BERLIN 2012
CONFERENCE

17th-19th October

LibreLogo

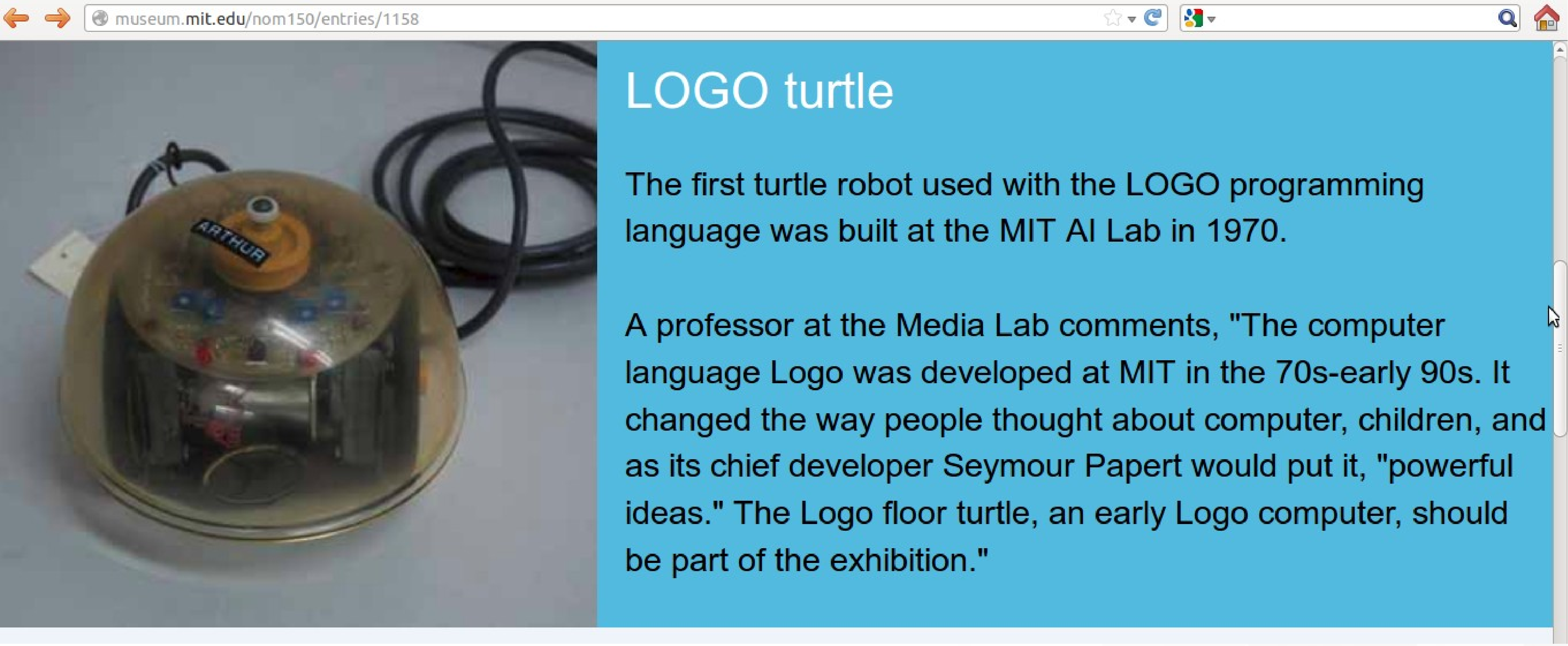
Turtle Vector Graphics for Everybody

- László Németh «nemeth@numbertext.org»
- 2012-10-17

LibreOffice

Why turtle graphics?

- “Turtles”: drawing robots

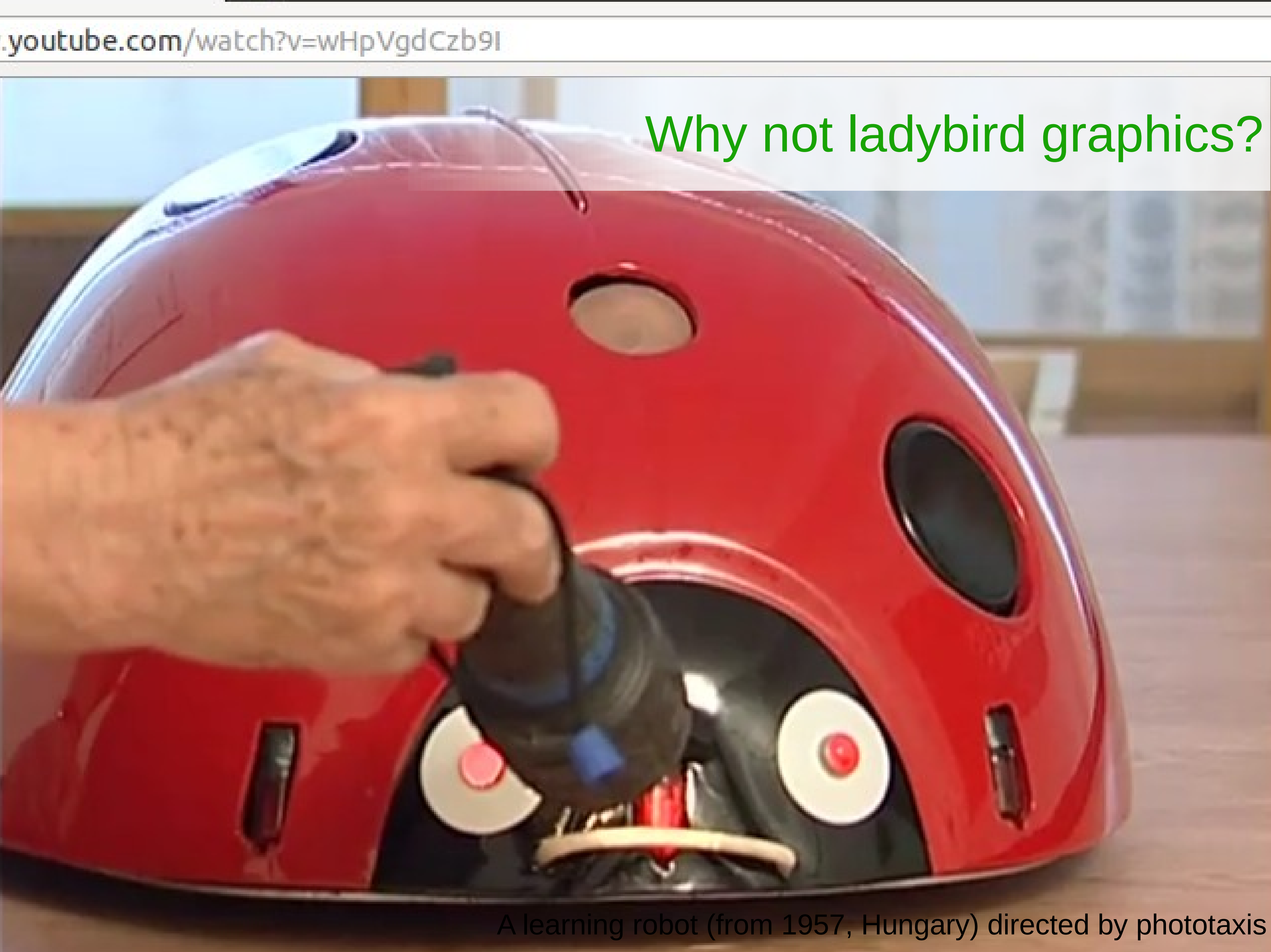


LOGO turtle

The first turtle robot used with the LOGO programming language was built at the MIT AI Lab in 1970.

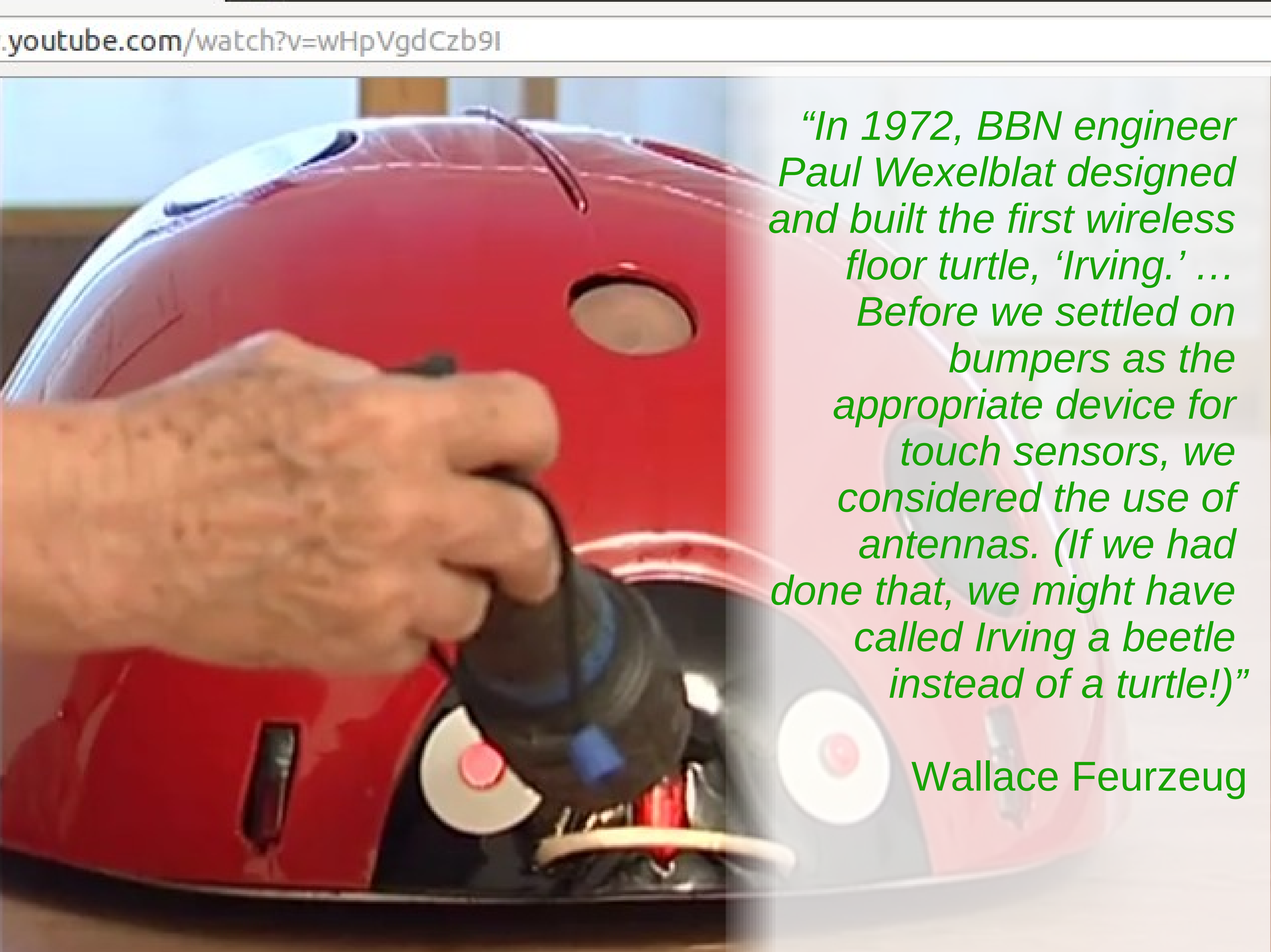
A professor at the Media Lab comments, "The computer language Logo was developed at MIT in the 70s-early 90s. It changed the way people thought about computer, children, and as its chief developer Seymour Papert would put it, "powerful ideas." The Logo floor turtle, an early Logo computer, should be part of the exhibition."

Why not ladybird graphics?



.youtube.com/watch?v=wHpVgdCzb9I

A learning robot (from 1957, Hungary) directed by phototaxis



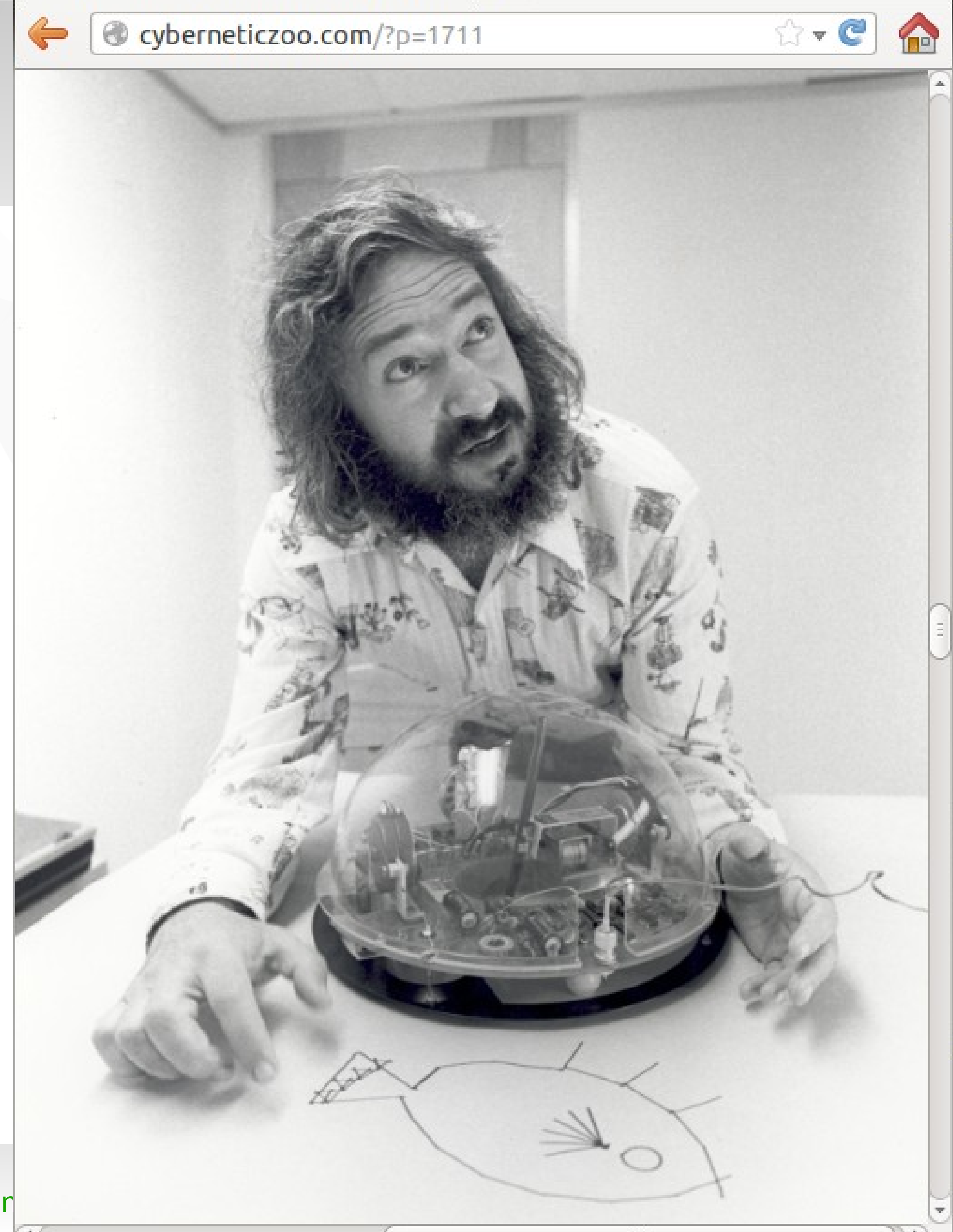
.youtube.com/watch?v=wHpVgdCzb9I

“In 1972, BBN engineer Paul Wexelblat designed and built the first wireless floor turtle, ‘Irving.’ … Before we settled on bumpers as the appropriate device for touch sensors, we considered the use of antennas. (If we had done that, we might have called Irving a beetle instead of a turtle!)”

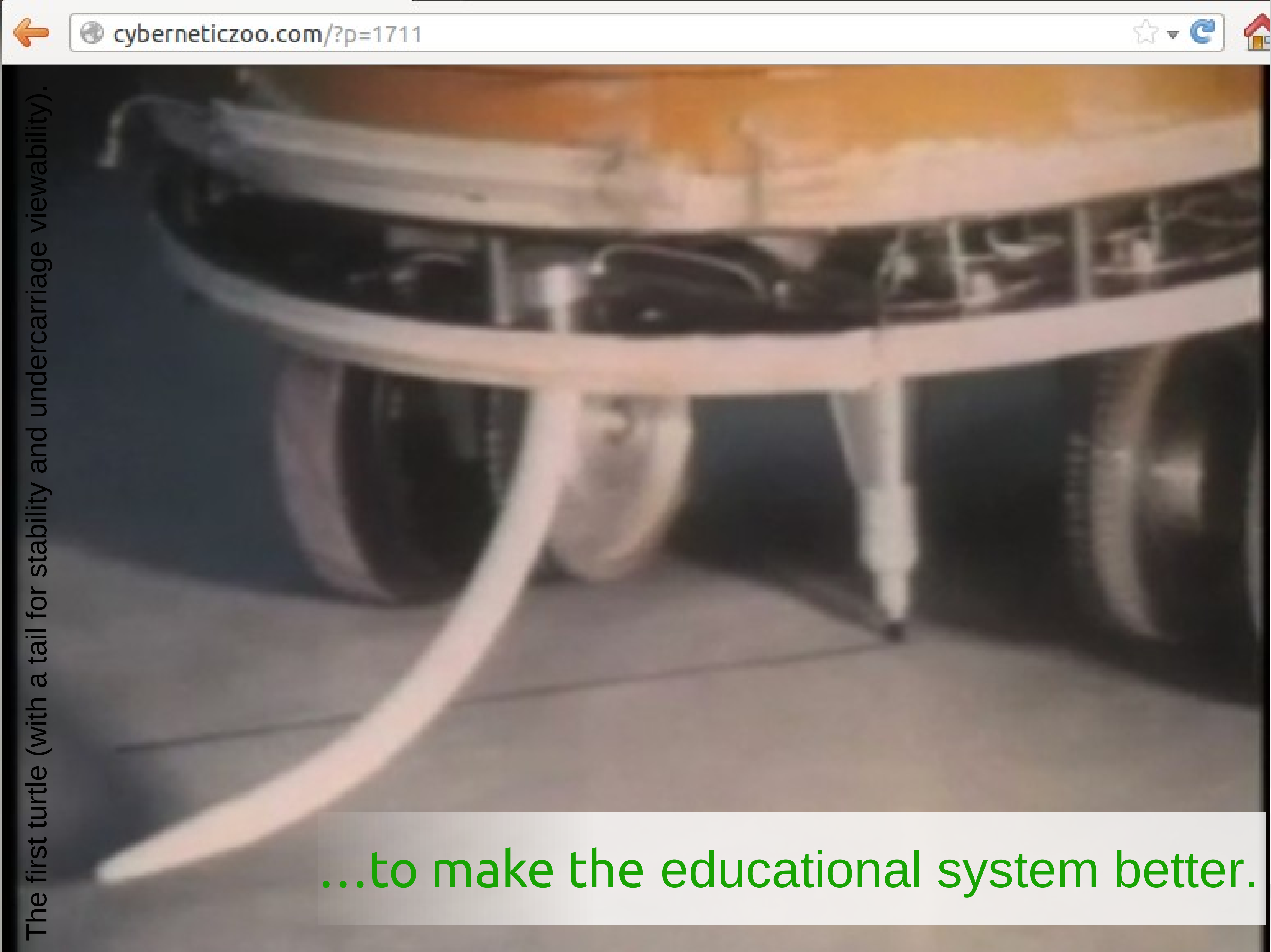
Wallace Feurzeug

Seymour Papert

“The Turtle is seen as a metaphor, an ‘object-to-think-with’”



