this is CS?)
- since 2007: optional CS-classes [a] , little success (other electives easier!)
- 2013: proposed school reform with "ICT/Media" across disciplines

[a] Thanks to the support of the private foundation Hasler Stiftung

`http://www.fit-in-it.ch/`

European Commission: Digital Agenda [a]

Survey of Schools: ICT in Education → only USE of COMPUTERS!

Country Profile: Spain [b]

- *Equippment:* above EU mean
  laptops, high-speed broadband , levels of connectedness
- *Intensity of use:* around EU mean
- *Teachers' and students' confidence in their ICT skills:* below EU mean
- *ICT coordinators in schools:* more than EU mean
- *Digitally supportive schools* with ICT trained teachers, does not seem to translate into levels of ICT confidence or of use in lessons that might be expected.

---

[a] https://ec.europa.eu/digital-agenda/

[b] https://ec.europa.eu/digital-agenda/sites/digital-agenda/files/Spain%20country%20profile.pdf

Fundamentals of Informatics still missing in many schools

- Fundamentals (long lasting knowledge), algorithms, data structure, representation of text, pictures, music, recursion, ...
- George Forsythe 1961:
  In spite of the diversity of the applications, the methods of attacking the difficult problems with computers show a great unity, and the name of Computer Science is being attached to the discipline as it emerges. [a]
- Sandra Forsythe et al. published already 1969 a textbook for schools!

  Computer Science: A First Course [b]

