A bit of terminology

26 May 2010

Our first workshop weekend is over. We now see that there are some participants for whom programming is still an unknown skill, whereas for some others it is no problem at all. It would be great if we could still stick together, and everyone of us gains something, no matter where we stand. However, this will not be possible without conscious dedication on behalf of each one of us. Let us try to cooperate towards this goal!

Basic concepts

What has come into the world, when computer and information technology arrived, is a new kind of *thinking*: it is *algorithmic thinking*, or thinking in terms of computability. Only such things and processes can become subject matter of computing that before have gone through three steps of reduction:

- reduction from the world as we know it (because we are part of it) to the semiotic realm. In consequence, only signs and semioses (sign processes) may become operational elements in computing.
- reduction from the world of signs to the realm of syntactics, and that is: to signals. All meaning, purpose, beauty is taken away from the signs, and only their explicit forms remain.
- reduction from the world of signals to the realm of computable signals.

So, when something from the real world is prepared for data processing, all that remains after the semiotic, syntactic, and algorithmic transformations, are computable signals and signal processes. Only the quantitative aspects remain, no qualitative features remain. Only otherness is still kept: one signal is different from another signal. When I can distinguish two signals, they can be dealt with by computers.

A program is a text: a machine-readable description of an algorithm. An *algorithm* is a computable function. Computable functions are defined in mathematics. E.g. by the Turing machine, or by the λ-calculus (CHURCH), or by replacement rules. A program can also be viewed as a structured organization of statements. *Statements* are often also called commands. But that does not fit very well any more.

The statement is the basic form element of programs. In a programming language (like Processing or Java), statements take on special form. As a program designer, you cannot deviate from those forms. Programming is, therefore, similar to learning a foreign language. There are simple and structured statements. One group of statements describes the things a program is allowed to use. These statements are called *declarations*. Other statements describe actions the program is to carry out (to execute). Simple statements are the assignment statement and the function call. Structured statements are the iteration, the alternative, the conditional statement.

summer term 2010 | 2 sws | lab & workshop in blocks | for students of digital media (bachelor or master, all four schools)
extra, independent of modules, no credit | independent study possible for credit, based on this course
limited to 20 participants | register by email | bring your laptop | download Processing from www.processing.org
meetings: 16 april from 16 to 18; 22 & 23 may; 5 & 6 june; 26 june 2010, each from 11 to 18 at HfK 3.09.100

Frieder Nake | nake(at)informatik.uni-bremen.de | (0421) 218 3525 Linzer Str. 9a, room 3015
office hours Wednesday 12 to 14 Linzer Str. & Tuesday at HfK upon arrangement | www.agis.informatik.uni-bremen.de

An important concept in programming is that of a *variable*. A variable is an entity of some sort that may take on a value (of a special kind, called its type). The variable has a name and a location. The name is used to refer to the variable, and thereby to its value. The variable may change its value under the control of the program. In fact, the job of the program is to change the values of variables. Usually, this is done for a certain purpose.

The variable can be thought of as an entity of the structure

(name, type, value, location)

In processing, some types are

int (the set of integer numbers, from a given smallest negative to a given largest positive

float (the set of real numbers, from a smallest to a largest. These are some of the numbers that can be written as decimals)

boolean (the *truth* values true and false)

char (the set of the Latin letters, and the ten decimals, and some more from the keyboard characters. They are written as $'a', 'A', '\bullet'$ etc.)

byte (the set of patterns of 8 bits, from 0000 0000 to 1111 1111. There are 2^8 = 256 different patterns)

color (the set of permitted RGB or HSB codes)

string (the set of sequences of characters, as "John" or "iPhone" or "you+me". They are identified by double quotes.)

array (a field in one, two, or more dimensions of elements of equal type. Access to the elements of an array is via indices as in $a[i, j]$)

A type should be thought of as an algebraic structure. That is a set of elements together with a set of operations on the elements. The type int, e.g., allows for the arithmetic operations + - * / % that we know from school, but also for logical comparisons like < <= >= > == .

In object-oriented programming, this advanced kind of mathematical thinking is itself taken as the subject matter of programming.

Geometric transformations

The basic elements of geometry are points, straight lines, and planes. Restricting our attention to two dimensions (planar geometry), we can describe a point by two coordinates, say P = (x, y).

A straight line segment is nothing but two connected points: (P1, P2).

A plane can be defined by three points: (P0, P1, P2) which is also interpretable as a triangle. (The boundary curve consisting of three straight line segments from P0 toP1, from P1 to P2, and from P2 to P0).

We can generalize this to a polygon $(P_0, P_1, \ldots, P_{n-1})$ of n vertices and n edges.

The most basic geometric transformations are translation, rotation, and scaling.

Translation

The point $P = (x, y)$ is transformed into $P' = (x + t_x, y + t_y)$ where $T = (t_x, t_y)$ is the vector of translation.

Rotation

The point $P = (x, y)$ is transformed into $P' = (x \cos\alpha - y \sin\alpha, x \sin + y \cos\alpha)$ where a is the angle of rotation about the origin of the coordinate system

Scaling

The point $P = (x, y)$ is transformed to $P' = (s_x x, s_y y)$ where s_x and s_y are the scale factors along the two axes.