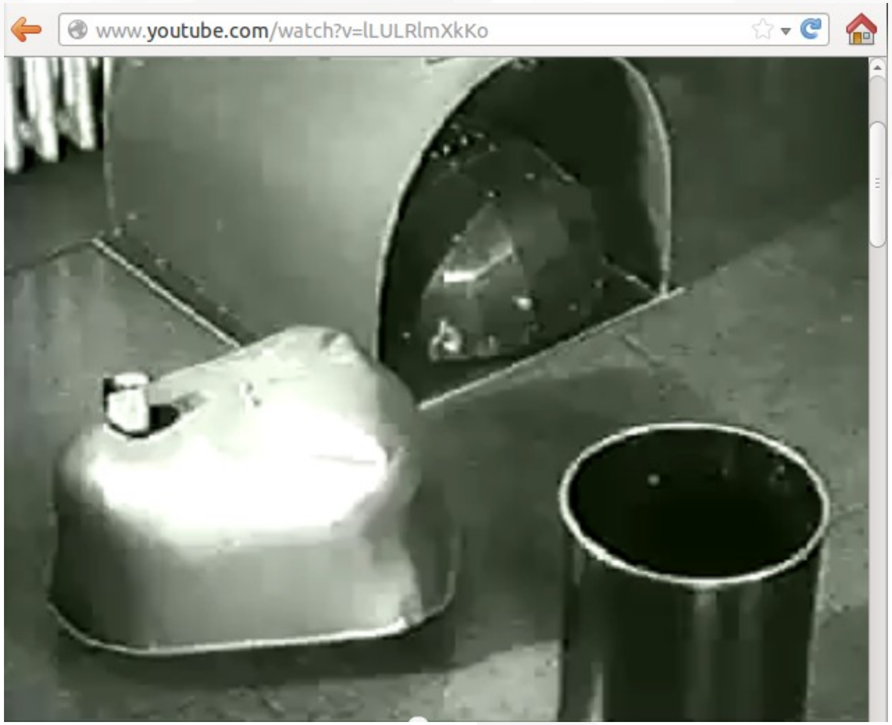
…in experiments on children



The first turtle (with a tail for stability and undercarriage viewability).

…to make the educational system better.

Grey Walter’s “tortoises”



Grey Walter’s first robots, which he used to call Machina speculatrix *and named Elmer and Elsie, were constructed between 1948 and 1949 and were often described as tortoises due to their shape and slow rate of movement - and because they “taught us” about the secrets of organization and life.* (Wikipedia)

Logo

- 1968, BBN, Logo class for 7th grades by Seymour Papert (former collegaue and protégé of Jean Piaget) and Cynthia Solomon
- 1970, MIT AI Lab, “Yellow turtle”, Logo class for 5th grades

  http://logothings.wikispaces.com

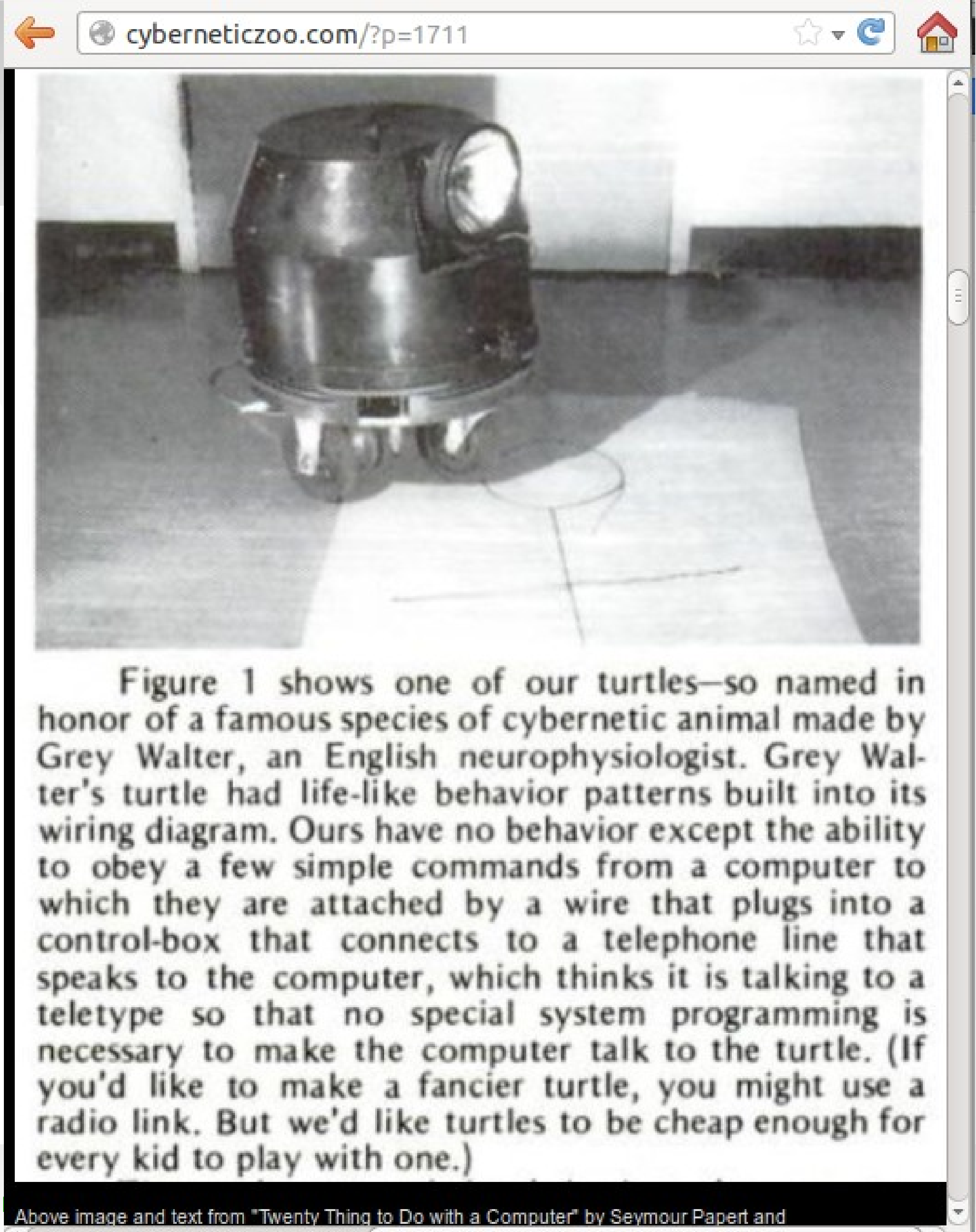
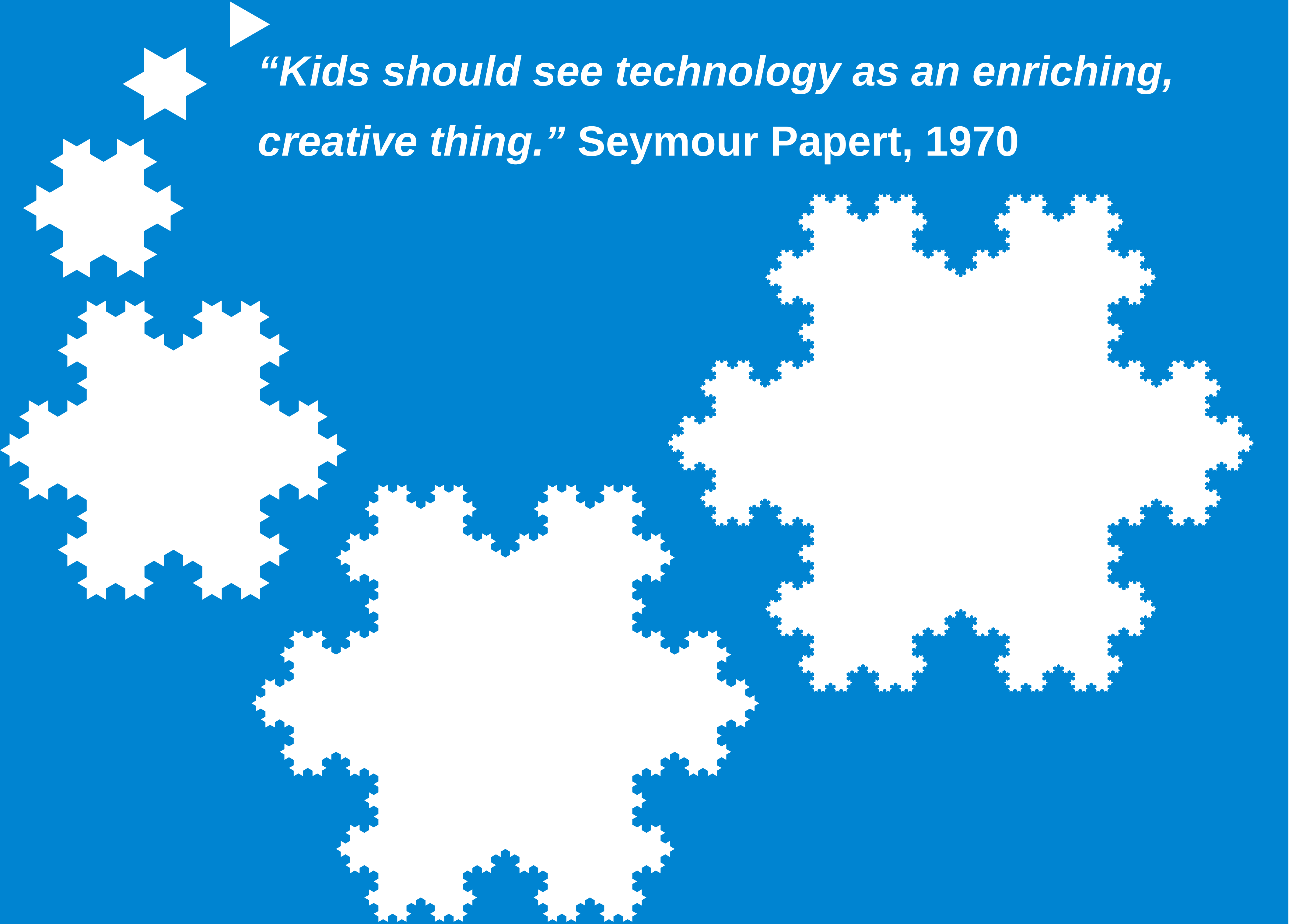


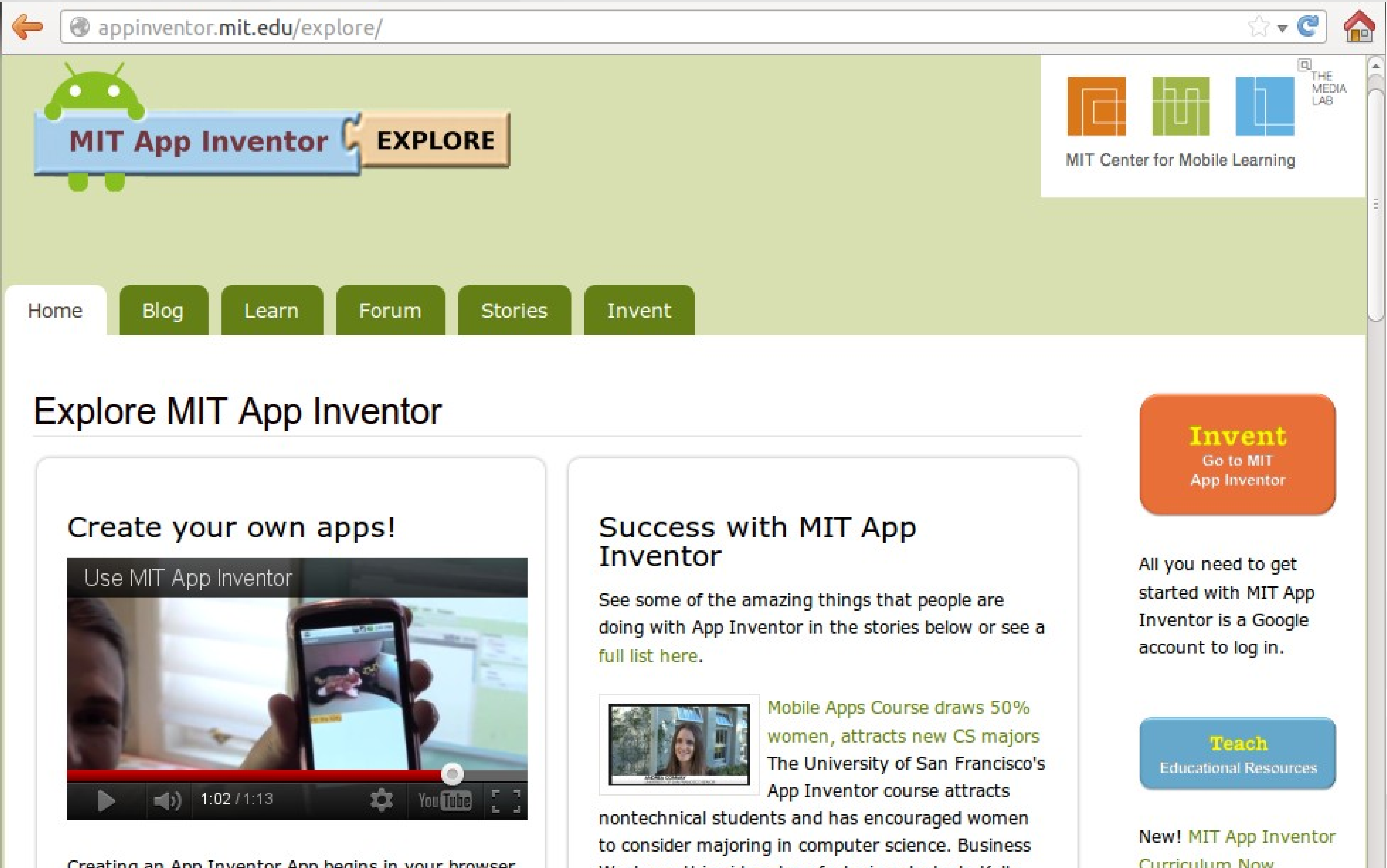
Figure 1 shows one of our turtles—so named in honor of a famous species of cybernetic animal made by Grey Walter, an English neurophysiologist. Grey Walter's turtle had life-like behavior patterns built into its wiring diagram. Ours have no behavior except the ability to obey a few simple commands from a computer to which they are attached by a wire that plugs into a control-box that connects to a telephone line that speaks to the computer, which thinks it is talking to a teletype so that no special system programming is necessary to make the computer talk to the turtle. (If you'd like to make a fancier turtle, you might use a radio link. But we'd like turtles to be cheap enough for every kid to play with one.)

Above image and text from "Twenty Thing to Do with a Computer" by Seymour Papert and

“Kids should see technology as an enriching, creative thing.” **Seymour Papert, 1970**



Like mobile applications…



appinventor.mit.edu/explore/

Explore MIT App Inventor

Create your own apps!

Creating an App Inventor App begins in your browser

Success with MIT App Inventor

See some of the amazing things that people are doing with App Inventor in the stories below or see a full list here.

Mobile Apps Course draws 50% women, attracts new CS majors
The University of San Francisco's App Inventor course attracts nontechnical students and has encouraged women to consider majoring in computer science. Business

All you need to get started with MIT App Inventor is a Google account to log in.

New! MIT App Inventor Curriculum Now

And what about the kids?

Will every 5th grades be interested in Android programming?

And what about office suites?

Can word processing be fun in schools?

Revival of command line?

- Python console, eg. in the Blender 3D modeling creation suite
- Windows PowerShell
- Spring Roo for Java EE development
- Unity Dash on Ubuntu
- Firefox Command Line for developers
- Formula Bar in LibreOffice Writer (press F2)

Theory of Logo and turtle graphics

- Constructionism
  - Object-to-think-with
  - direct feedback (syntax error, bad turtle positions)

Practice of Logo and turtle graphics

- Simple syntax for children
  - Native and understandable words
  - Only letters, without parenthesization
- Fun with drawing robots

Logo in Hungarian schools

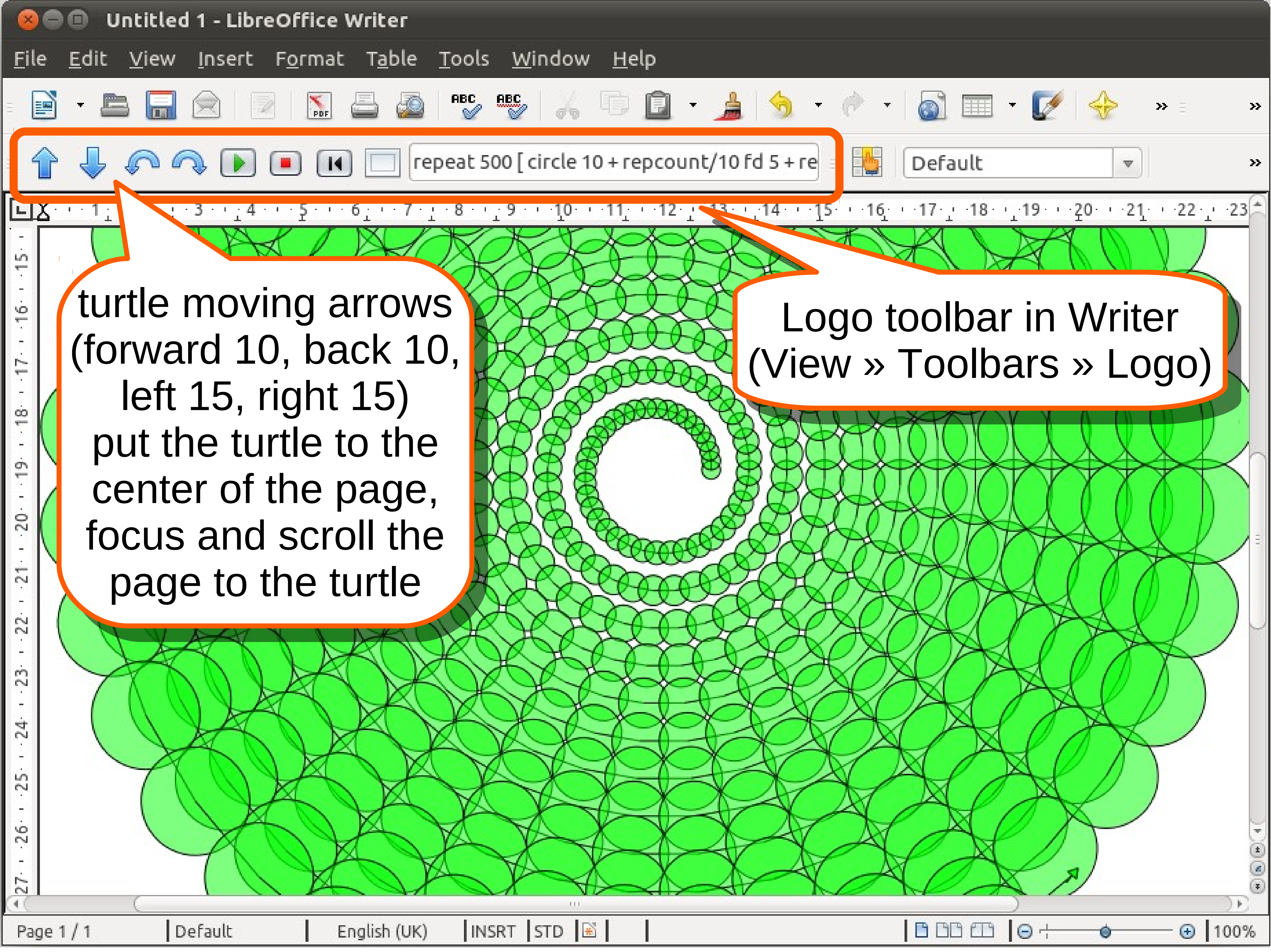
- Logo programming
  - teaching of computing
  - teacher education
- Closed source (Comenius Logo, Imagine Logo)
- Platform-dependent (Windows)
- Native (“elő́re” = “forward”)

LibreLogo – an answer for the problems

- Free/open source
- Portable (also HTML 5 was tested before)
- LibreOffice extension
  - http://extensions.libreoffice.org/extension-center/librelogo/

What is LibreLogo?