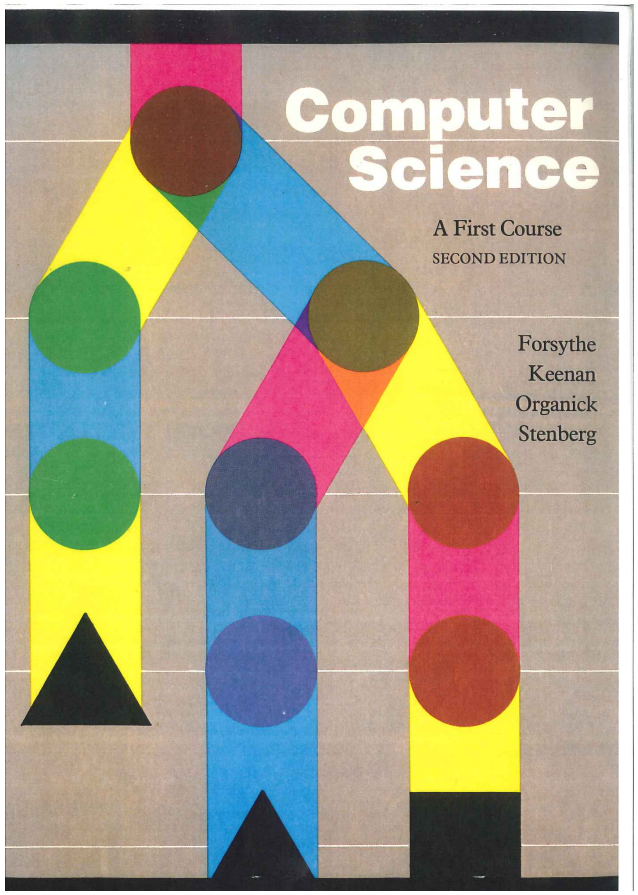
---

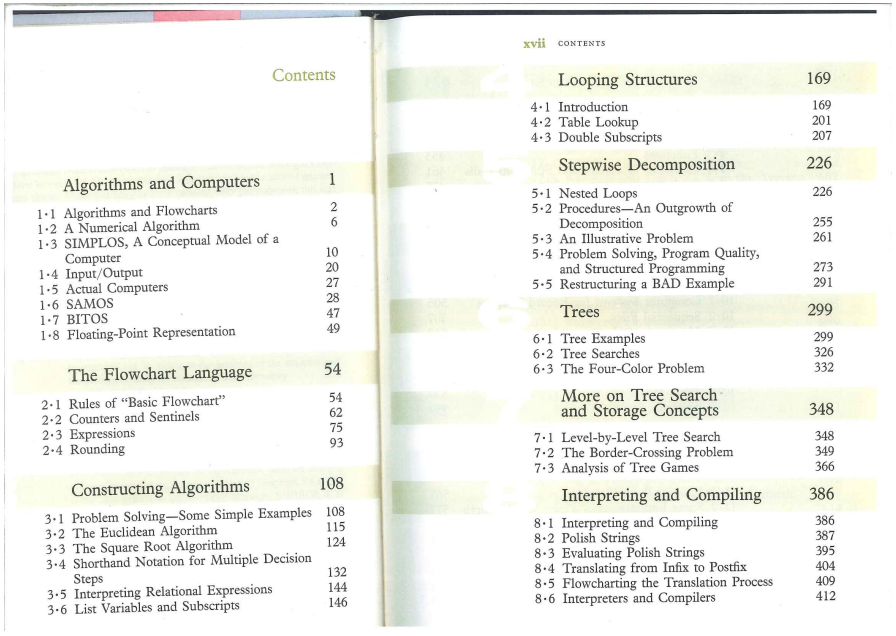
[a] Engineering students must learn both computing and mathematics. J. Eng. Educ. 52 (1961), 177-188.

[b] A.I. Forsythe, T.A. Keenan, E. I. Organick, and W. Stenberg, Computer Science: A First Course. Wiley & Sons, 1969 (1st ed.), 1975 (2nd ed.)

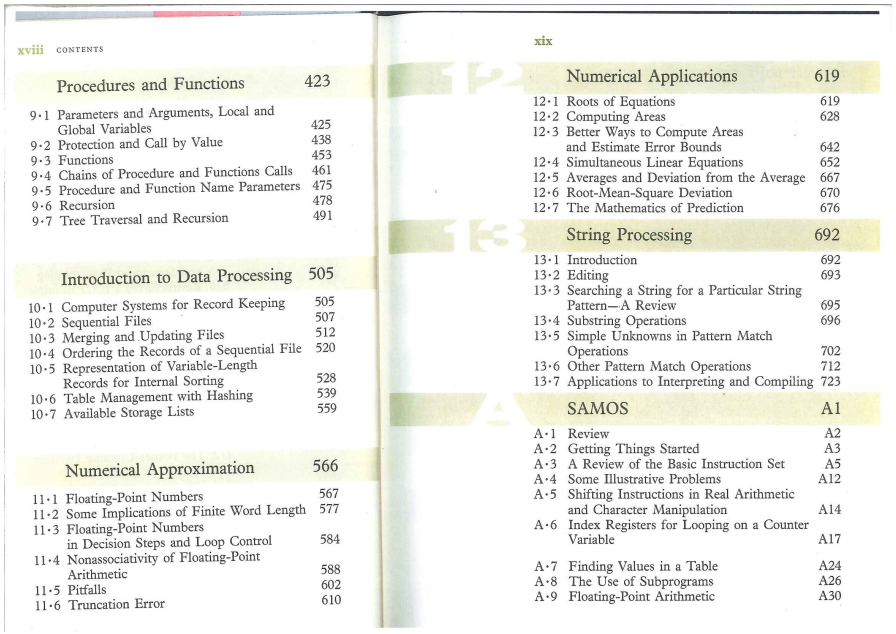
A Textbook of 1969/1975!

more than 40 years ago!!





Contents



Michael Gove
Secretary of State
for Education in UK

Speech of January 2012 [a]

- *the UK had been let down by an ICT curriculum that neglects the rigorous computer science and programming skills which high-tech industries need.*
- *In short, just at the time when technology is bursting with potential, teachers, professionals, employers, universities, parents and pupils are all telling us the same thing: ICT in schools is a mess.*

[a] `https://www.gov.uk/government/speeches/michael-gove-speech-at-the-bett-show-2012`

Michael Gove (cont.)

- *The new Computer Science courses will reflect what you all know: that Computer Science is a rigorous, fascinating and intellectually challenging subject.*
- *Initiatives like the Raspberry Pi scheme* [a] *will give children the opportunity to learn the fundamentals of programming with their own credit card sized, single-board computers.*
  It could bring the same excitement as the BBC Micro did in the 1980s[b]
- *Imagine the dramatic change which could be possible in just a few years, once we remove the roadblock of the existing ICT curriculum. Instead of children bored out of their minds being taught how to use Word and Excel by bored teachers, we could have 11 year-olds able to write simple 2D computer animations using an MIT tool called Scratch. (`http://scratch.mit.edu/`)*

[a] ICT↔CS:David Braben
[b] also: Commodore 64, Amiga ...

What is CS? What should be taught in schools?

- The "Informatics Europe&ACM Europe Working Group" defined in their report (April 2013): [a]

  Computer Science in Schools = Digital Literacy + Informatics

- Digital Literacy (often called ICT) is about the use of computers
- Informatics covers the science behind information technology
- Both parts should be taught compulsory in European schools for all students from first grade on.

---

[a] Informatics education: Europe cannot afford to miss the boat. Report of the joint *Informatics Europe & ACM Europe Working Group on Informatics Education*, April 2013 `http://www.informatics-europe.org/images/documents/informatics-education-europe-report.pdf`

Digital Literacy. The basic skills topics include:

- Being able to type.
- Being able to compose and revise documents, presentations and drawings.
- Understanding the fundamental measurement units (size and speed).
- Understanding the basic properties of digital files for text, audio, photos and movies.
- Knowing how to search, copy and store information as digital files.
- Knowing how to organize information by storing files in directories (copy, delete, rename) and to make backups.
- Being able to communicate via various media such as email, social networks and forums.
- Being aware of the currently most suitable software for these activities and able to search for suitable tools.

Digital Literacy (cont.)

The effective, safe and ethical use topics include:

- Knowing how to behave ethically on the web.
- Being able to distinguish between the real and the virtual.
- Being aware of security and fraud issues.
- Being able to exercise caution and to apply a critical mind when dealing with information gathered on the Internet.
- Knowing how to use information found on the Internet properly and ethically for one's own work; in particular, understanding the difference between legitimate reuse and plagiarism.
- Being familiar with privacy issues; knowing how to respect others' privacy and to protect one's own privacy with IT tools.

Informatics in School

Goals:

- Understand the principles and functioning of todays digital world
- Train the students in constructive problem solving
  - analyze a task or problem, model and formulate it mathematically
  - search for a way to solve it, find or design an algorithm
  - program
  - run the program: let the computer work, maybe correct, modify the program
- Train "Computational Thinking"

Computational Thinking: Topic of General Education.

- Definition by Jan Cuny, Larry Snyder, and Jeannette M. Wing [a]

  Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.

- What is Computational Thinking? [b]
  (Singapore Management University!)
  Computational thinking is about problem solving that uses fundamental concepts in computer science, such as abstraction, decomposition, recursion, heuristic reasoning, just to name a few. It can be used to algorithmically solve complex problems of scale, and is often used to realize large improvements in efficiency.

[a] Carnegie Mellon University, USA, `http://www.cs.cmu.edu/~CompThink/`
[b] `http://sis.smu.edu.sg/computationalthinking`

Programming — the Fundamental of Informatics!

George Forsythe 1959: *The automatic computer really forces that precision of thinking which is alleged to be a product of any study of mathematics.*[a]

George Forsythe 1966: *The major thing which distinguishes computer science from other disciplines is its emphasis on algorithms.*

There are few problems for which a good algorithm of probable permanent value is known... Small details are of the greatest importance... The development of excellent algorithms requires a long time, from discovery of a basic idea to the perfection of the method... A useful algorithm is a substantial contribution to knowledge. Its publication constitutes an important piece of scholarship. [b]

[a] The role of numerical analysis in an undergraduate program. Amer. Math. Monthly 66 (1959), 651-662.

[b] Algorithms for scientific computation. CACM 9 (Apr. 1966), 255-256.

Teaching a Machine

If you want to learn something, teach it. You are successful if people understand. However, they may say they understand even if they don't.

The ultimate test if you are doing well is to teach it to a machine!