- A programming environment for children and graphic design
  - Back compatible with older educational Logo environments
  - with full localization (set the language of the document)
  - Vector graphics for printing quality
  - Drawing objects: combine programming and image editing

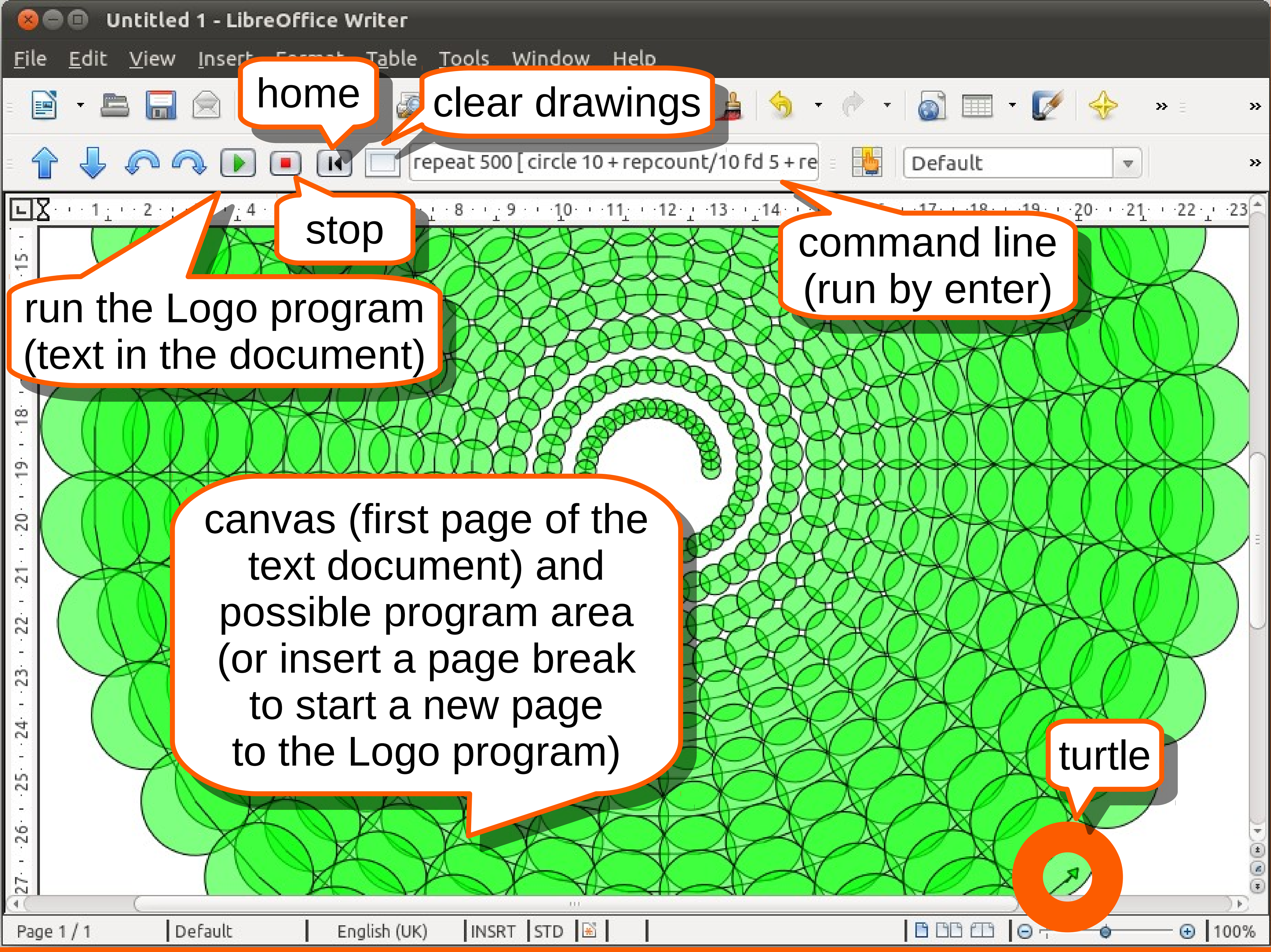


Untitled 1 - LibreOffice Writer
File Edit View Insert Format Table Tools Window Help
Default
Page 1 / 1
Default
English (UK)
INSRT
STD
100%

```
repeat 500 [ circle 10 + repcount/10 fd 5 + re
```

turtle moving arrows (forward 10, back 10, left 15, right 15) put the turtle to the center of the page, focus and scroll the page to the turtle

Logo toolbar in Writer (View » Toolbars » Logo)


Untitled 1 - LibreOffice Writer
File Edit View Insert Format Table Tools Window Help
Default
Page 1 / 1
Default
English (UK)
INSRT
STD
100%

```
repeat 500 [ circle 10 + repcount/10 fd 5 + re
```

home

clear drawings

stop

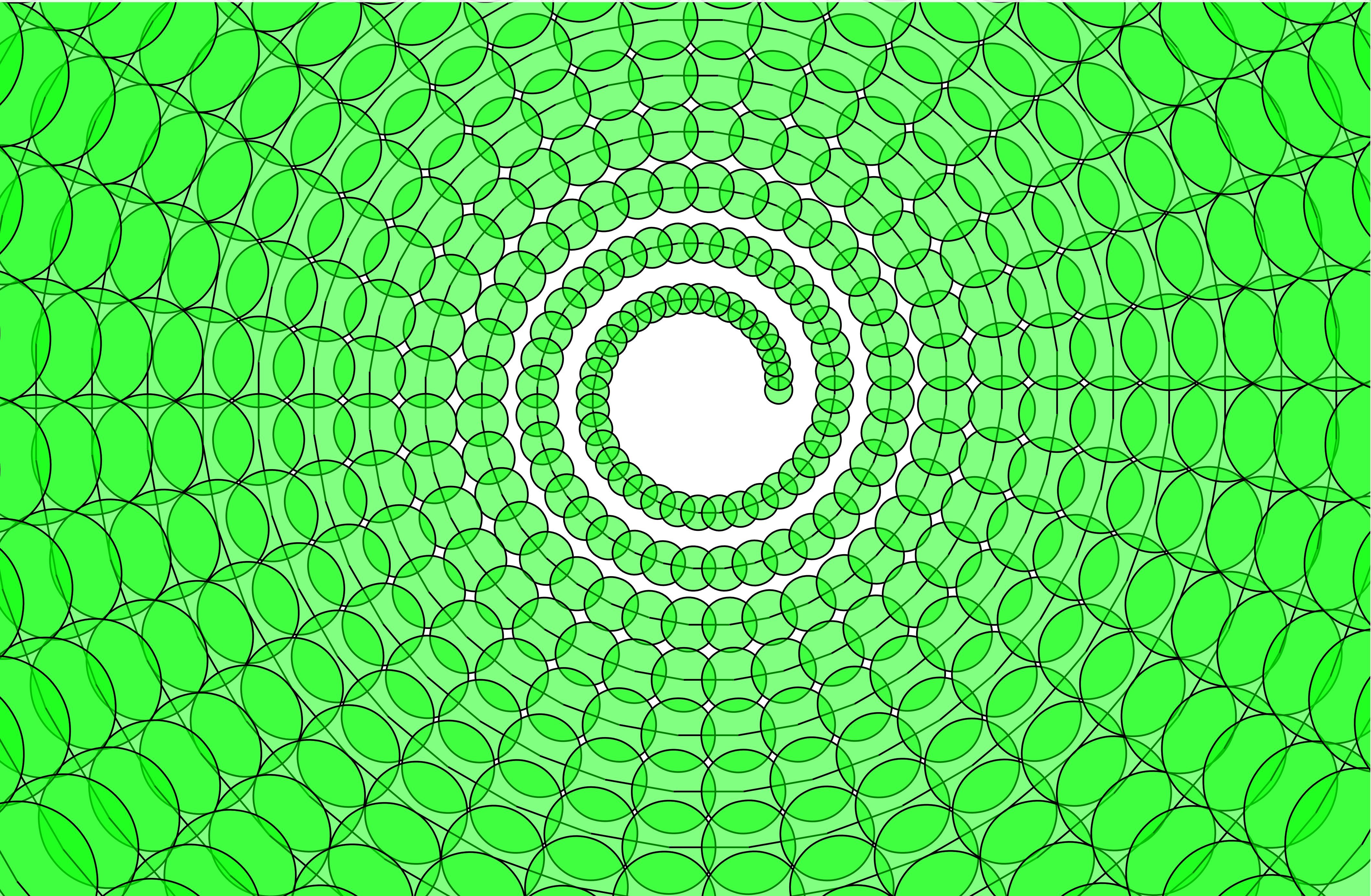
run the Logo program (text in the document)

command line (run by enter)

canvas (first page of the text document) and possible program area (or insert a page break to start a new page to the Logo program)

turtle

```
repeat 500 [ circle 10 + repcount/10 fd 5 + repcount/10 lt 10 ]
```



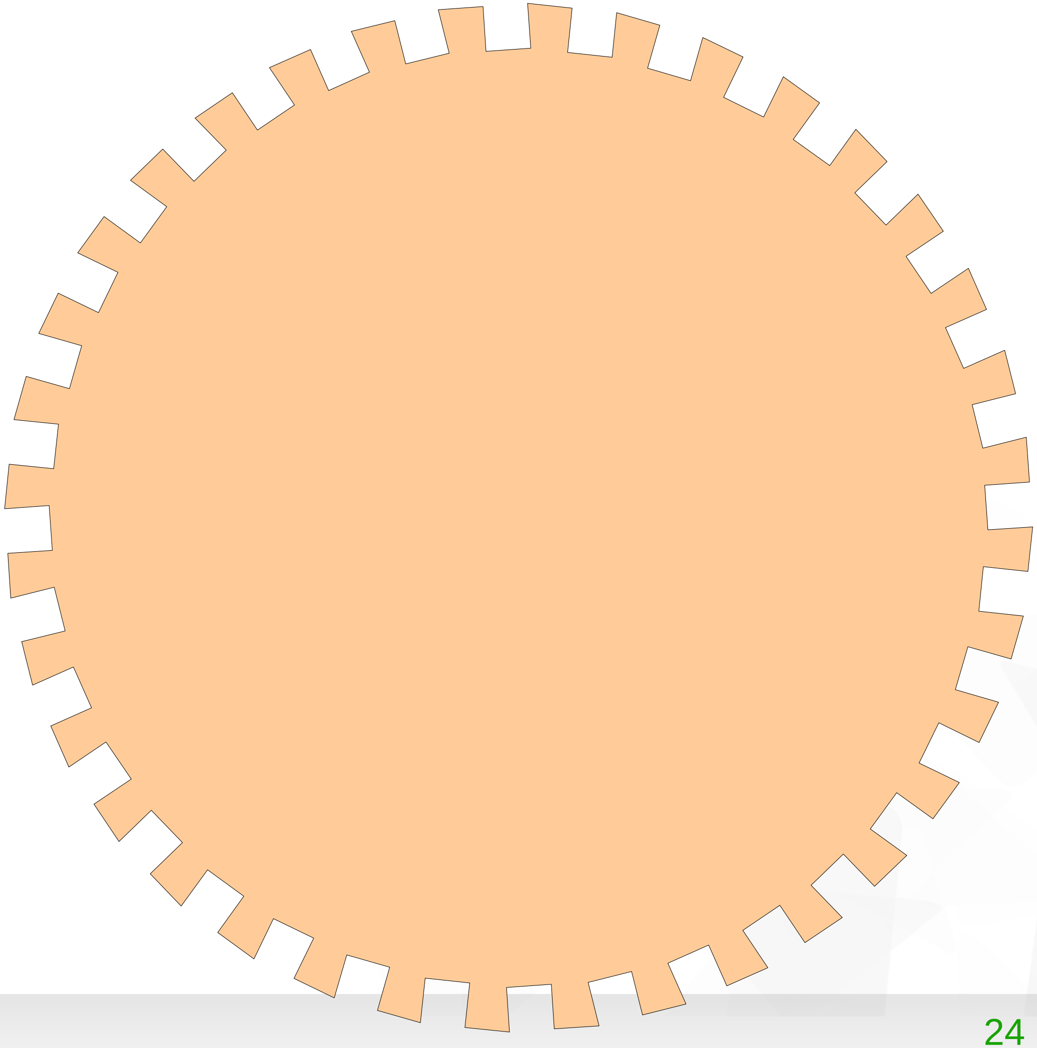
Advantages of LibreOffice

- Free and portable
- Vector graphics
- Interactive shapes or shape groups
- Direct use of the Logo drawing objects (print, or paste into documents only via clipboard)
- Printing quality: PDF or (via Draw) SVG export
- Anti-aliasing
- Zoom
- Automatic scroll (turtle tracing)
- Alpha transparency
- Standard file format (OpenDocument)
- Unicode support, TrueType/Graphite font technology

Vector graphics

- One of the first shapes created by the prototype:

```
repeat 36 [
   forward 5 left 90
   forward 5 right 90
   forward 5 right 90
   forward 5 left 100
]

fill
```



Complex shapes

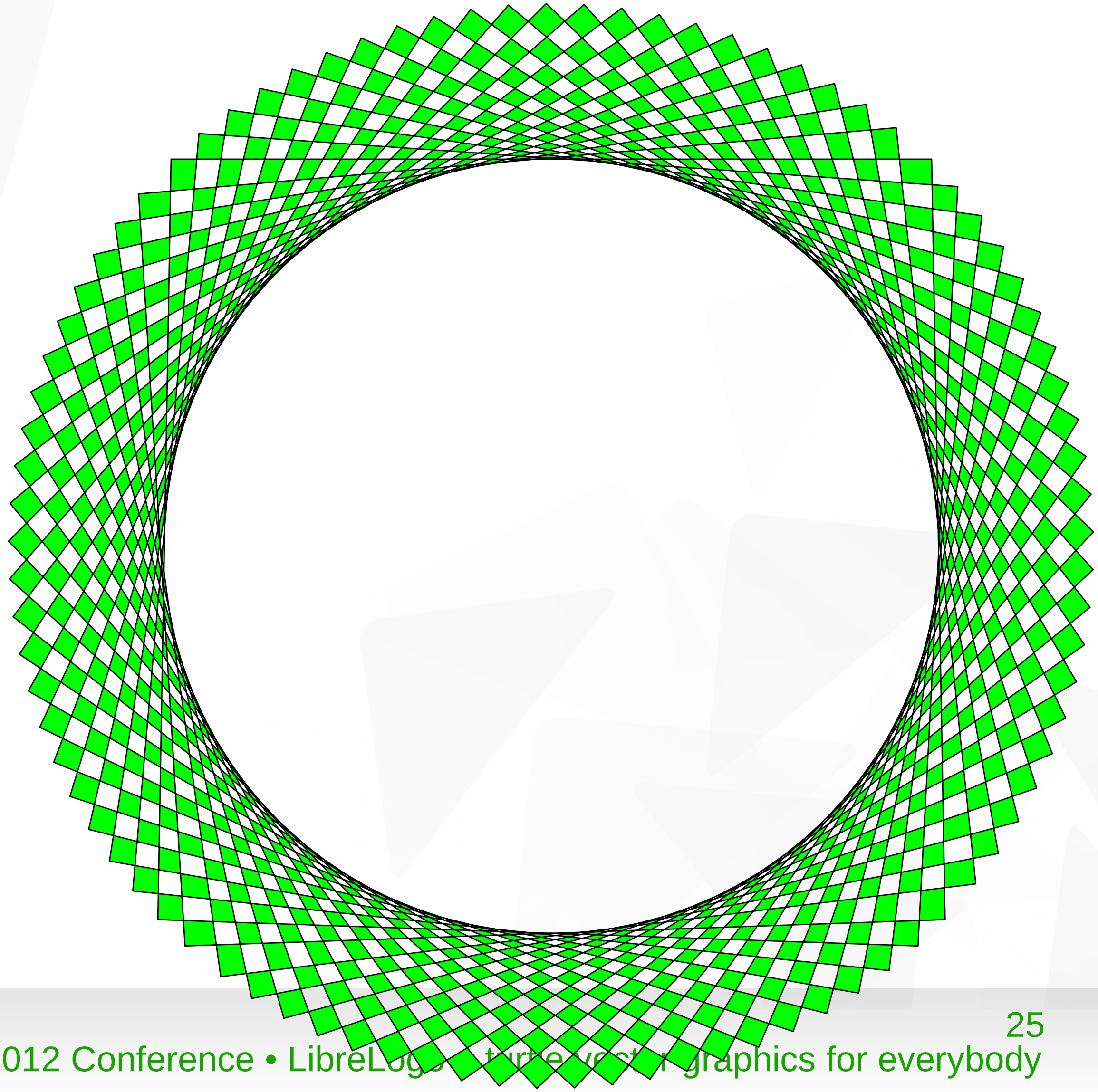
- Shapes with intersecting sides have got complex filling

```
repeat 88 [
   forward 200
   left 89
]

fill
```