(not literally)

Don Knuth

Swiss Olympiad in Informatics

ETH Jan 14, 2012

Programming a machine is part of our culture

Berufsbildungskonferenz
Nov 9, 2012, Bern

Mauro Dell'Ambrogio
Secretary of State
Education and Research
Switzerland

President Obama: [a]

High Schools Should Offer Programming and Graphic Design Courses [b]

> *Given how pervasive computers and the Internet is now and how integral it is in our economy and how fascinated kids are with it, I want to make sure that they know how to actually produce stuff using computers and not simply consume stuff*
>
> ...
>
> *We're going to start setting those programs in our high schools, not waiting to go to community college.*

---

[a] Fireside Hangout on Google+, Mountain View, Feb 2013

[b] `http://tinyurl.com/pgf2cx2`

Everybody Should Learn to Program `http://www.code.org/`

Leaders and trendsetters agree more students should learn to code

President Bill Clinton
"At a time when people are saying "I want a good job - I got out of college and I couldnt find one," every single year in America there is a standing demand for 120,000 people who are training in computer science."

Marco Rubio
Senator, Florida
"Computer programmers are in great demand by American businesses, across the tech sector, banking, entertainment, you name it. These are some of the highest-paying jobs, but there are not enough graduates to fill these opportunities."

Bill Gates
Chairman, Microsoft
"Learning to write programs stretches your mind, and helps you think better, creates a way of thinking about things that I think is helpful in all domains."

Mark Zuckerberg
Founder, Facebook
"Our policy at Facebook is literally to hire as many talented engineers as we can find. There just aren't enough people who are trained and have these skills today."

will.i.am
Musician/The Black Eyed Peas and Entrepreneur
"Here we are, 2013, we ALL depend on technology to communicate, to bank, and none of us know how to read and write code. It's important for these kids, right now, starting at 8 years old, to read and write code."

Sheryl Sandberg
Chief Operating Officer, Facebook
"An understanding of computer science is becoming increasingly essential in today's world. Our national competitiveness depends upon our ability to educate our children – and that includes our girls – in this critical field."

Vice President Al Gore
"Our civilization is experiencing unprecedented changes across many realms, largely due to the rapid advancement of information technology. The ability to code and understand the power of computing is crucial to success in today's hyper-connected world."

Chris Bosh
NBA All-star, Miami Heat
"Coding is very important when you think about the future, where everything is going. With more phones and tablets and computers being made, and more people having access to every thing and information being shared, I think it's very important to be able to learn the language of coding and programming."

\+ 756'432 others . . .

A pledge for Programming by Celebrities Short version

Why is Programming IMPORTANT for General Education?

- Not for increasing the IT-workforce ! (Though also badly needed)

  Teaching mathematics, physics, chemistry is also not for increasing workforce but for understanding our world.

- Programming is an activity which is
  - creative and
  - constructive work like an engineer!

  and teaches
  - precise working and
  - computational thinking

Example: Shipwrecked Sailors (Quiz in American. Newspaper 1926)

- 5 sailors strand on an island, collect coconuts and want to divide them next day. Go to sleep.
- First sailor wakes up, divides the nuts, one is left for the monkey, hides his part, shuffles the leftover together, goes back to sleep.
- The same repeats with the other sailors.
- Next morning, no one makes a remark, they divide the pile again, and again one nut is left for the monkey.
- How many nuts did they collect?

Solution:

- 1926 solve diophantine equation.
- Today: Simulate! Program the dividing process for nuts $n = 1, 2, 3, \ldots$ until a number is found which fulfills the conditions.

Program: Shipwrecked Sailors

```
function [n,parts]=nuts;
n=0;                                  % initialize number of nuts
good=0;                               % boolean variable
while ~good
  n=n+1;                              % try with next n
  leftover=n;
  good=1;                             % optimistic
  i=0;
  while (i<5) & good                  % try to divide for all sailors
    good=rem(leftover,5)==1;          % good if one nut remains
    if good,
      i=i+1;                          % count sailor
      parts(i)=fix(leftover/5);       % saylor i takes his part
      leftover =parts(i)*4;           % shuffles the leftover together
    end
  end
  good=good & (rem(leftover,5)==1);% next morning:one nut left for monkey
  parts=(leftover-1)/5+parts;         % add morning share to each sailor
end
```

Results

- ```
  >> [n,parts]=nuts
  n = 15621
  parts = 4147   3522   3022   2622   2302
  ```
- for the variant that no nut is leftover for the monkey in the morning we change

  ```
  good=good & (rem(leftover,5)==1); % next morning:one nut left for monkey
  parts=(leftover-1)/5+parts;       % add morning share to each sailor
  ```

  to

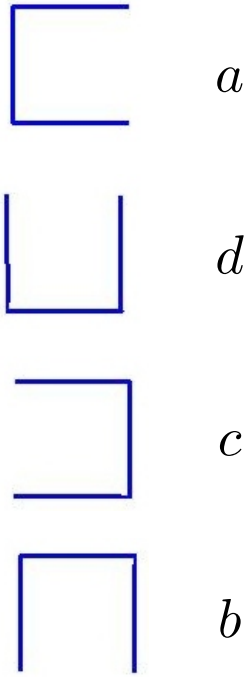
  ```
  good=good & (rem(leftover,5)==0); % next morning: no nut for monkey
  parts=leftover/5+parts;           % add morning share to each sailor
  ```

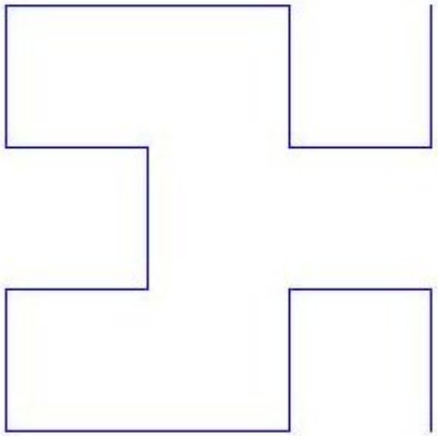
  and get

  ```
  >> [n,parts]=nuts
  n = 3121
  parts = 828   703   603   523   459
  ```
  ```

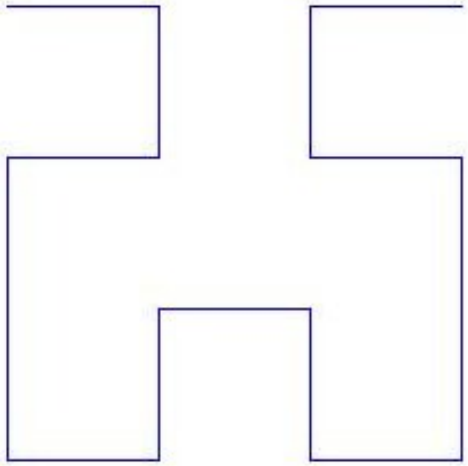
Recursion: Hilbert Curve

(N. Wirth: Algorithms + Data Structures = Programs)





. $a_2 : d \leftarrow a \downarrow a \rightarrow b$



. $d_2 : a \downarrow d \leftarrow d \uparrow c$

Programming the Four Cases

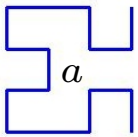
hilbert2

```
function a(i);
global x y h;
if i>0,
    d(i-1); plot([x-h,x],[y,y]); x=x-h;
    a(i-1); plot([x,x],[y-h,y]); y=y-h;
    a(i-1); plot([x,x+h],[y,y]); x=x+h;
    b(i-1);
end
```

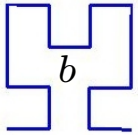
```
function b(i);
global x y h;
if i>0,
    c(i-1); plot([x,x],[y,y+h]); y=y+h;
    b(i-1); plot([x,x+h],[y,y]); x=x+h;
    b(i-1); plot([x,x],[y-h,y]); y=y-h;
    a(i-1);
end
```