- (Or without loop, press enters on the following command line:

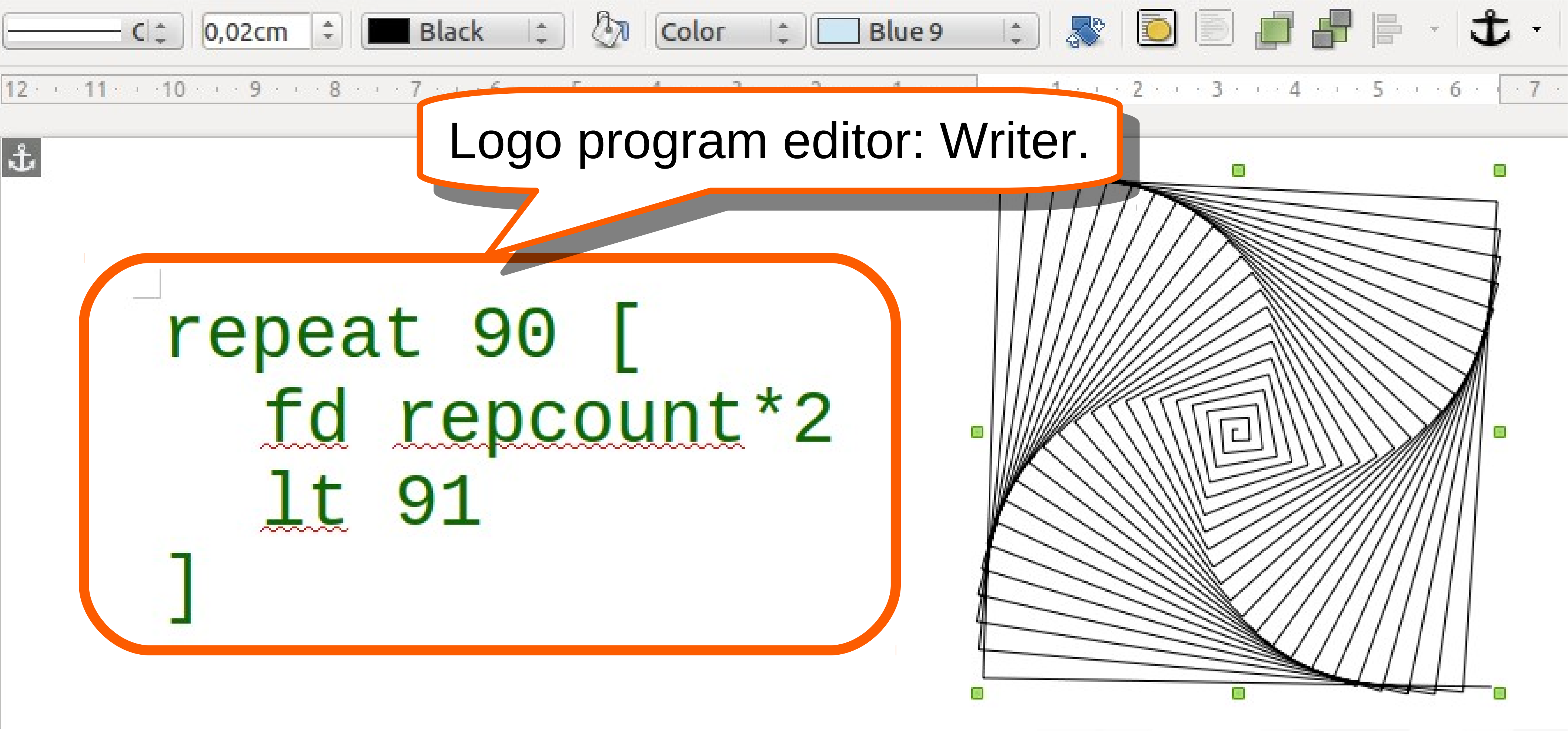
```
fd 200 lt 89
```

and in the end, “fill” it.)



Teaching of computing: word processing

- Simple IDE: error message and the cursor jumps to the faulty line.

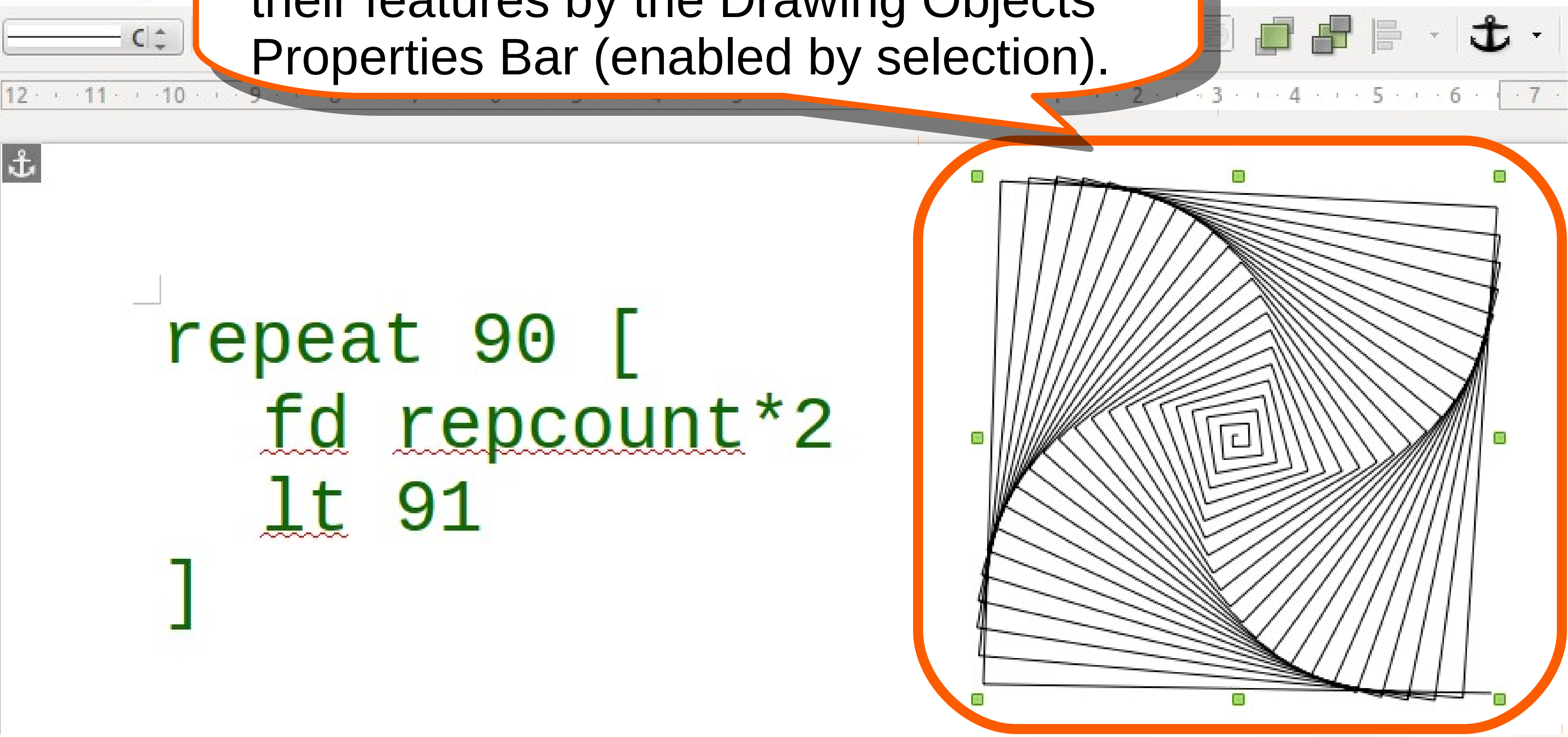


Logo program editor: Writer.

```
repeat 90 [
   fd repcount*2
   lt 91
]
```

Teaching of computing: image handling

LibreLogo drawings: shapes or objects. Select, move, resize (hold Shift to keep ratio), add text (double click), modify their features by the Drawing Objects Properties Bar (enabled by selection).

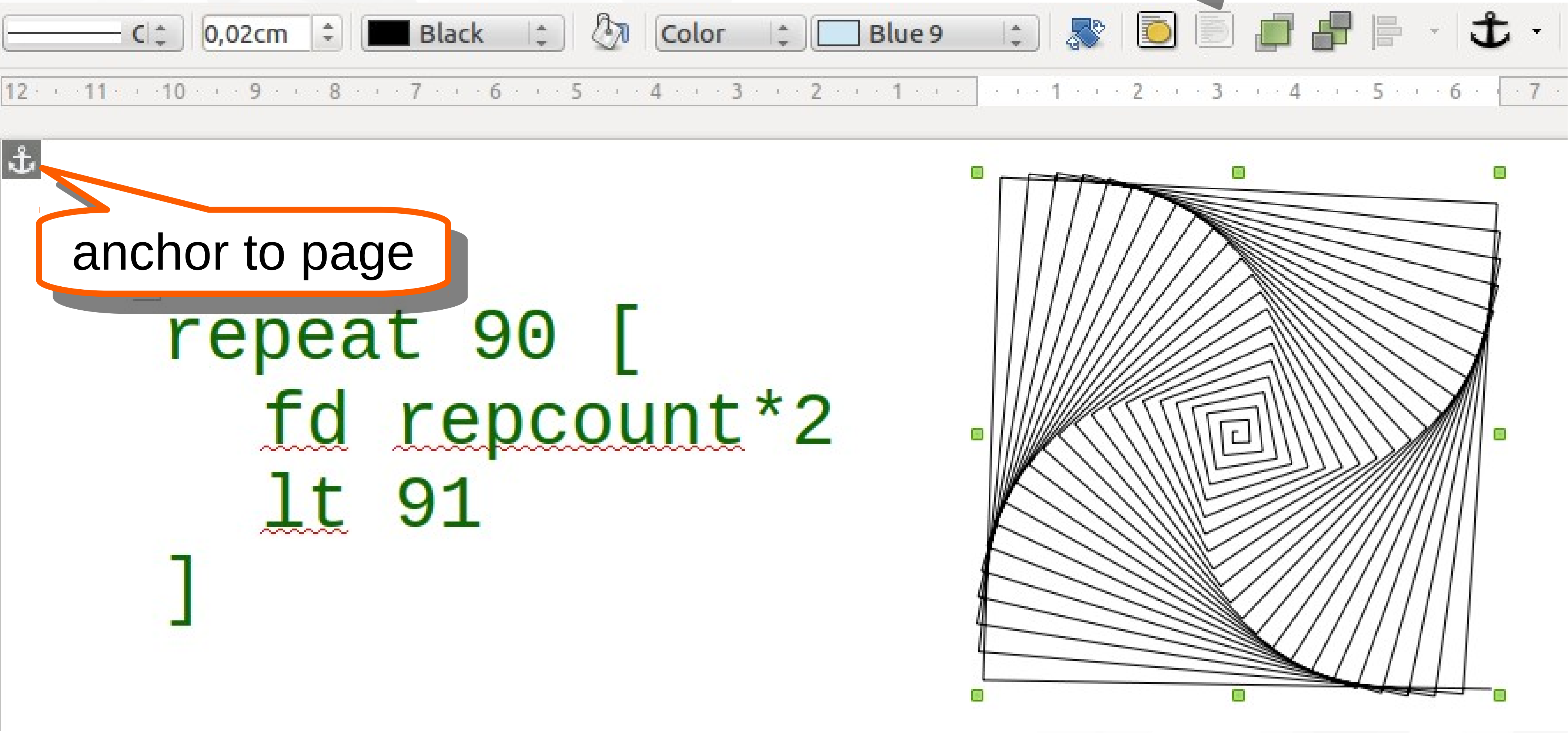


```
repeat 90 [
   fd repcount*2
   lt 91
]
```

Teaching of computing: image handling

- default settings of the shapes

wrap in background



anchor to page

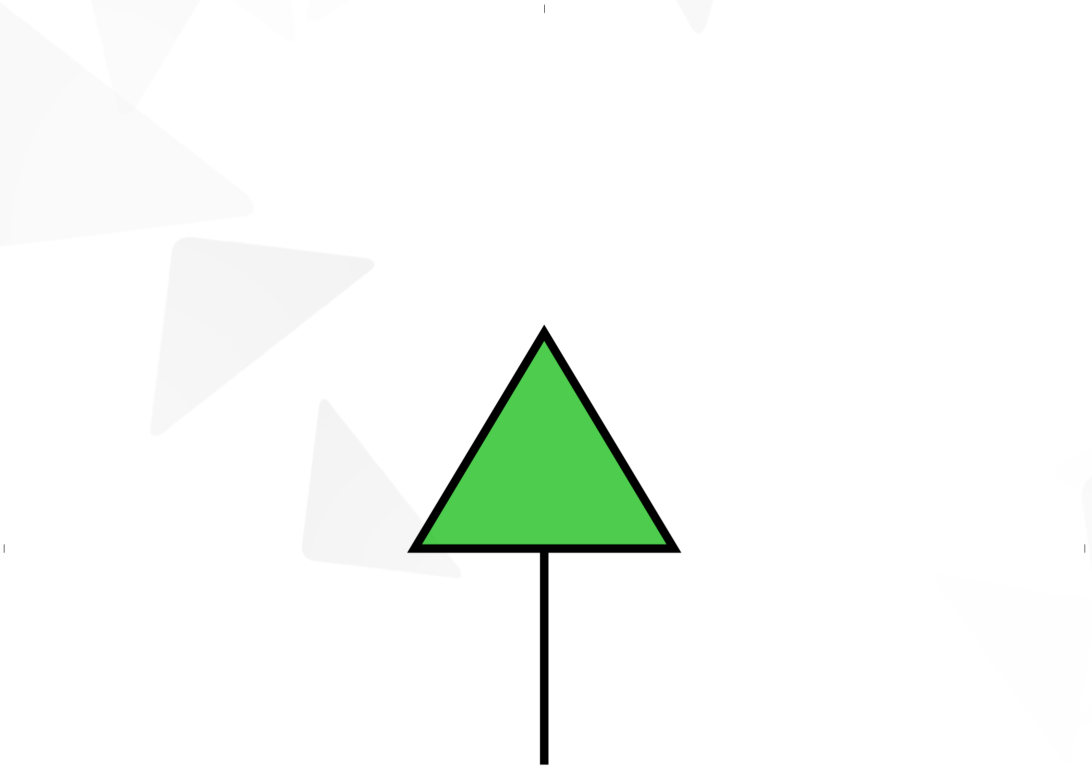
```
repeat 90 [
  fd repcount*2
  lt 91
]
```

Teaching of computing: Python programming

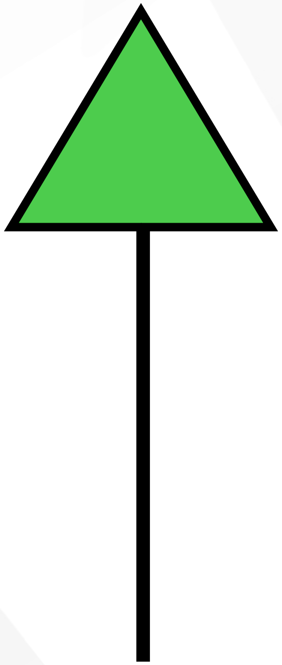
- LibreLogo programs converted to Python and running in PyUNO environment by the embedded Python of LibreOffice
- LibreLogo supports Python
  - for+in cycle (localized versions)
  - string operations
  - regular expressions
- and advanced Python data structures
  - lists
  - tuples
  - dictionaries
  - sets
- open Python source of LibreLogo

Turtle in LibreLogo

- Arrow-like shape
- Moveable and rotatable like the standard drawing objects
- Its line width, style and colors show the turtle settings

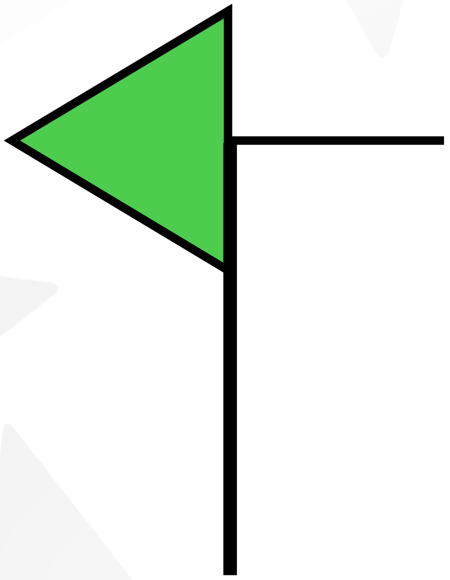


Go forward by the given typographical points

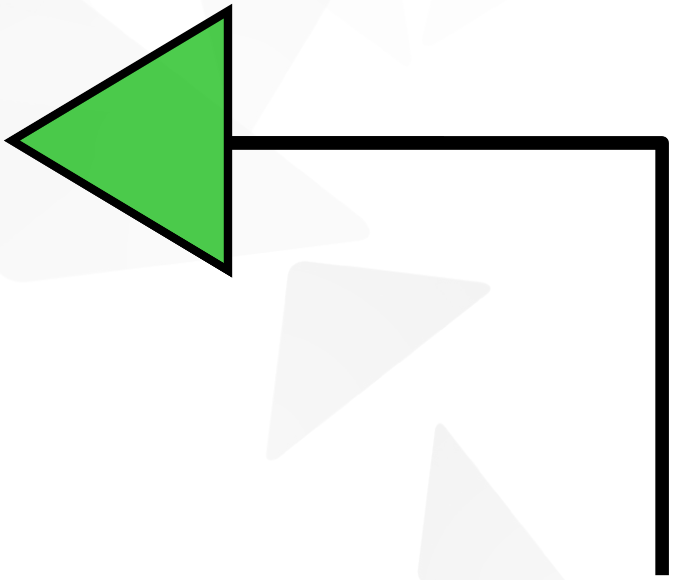


```
forward 10
```

Turn left (counterclockwise) the given degrees

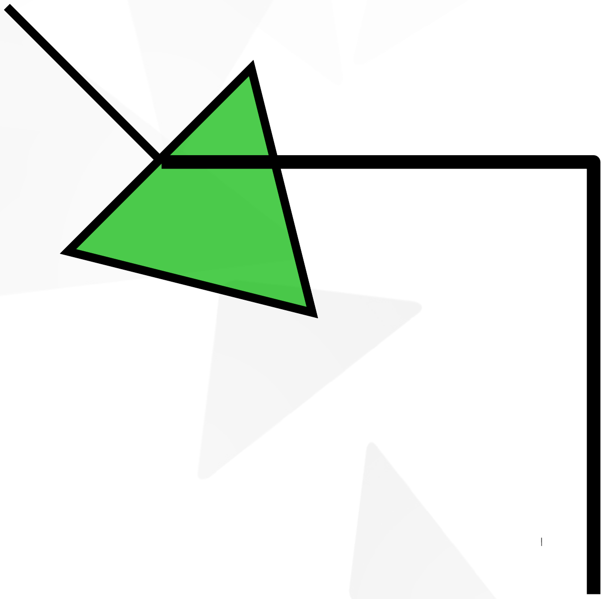


```
forward 10
left 90
```



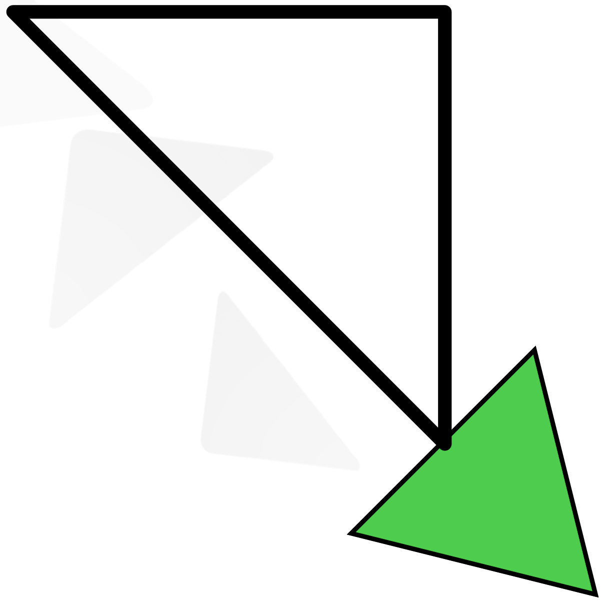
```
forward 10
left 90
forward 10
```

Expressions as function parameters



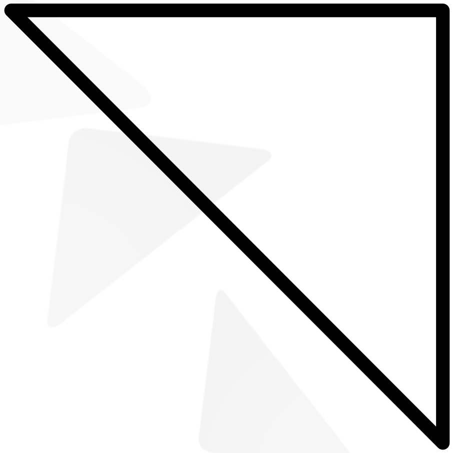
```
forward 10
left 90
forward 10
left 90+45
```