```
function c(i);
global x y h;
if i>0,
    b(i-1); plot([x,x+h],[y,y]); x=x+h;
    c(i-1); plot([x,x],[y,y+h]); y=y+h;
    c(i-1); plot([x-h,x],[y,y]); x=x-h;
    d(i-1);
end
```

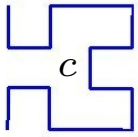
```
function d(i);
global x y h;
if i>0,
    a(i-1); plot([x,x],[y-h,y]); y=y-h;
    d(i-1); plot([x-h,x],[y,y]); x=x-h;
    d(i-1); plot([x,x],[y,y+h]); y=y+h;
    c(i-1);
end
```



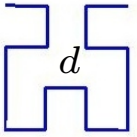
$a_2 : d \leftarrow a \downarrow a \rightarrow b$



$b_2 : c \uparrow b \rightarrow b \downarrow a$



$c_2 : b \rightarrow c \uparrow c \leftarrow d$



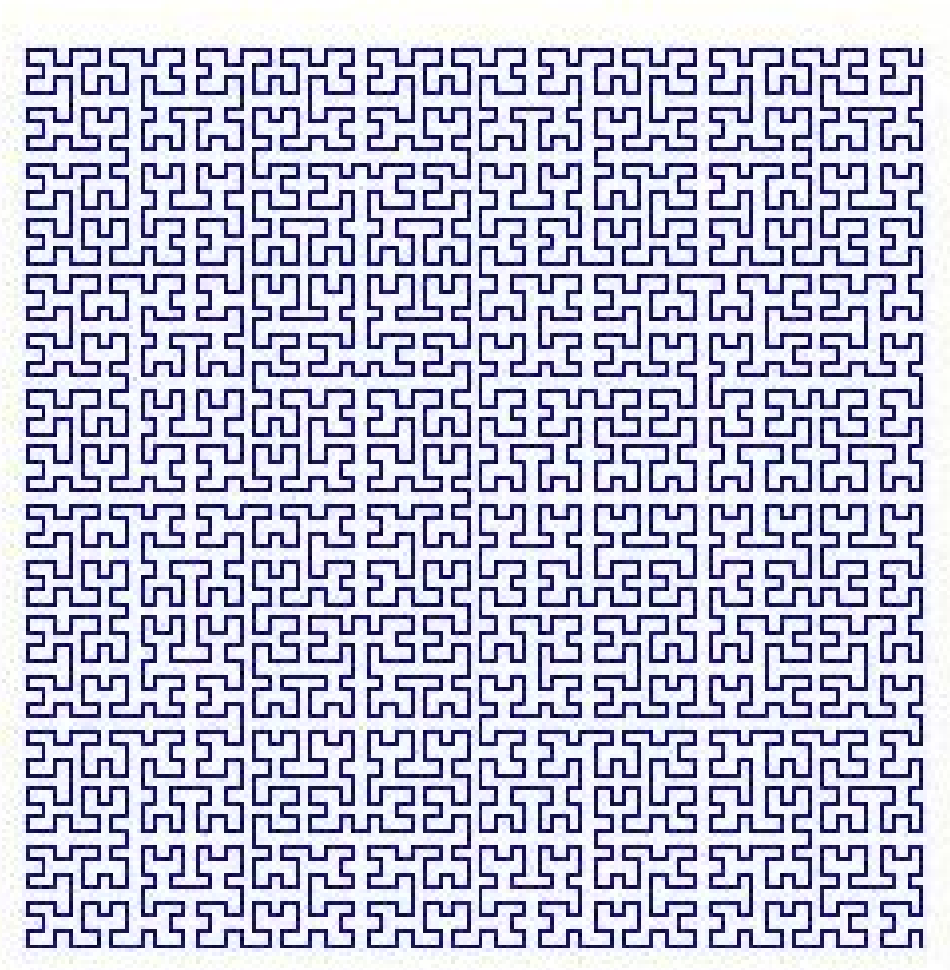
$d_2 : a \downarrow d \leftarrow d \uparrow c$

Hilbert Curve

$a(6)$

hilbert2.m

```
% HILBERT CURVES
%pause off
global x y h;
h0=1024;
for n=1:6
  clf
  axis([-600,800,-600,800])
  axis square, hold
  x=600; y=600 ;
  h=h0/2^n; n
  a(n)
  pause(2)
end
```



Fractal

- $f(z) = z^3 - 1$ has 3 roots:

$$\zeta_1 = 1, \quad \zeta_2 = -\frac{1}{2} + \frac{\sqrt{3}}{2}i, \quad \zeta_3 = -\frac{1}{2} - \frac{\sqrt{3}}{2}i$$

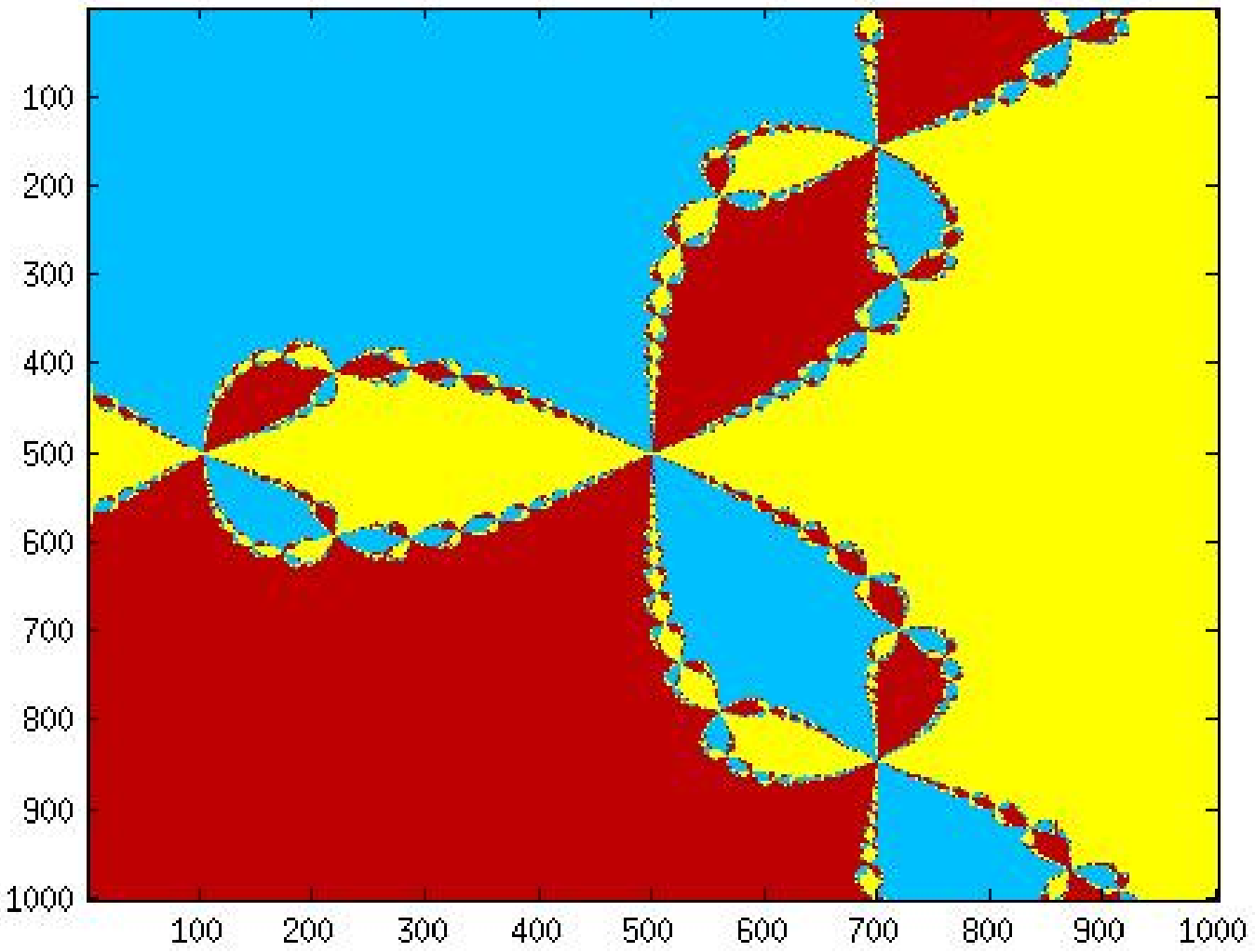
- Apply Newton's Method to compute a root

$$z_{k+1} = z_k - \frac{z_k^3 - 1}{2z_k^2}$$

- Basin of attraction for $\zeta_j = \{z_0 | \lim_{k\to\infty} z_k = \zeta_j\}$
- Compute points of basins for square $z = x + iy$ where $-1 \leq x, y \leq 1$

Compute Basin of Attraction