Expression with a square root function (sqrt)



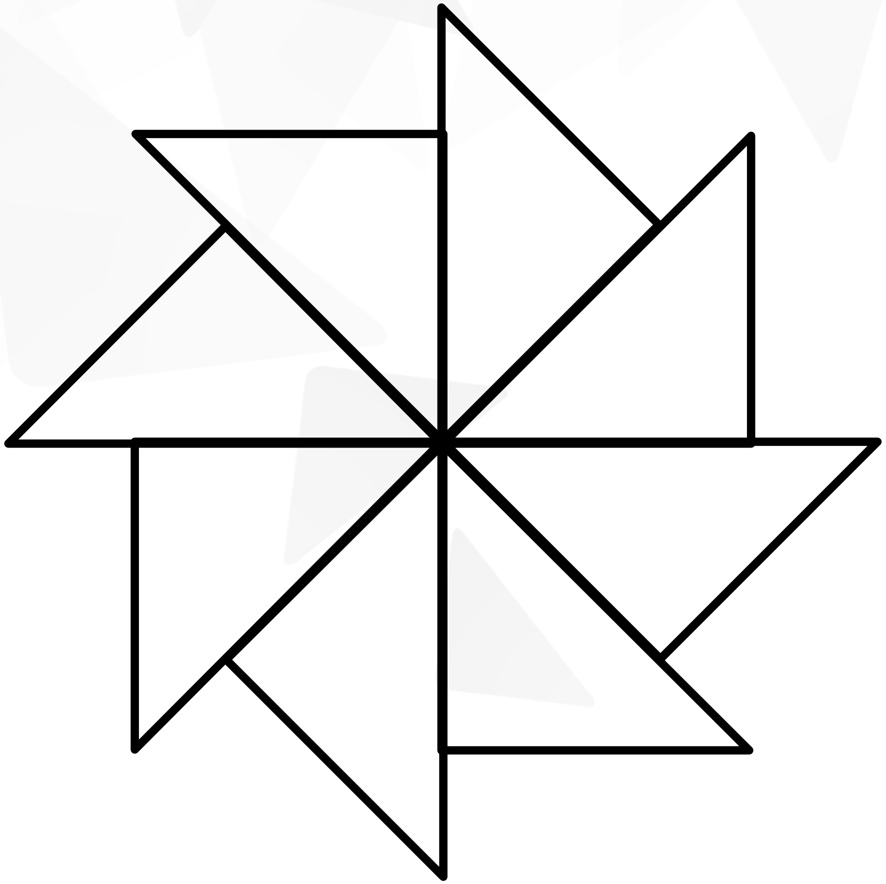
```
forward 10
left 90
forward 10
left 90+45
forward sqrt 2*10*10
```

Hide the turtle (until the command showturtle)



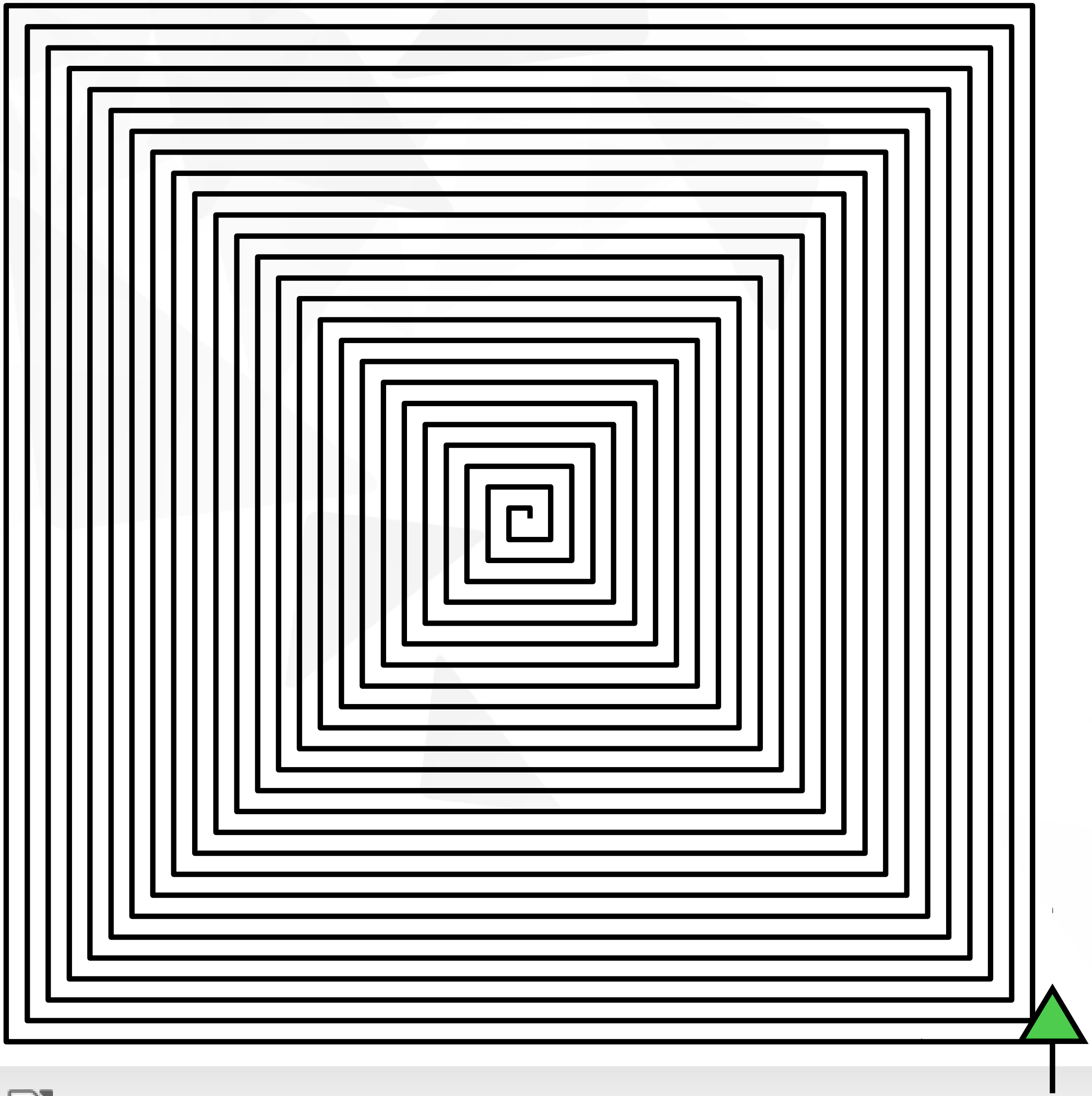
```
forward 10
left 90
forward 10
left 90+45
forward sqrt 2*10*10
hideturtle
```

Loop – repeat n times [commands]



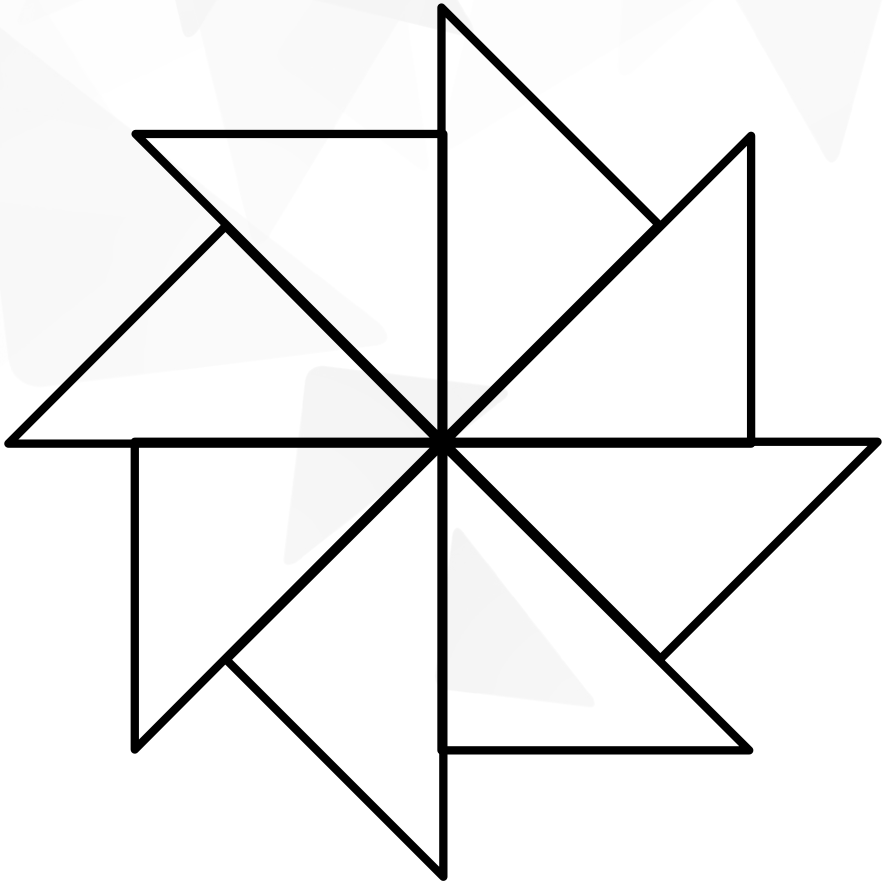
```
repeat 8 [
   forward 10
   left 90
   forward 10
   left 90+45
   forward sqrt 2*10*10
   hideturtle
]
```

Repcount – default loop variable from 1 to n



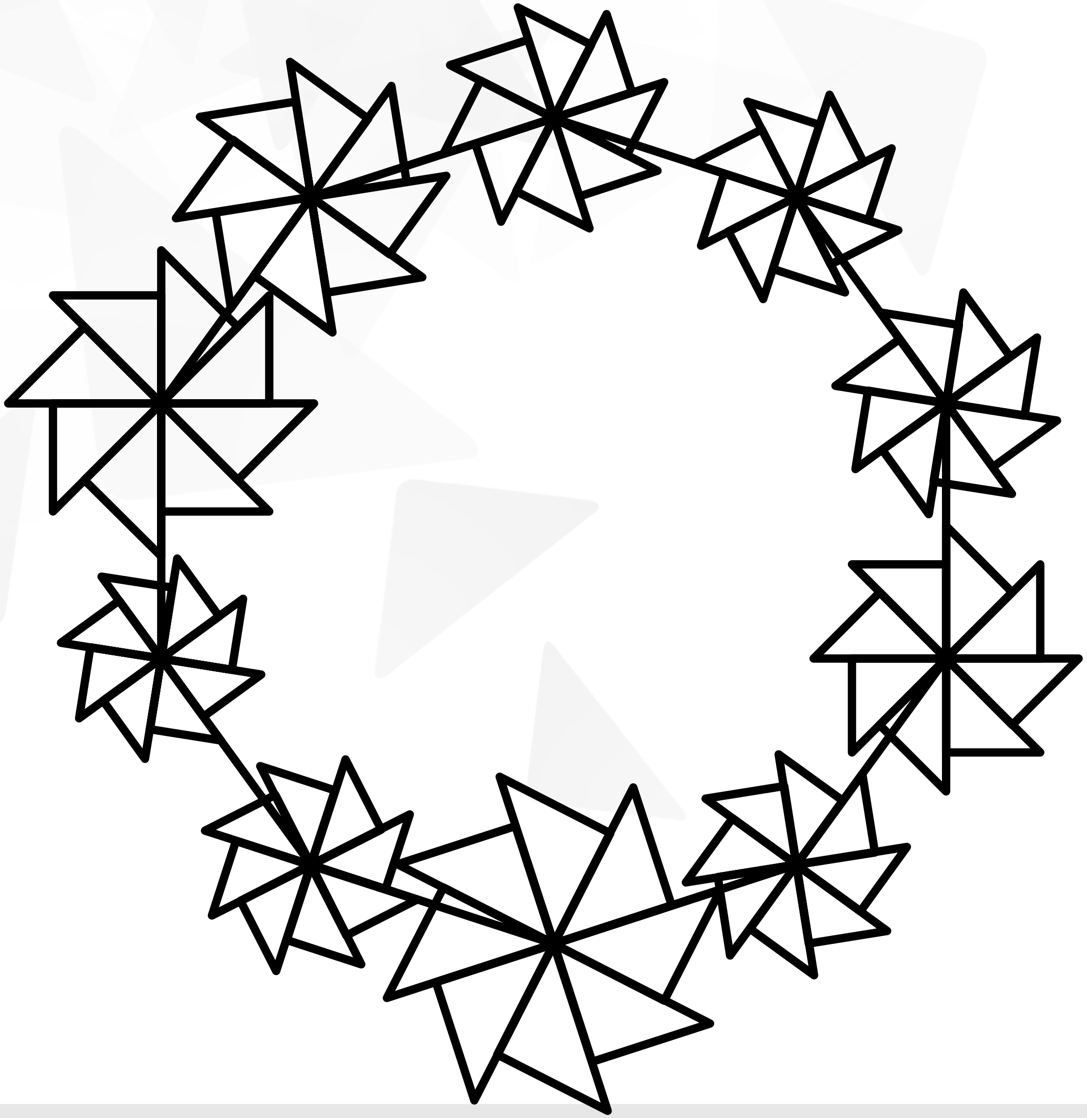
```
repeat 100 [
   fd repcount
   lt 90
]
```

Comments and abbreviations



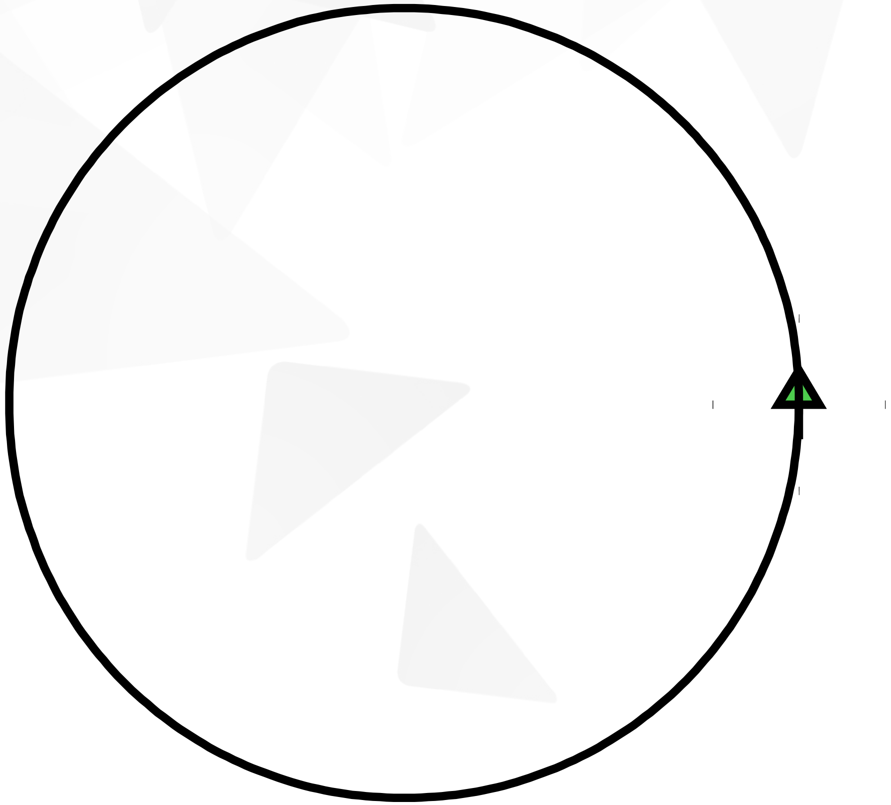
```
; some commands have
; abbreviated forms

repeat 8 [
   fd 10 lt 90
   fd 10 lt 90+45
   fd sqrt 2*10*10
]
```

New words (procedures)