```
n=1000; m=30;
x=-1:2/n:1;
[X,Y]=meshgrid(x,x);
Z=X+1i*Y;                          % define grid for picture
for i=1:m
  Z=Z-(Z.^3-1)./(3*Z.^2);          % perform m iterations in parallel
end;
a=20;
image((round(imag(Z))+2)*a);       % transform roots to a,2a,3a
```


100
200
300
400
500
600
700
800
900
1000
100
200
300
400
500
600
700
800
900
1000

Informatics as New Basic Subject in Schools?

- Opposition
  - No money. Crises in Europe. High unemployment rates.
  - Expensive: support, maintaining equipment, license fees
  - Policy makers often don't know what programming means. They also often don't know the difference between digital literacy (ICT) and informatics

    Why is ICT necessary? Kids learn the handling of the new devices anyway by themselves.
- Greek solution: eliminate ICT lessons from schools.

  Easy way to save money.

  Solution for poor countries? Resign and give up education in the essential technology for innovations?

Alternative

- Training computational thinking and problem solving does not need expensive equipment .
- Inexpensive computer like Raspberry Pi for $ 25.–:

  `http://www.raspberrypi.org/`
- Public domain software is free of charge:
  - Linux operating system
  - LibreOffice (successor of OpenOffice)

    `http://www.libreoffice.org/default/`
  - LaTeX for professional typesetting
  - Logo, Pascal, Octave, . . . programming languages

Google engineer Neil Fraser
(March 2013, visiting Vietnamese high school kids) [a]

- *Problem: "Given a data file describing a maze with diagonal walls, count the number of enclosed areas, and measure the size of the largest one."*
- *The class had 45 minutes to design a solution and implement it in Pascal. Most of them finished, a few just needed another five minutes. There is no question that half of the students in that grade 11 class could pass the Google interview process.*
- *If nothing else, this snapshot into the Vietnamese school system shows what can be done despite limited funds.*

[a] `http://www.theregister.co.uk/2013/03/22/vietnam_kids_google_interview_pass/`

ETH Teaches Programming to Primary School Students

- Very successful pilot project of JURAJ HROMKOVIC

Oberkulm: Start zum dreijährigen Pilotprojekt Programmieren an der Primarschule Oberkulm

Programmieren – Ein Kinderspiel?

Die Primarschule Oberkulm führt als erste Pilotschule im Kanton Aargau ein dreijähriges Informatikprojekt zum Thema Programmieren durch. Die Klassen im vierten und fünften Schuljahr, bei den Klassenlehrpersonen Beate Welsche und Ruth Trüb, starteten als erste ins Pilotprojekt.

moha. Das Projekt steht unter der Leitung von ETH-Professor Juraj Hromkovic, und einem Team von Studenten und Doktoranden, welche die Schüler der beiden Mittelstufenklassen instruieren. Unterstützt wird das Vorhaben von der Hasler Stiftung in Bern. Auf die Altersstufe der Schüler ausgerichtet wurden spezielle Lehrmittel ausgearbeitet. Die Beratungsstelle imedias an der Pädagogischen Hochschule FHNW begleitet das Projekt, beobachtet dessen Fortgang und wertet es aus.

Das Pilotprojekt in Oberkulm star-

Keine Zeit für Pause: *Nachwuchs-Informatiker tüfteln gemeinsam an Lösungen.*

vier Studenten begleitet, welche abwechslungsweise die Assistenz übernahmen.

Intensiv und herausfordernd

sie kaum Zeit fanden, eine Pause zu machen. Für das Pilotprojekt wurden der Schule Oberkulm die nötige Anzahl Laptops zur Verfügung gestellt, oder die Kinder durften ihre eigenen

Children are fascinated

Logo Script also Spanish !

- Free download of Logo Script

`http://abz.inf.ethz.ch/primarschulen-unterrichtmaterialien`