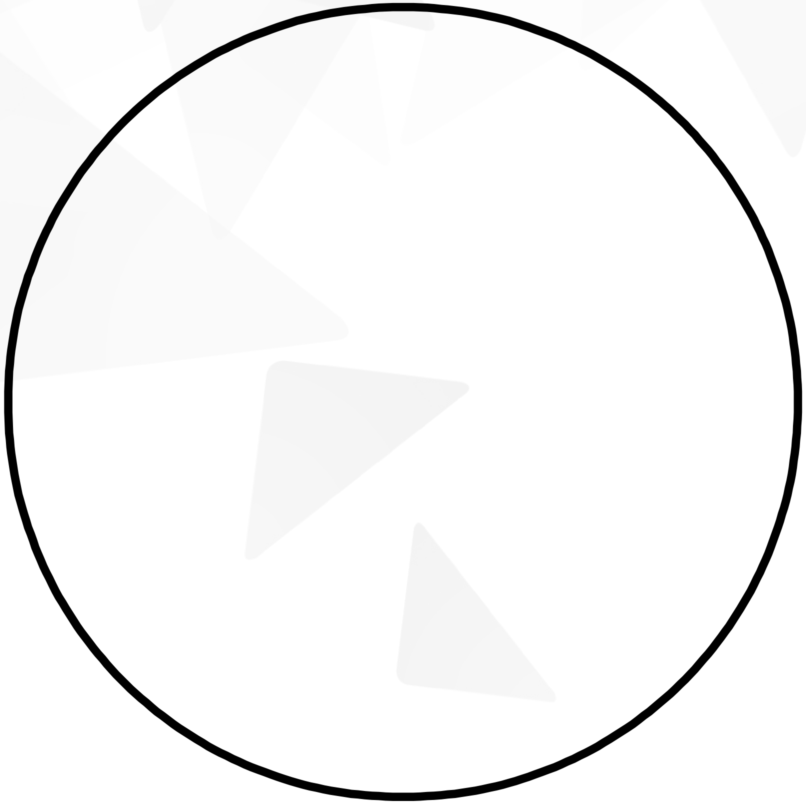
```
to wheel size
   repeat 8 [
      fd size lt 90
      fd size lt 90+45
      fd sqrt 2*size**2
   ]
end

repeat 10 [
   wheel 10 + random 10
   fd 40 lt 36
]
```

Drawing circle (polygon with 360 sides)



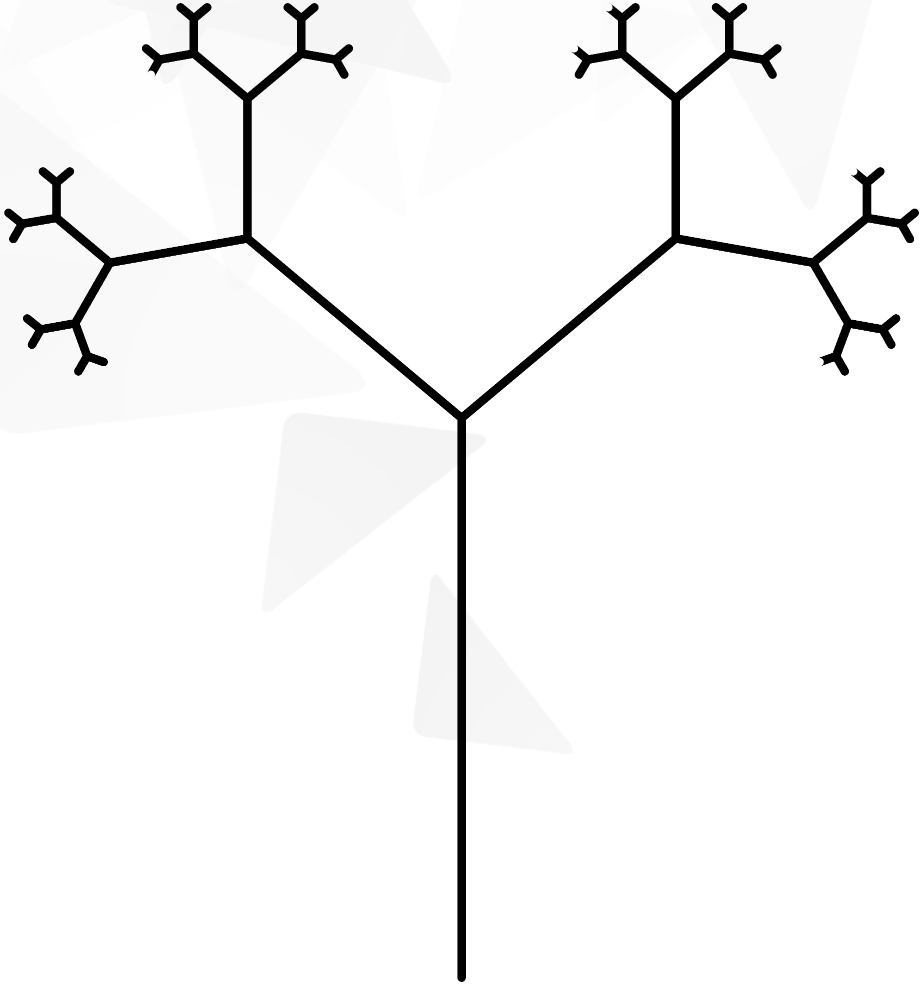
```
repeat 360 [
   fd 1
   lt 1
]
```

With recursion



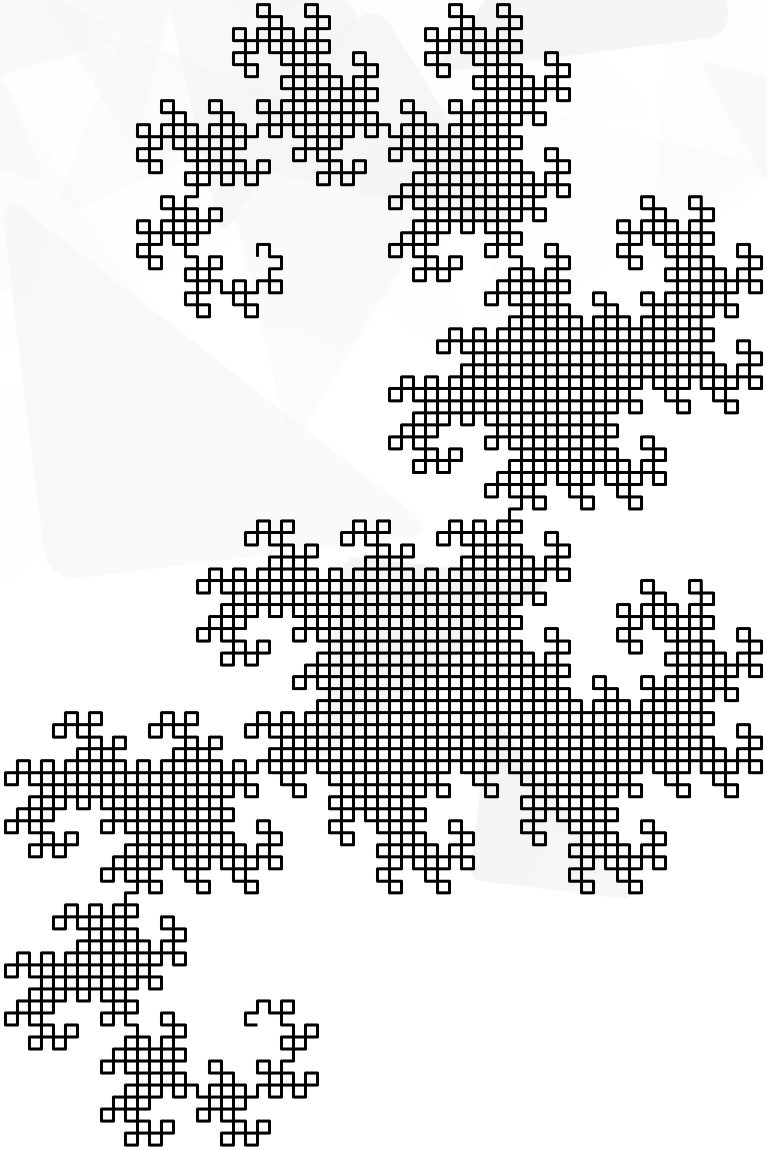
```
to polygon
   fd 1
   lt 1
   polygon
end

polygon
```

Conditions (eg. branch length of a recursive tree)



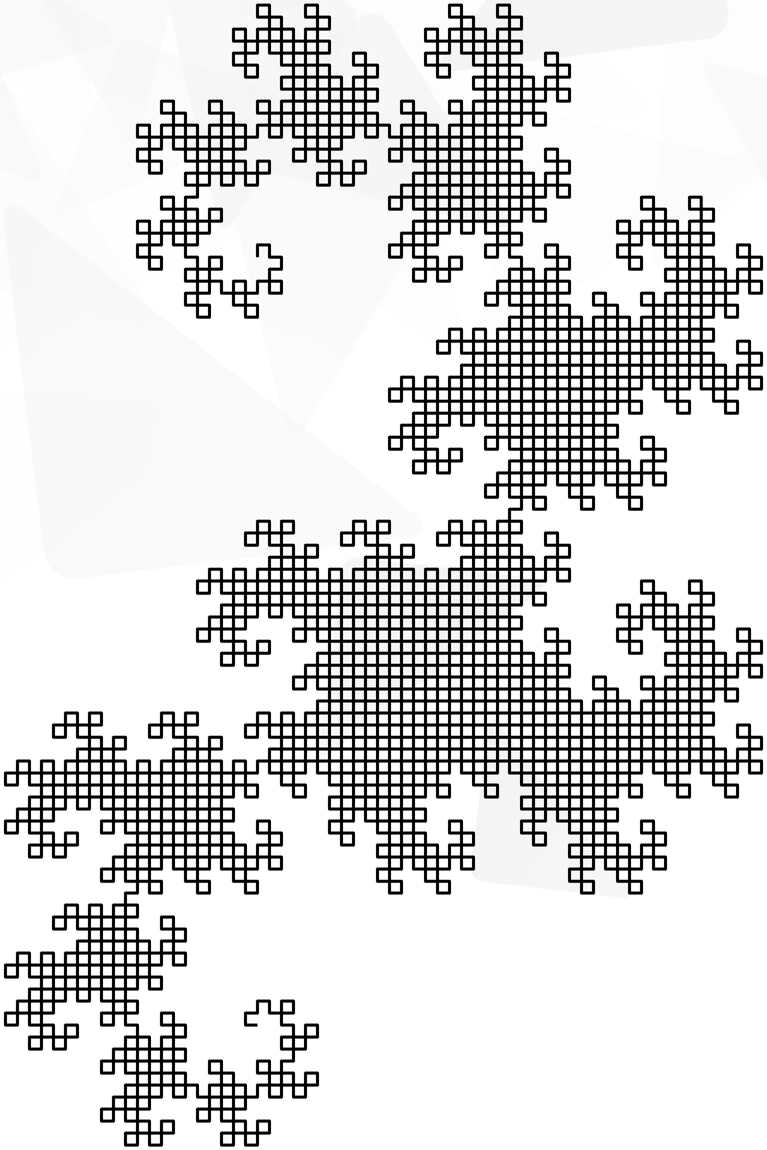
```
to tree len
   if len < 2 [ stop ]
   fd len lt 50
   tree len/2
   rt 100 tree len/2
   lt 50 bk len
end

tree 100
```

Dragon curve



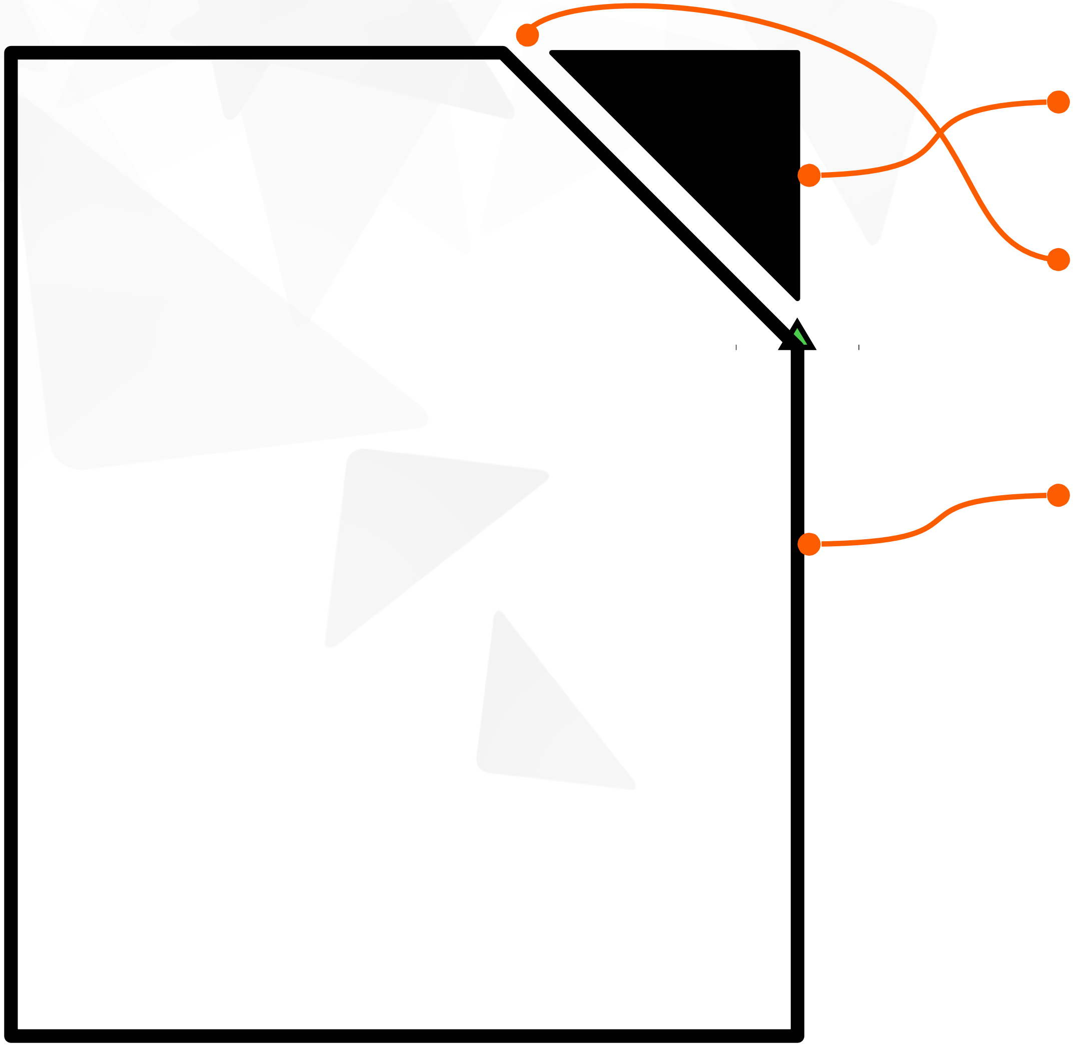
```
to x n
   if n = 0 [ stop ]
   x n-1 rt 90
   y n-1 fd 4
end

to y n
   if n = 0 [ stop ]
   fd 4 x n-1
   lt 90 y n-1
end

fd 4 x 12
```

LibreLogo 0.2 may need explicit parenthesization



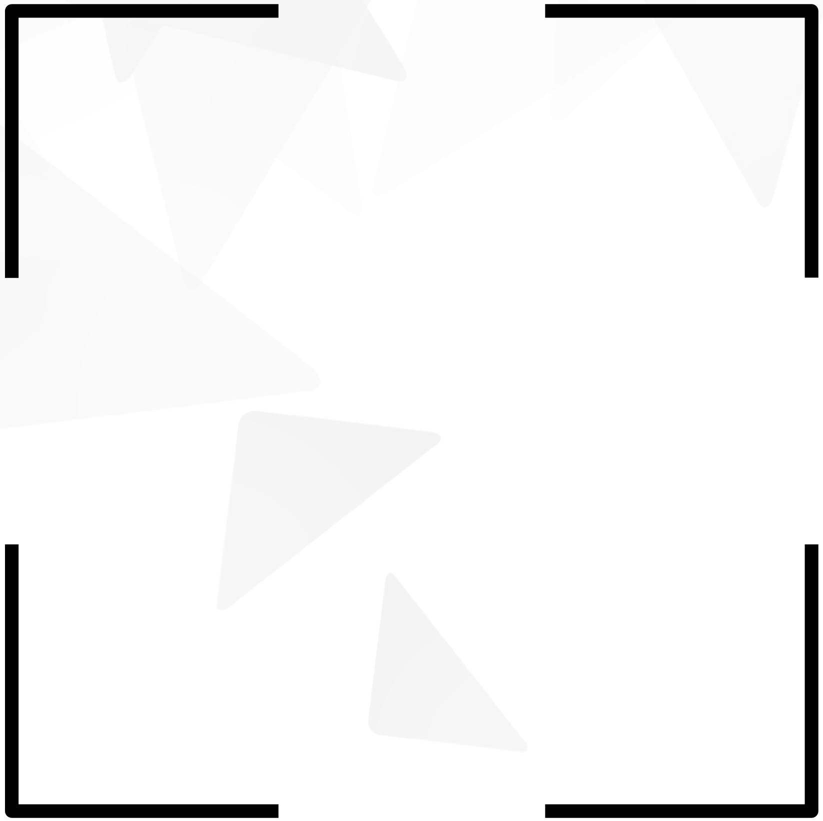
```
to x n
   if n = 0 [ stop ]
   x n-1 rt 90
   y (n-1) fd 4
end

to y n
   if n = 0 [ stop ]
   fd 4 x n-1
   lt 90 y n-1
end

fd 4 x 12
```

Fill and close the actual shape (not flood fill)



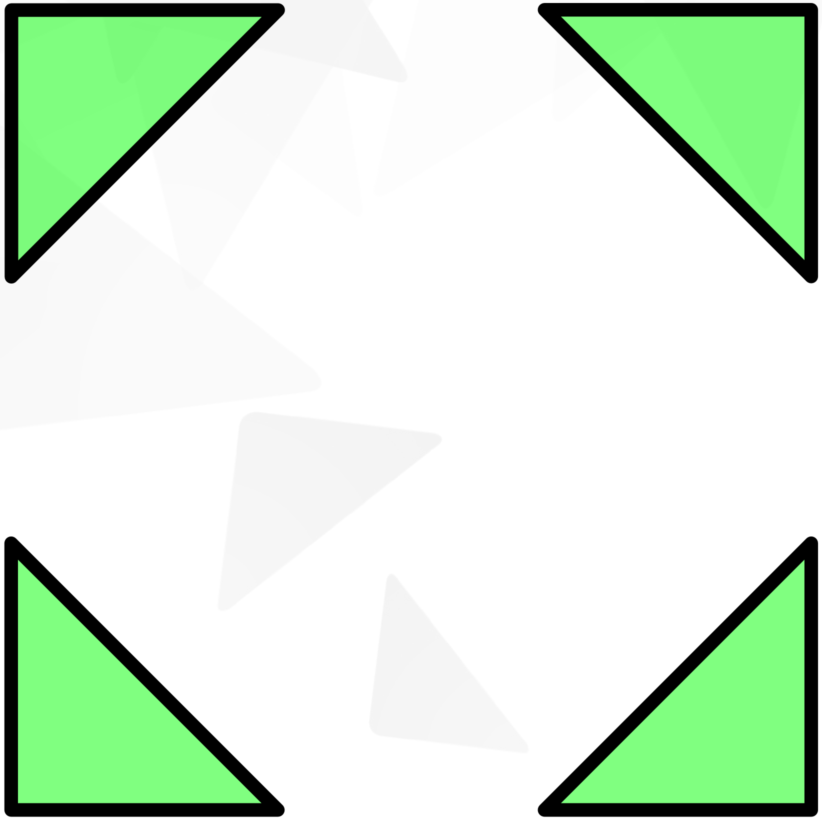
```
fd 25 lt 90 fd 25 fill

penup fd 5 pendown

fd 50 lt 90 fd 100
lt 90 fd 80 lt 90
fd 70 close
```

Penup (pu) doesn’t start new line shapes



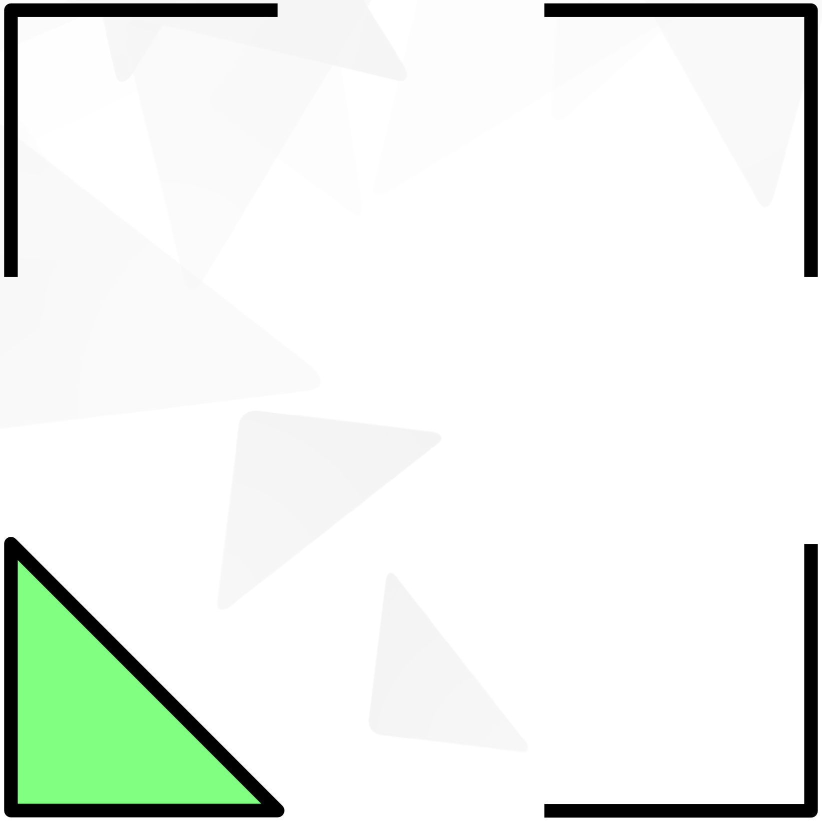
```
repeat 4 [
   pu fd 100 pd fd 100
   rt 90 fd 100
]
```

Penup (pu) doesn’t start new line shapes



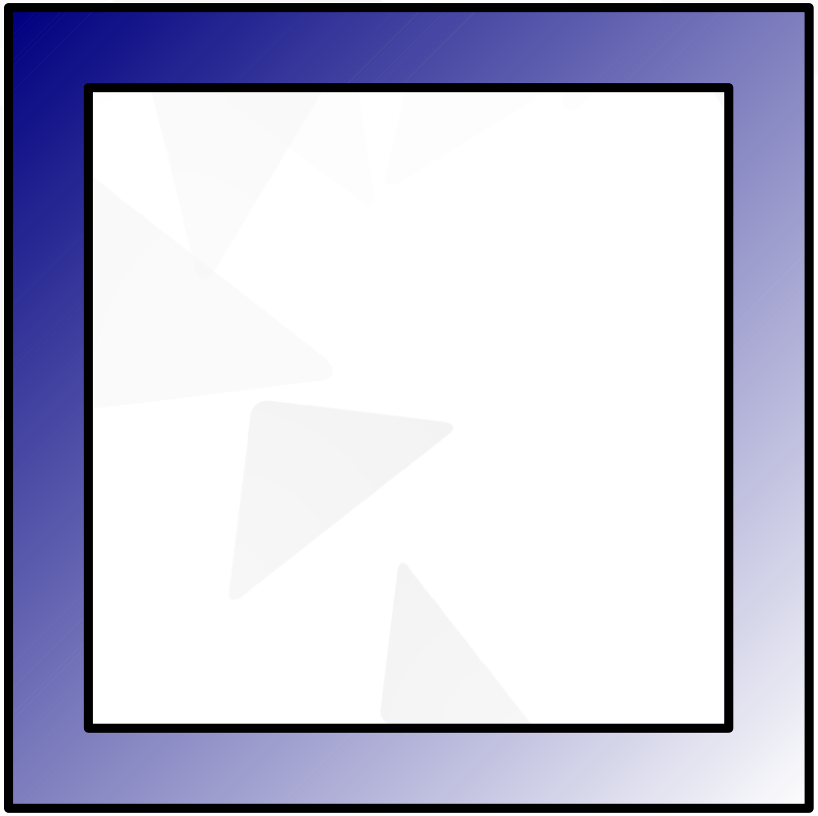
```
repeat 4 [
   pu fd 100 pd fd 100
   rt 90 fd 100
]

fill
```

Use “picture” (pic) for new line shapes



```
repeat 4 [
   pic
   pu fd 100 pd fd 100
   rt 90 fd 100
]

fill
```

Frame (a complex shape)



```
to box x
repeat 4 [ fd x rt 90 ]
end

; two disjoint squares
box 300
pu fd 30 rt 90 fd 30
lt 90 pd
box 240
fill
```

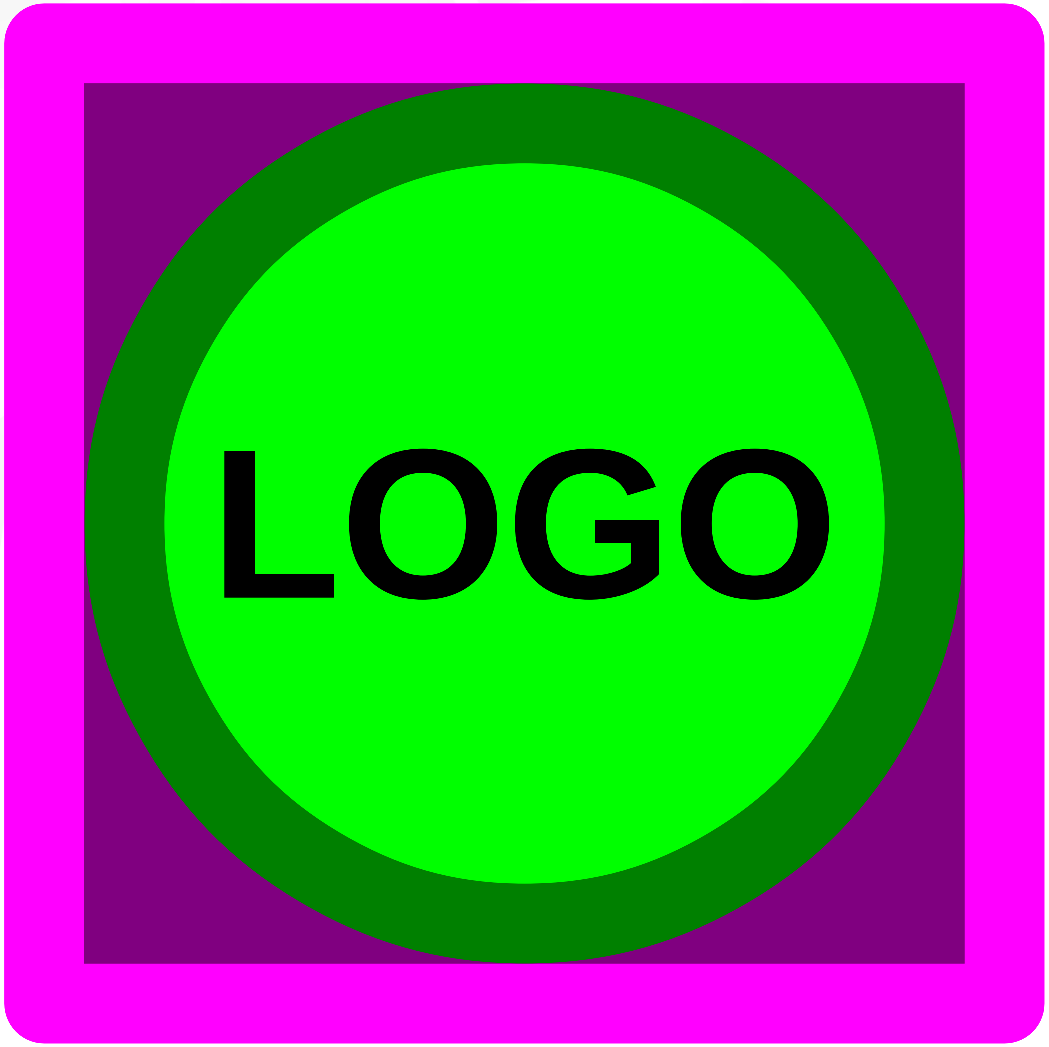
Logo and LibreLogo

Logo	Differences	LibreLogo
turnright 90 = rt 90	**optional clock positions ▶** (suitable for the lower grades)	turnright 90° = rt 90 = turnright 3h
forward 1 = fd 90	**DTP point, inch, cm, mm ▶** **◀ pixel**	forward 1pt = fd 1 = fd 1in/72 = fd 2.54cm/72
fill (flood-fill, need position)	**vector graphics ▶** **◀ raster graphics**	fill (close and fill actual shape)
"word [string]	text notation **writing standard ▶** **◀ formal (LISP)**	“string” (orthography, Writer), 'string' (Python), "word, "word"
lists [] (eg. 1-line instruction list)	Python in Logo turtle shell ▶ ◀ functional programming language	blocks [ ] (need space or line break) and lists [], eg. repeat 5 [ellipse [5, 10]]

Color names

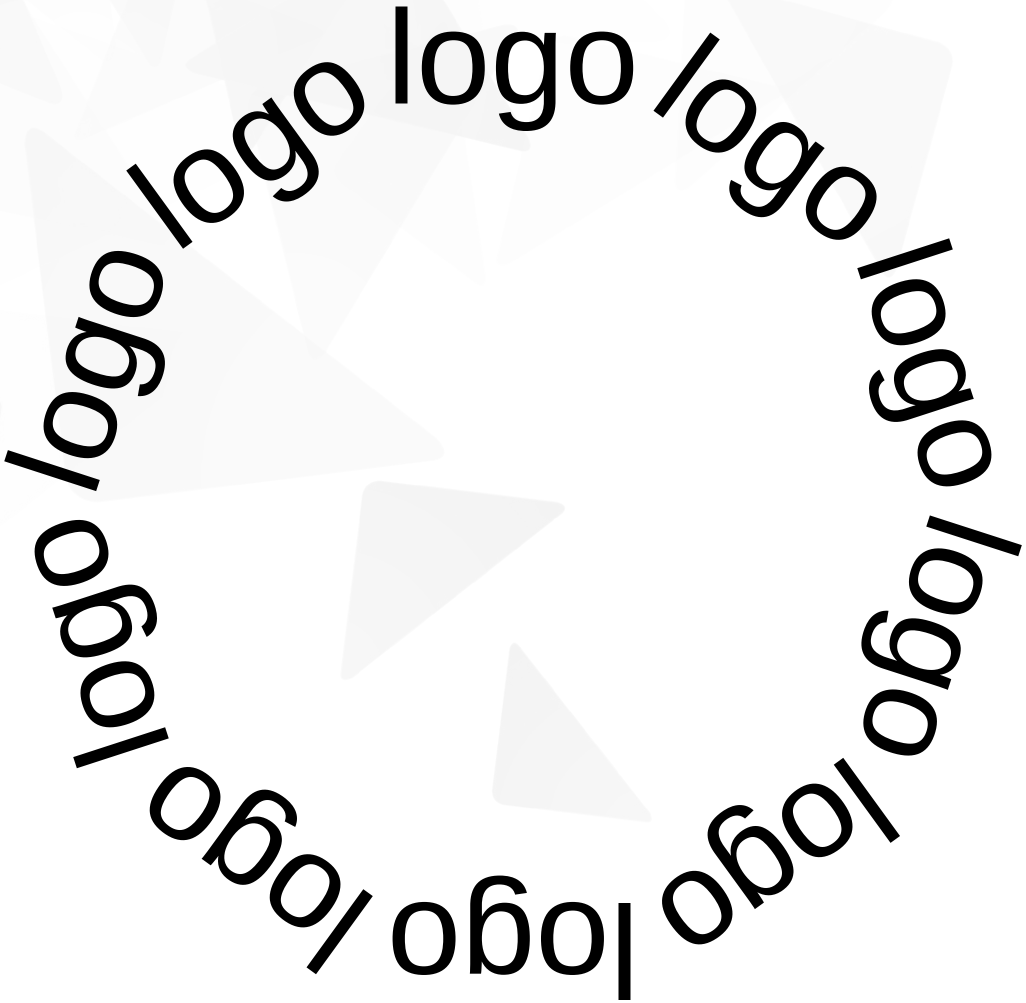
“black”	“silver”	“gray” “grey”	“white”	“maroon”
“red”	“purple”	“fuchsia” “magenta”	“green”	“lime”
“olive”	“yellow”	“navy”	“blue”	“teal”
“aqua” “cyan”	“pink”	“tomato”	“orange”	“gold”
“violet”	“skyblue”	“chocolate”	“brown”	**“invisible”**

Pensize (ps), pencolor (pc), fillcolor (fc), text



```
pensize 30
fillcolor “purple”
pencolor “fuchsia”
square 360
fillcolor “lime”
pencolor “green”
circle 300
fontsize 80

; text of the actual
; shape
text “LOGO”
```

Label

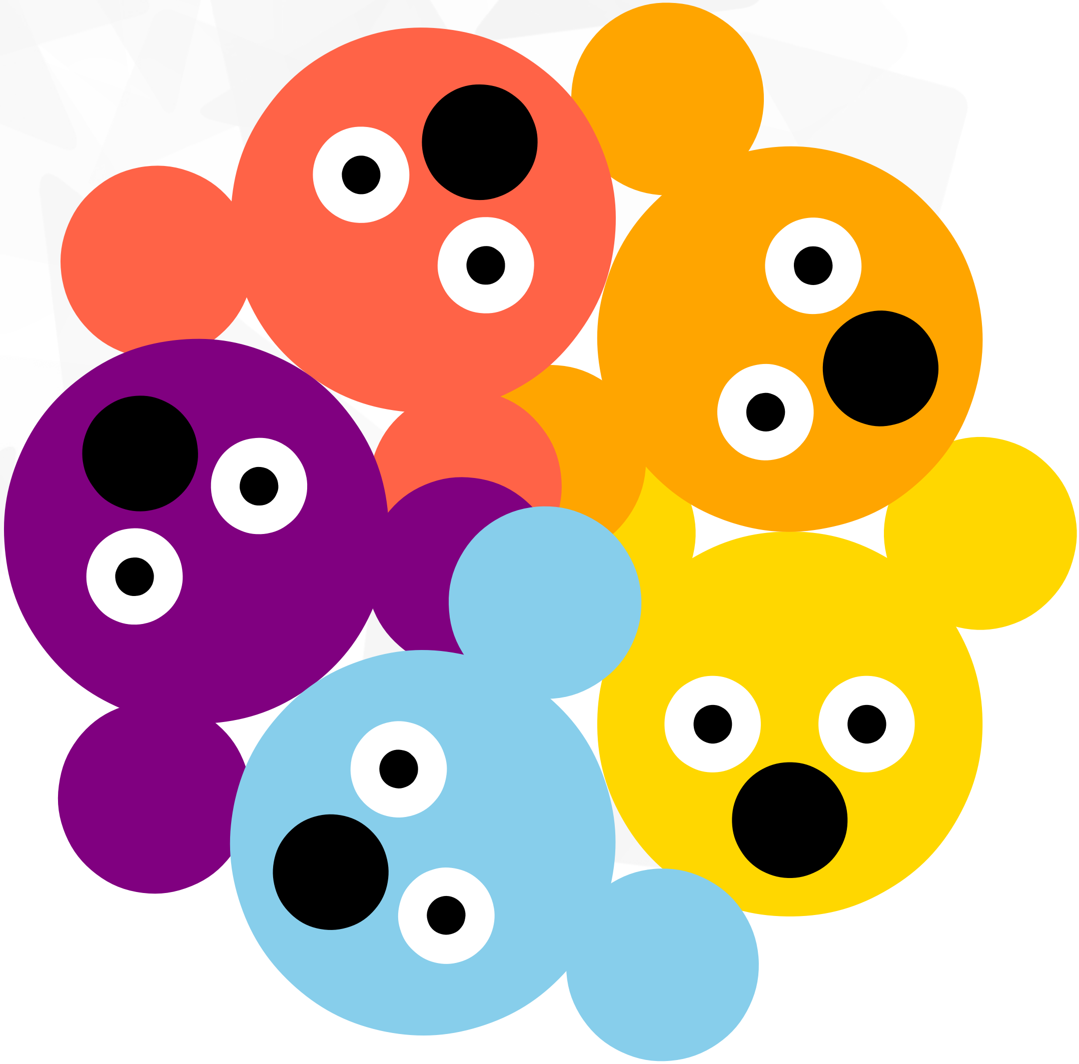


```
fontsize 50
penup
repeat 10 [
   forward 170
   label “logo”
   back 170
   lt 360/10
]
```

Random Unicode 6.0 emoticons

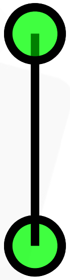
```
penup
fontfamily “Symbola”
repeat 10 [
   pos any
   fontcolor any
   fontsize 50 +
            random 50
   label random “😀😁😂
😃😄😅😆😇😈😉😊😋😌😍😎😏😐
😑😒😓😔😕😖😗😘😙😚😛😜😝😞
😟😠😢😣😤😥😦😧😨😩😪😫😬😭
😮😯😰😱😲😳😴😵😶😷😸😹😺😻
😼😽😾😿🙀”
]
```

Loop – for + in (lists, character strings)



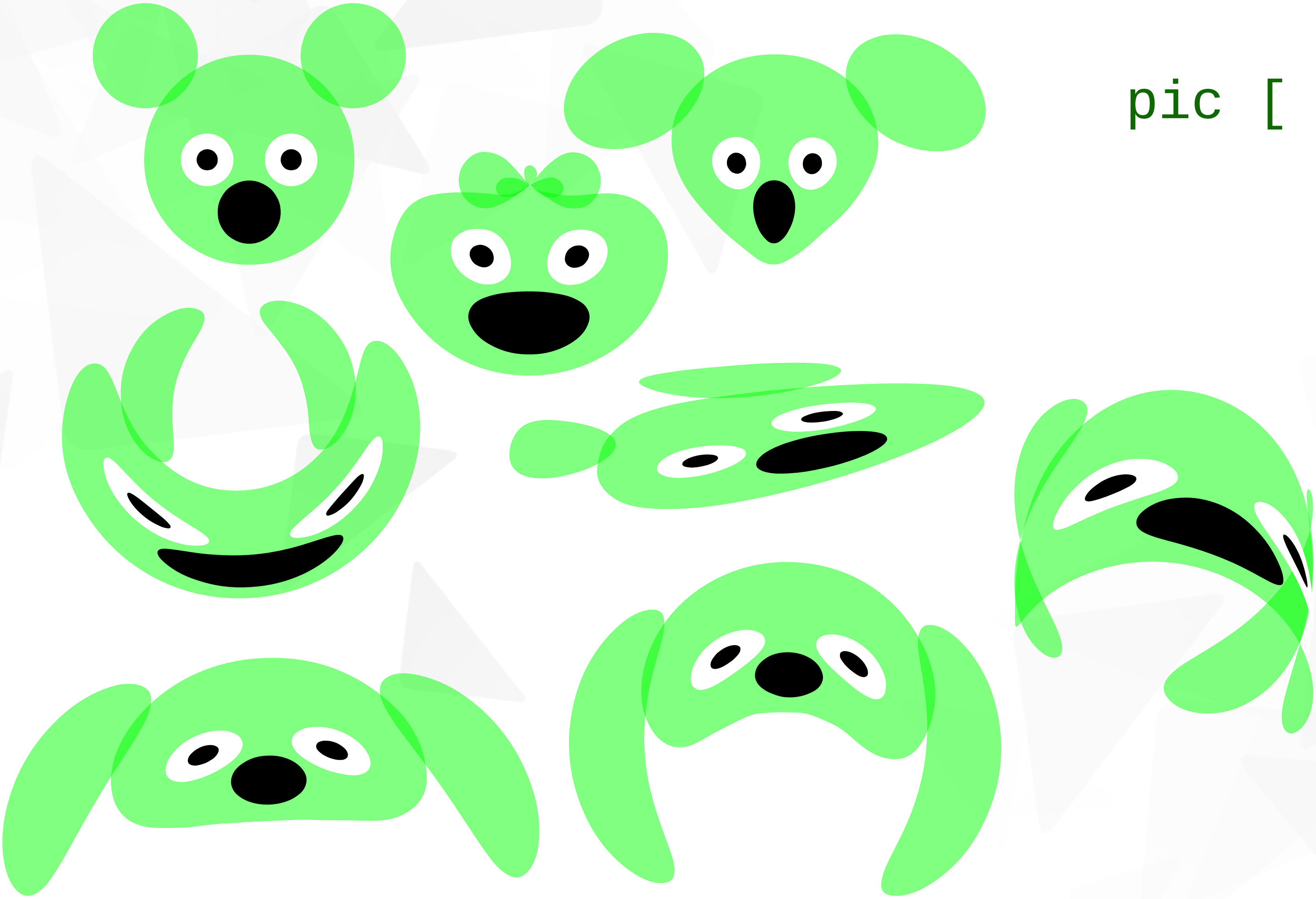
```
to bear
  pu circle 100 lt 45 fd 70
  circle 50 bk 70 rt 90 fd 70
  circle 50 bk 70 rt 45 bk 20
  ; eyes
  repeat 2 [
    fc “white” circle 25
    fc “black” circle 10
    fd 40
  ]
  bk 60 rt 90 fd 25 circle 30
  bk 25 lt 180
end

for k in [“gold”, “orange”, ~
  “tomato”, “purple”, ~
    “skyblue”] [
    fc k bear fd 100 lt 360/5
]
```

Shape groups by picture […]



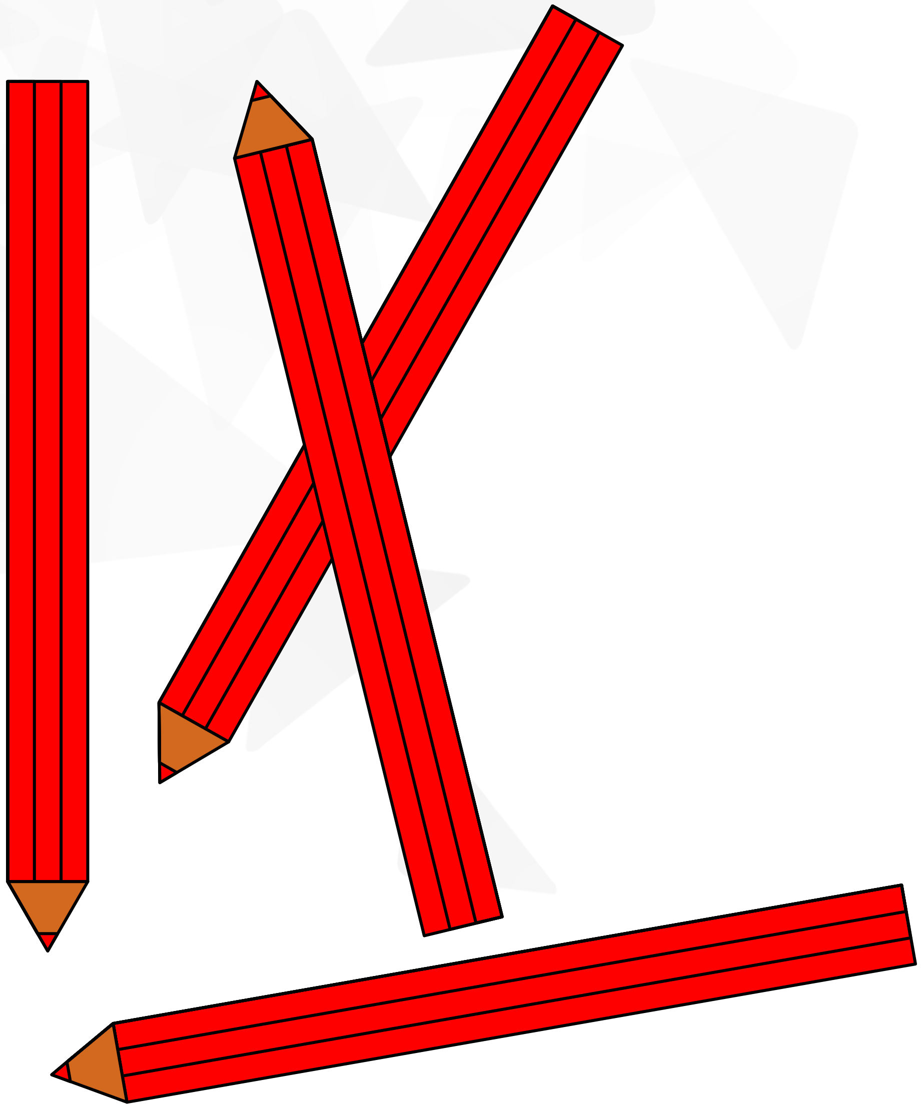
```
pic [
   circle 10
   fd 40 circle 10
]
```

Shape groups by picture […] + Draw effects



```
pic [ bear ]
```

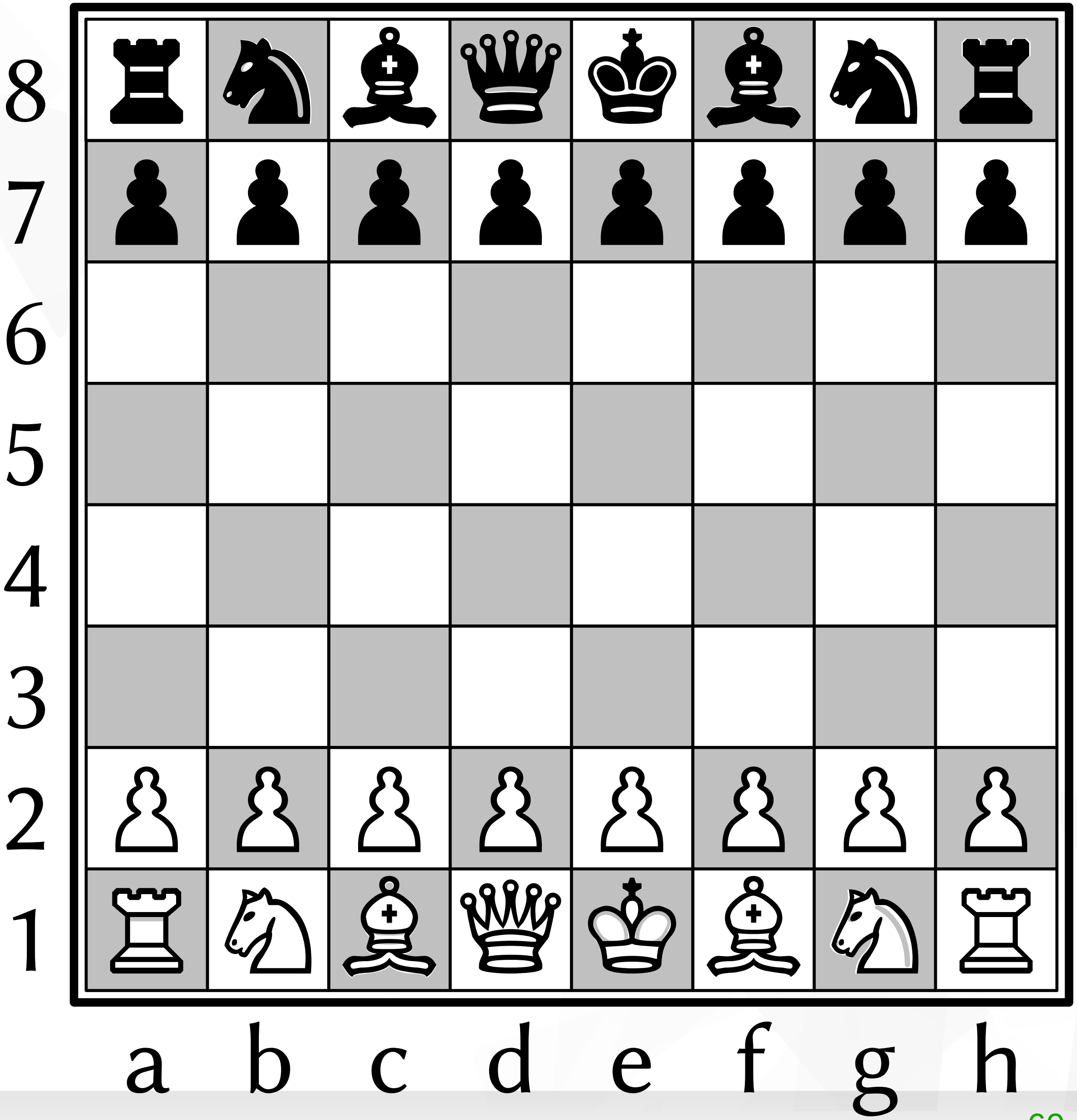
Shape groups by picture […]



```
to triangle size color
    repeat 3 [ fd size lt 120 ]
    fc color fill
end

to box size f
    repeat 2 [ fd size*10 rt 90
    fd size*f rt 90 ]
end

to pencil size color
    box size 1 fc color fill
    box size 2/3 box size 1/3
    close rt 150
    triangle size “chocolate”
    fd size*0.75
    triangle size/4 color
    back size*0.75 lt 150
]
end

pic [ pencil 100 “red” ]
```

Desktop publishing

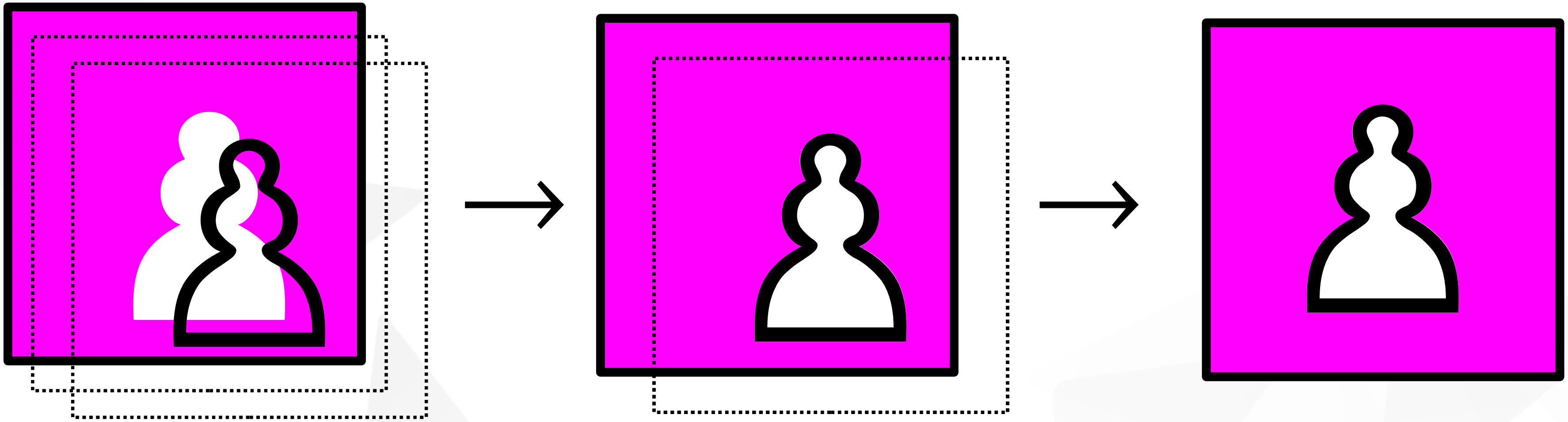
- Unicode chess figures
- automatic or
- manual positioning



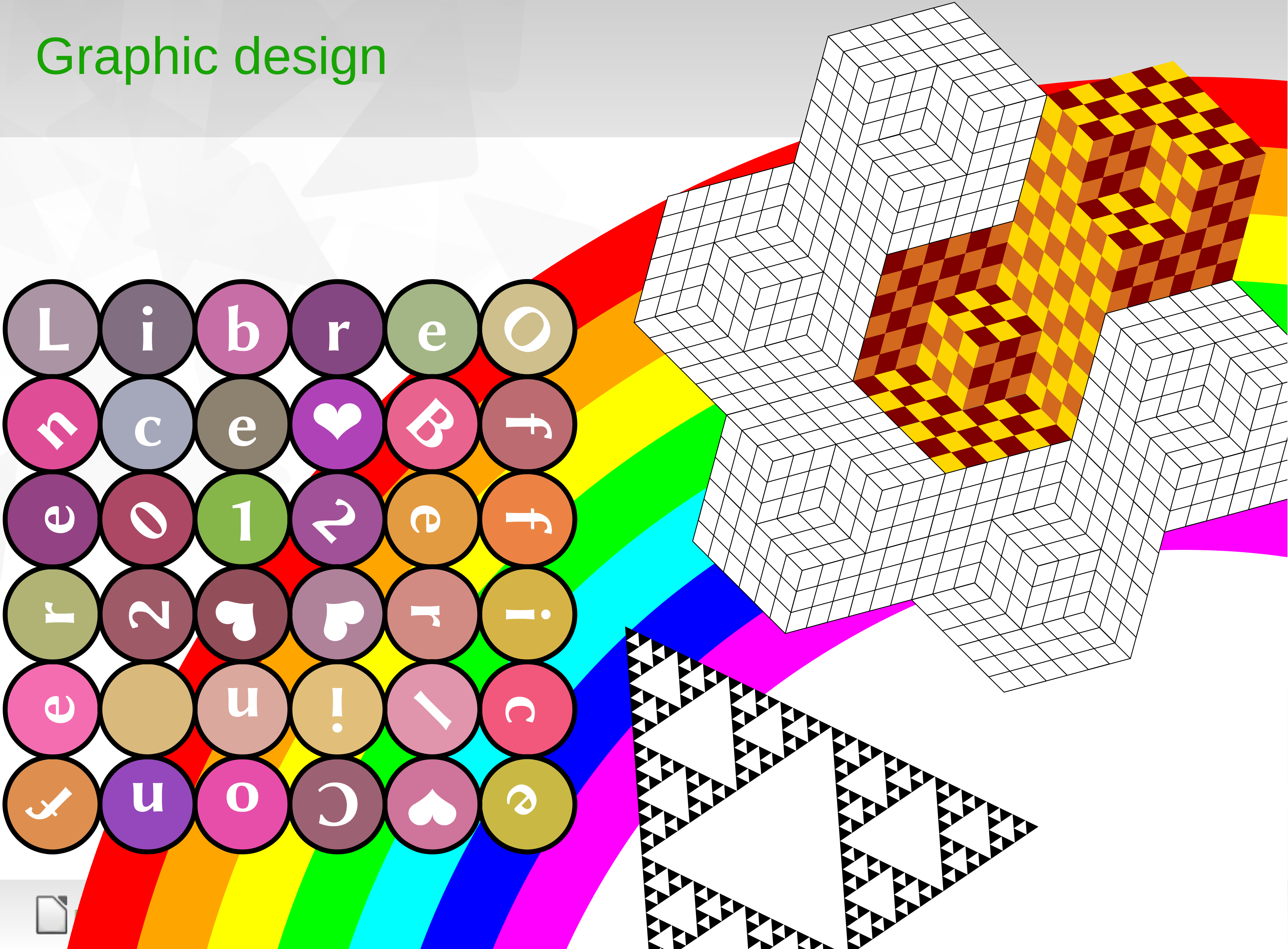
8	♜	♞	♝	♛	♚	♝	♞	♜
7	♟	♟	♟	♟	♟	♟	♟	♟
6								
5								
4								
3								
2	♙	♙	♙	♙	♙	♙	♙	♙
1	♖	♘	♗	♕	♔	♗	♘	♖
	a	b	c	d	e	f	g	h

Complex shape groups for graphical design

- Grouping of Unicode characters and square shapes for white background of figures and comfortable manual positioning



Graphic design



Goals

- Teaching of computing
  - Programming
    - free, portable and modern Logo alternative with
    - Python (data structures, open source of LibreLogo)
  - Word processors
    - Text editing, image handling
  - Migration to free software (LibreOffice)
- LibreOffice
  - Simple programming interface for graphic design
- LibreOffice development
  - Test bed for graphical features and PyUNO
  - Attract more future developers

Plans

- Integration with LibreOffice
- More supported native languages
- Automatic translation between the native language programs
- Reference programs
- Keep it simple (~1 thousand lines in Python)

Documentation

- Introduction: http://numbertext.org/logo/librelogo_en.pdf
- Commands: http://numbertext.org/logo/commands_en.txt

Thank you for your attention!

Find out more at
http://www.numbertext.org/logo