universität wien

MAGISTERARBEIT

Titel der Magisterarbeit

„How to familiarise pupils with the logic of algorithm“

Verfasserin

Daniela Bäck, BSc.

angestrebter akademischer Grad

Magistra Informatikmanagement (Mag^{a}. rer. soc. oec.)

Wien, im Juni 2009

Studienkennzahl lt. Studienblatt:	A 066922
Studienrichtung lt. Studienblatt:	Informatikmanagement
Betreuerin / Betreuer:	Univ.-Prof. Dr. Wilfried Grossmann

"Only the curious will learn and only the resolute overcome the obstacles to learning. The quest quotient has always excited me more than the intelligence quotient."

Eugene S. Wilson

To my parents for supporting me in every way anybody can think of!

ACKNOWLEDGMENTS

I would like to thank the following people for their help, co-operation and support throughout the project:
Daniel Makowitsch, Gabriele Matschek, Sonja Gutschi, Conny Josefsson, Gerhard Pachler, Dörthe Steup

Especially I want to thank my project supervisor Prof. Wilfried Grossman for all the help, support and guidance.

How to familiarise pupils with the logic of algorithm

1 Introduction ... 7

1.1 Definition of didactics in general and didactics in computing ... 10

1.1.1 Didactics in Computing ... 10

1.1.2 Definition of pedagogy ... 11

1.1.3 Definition of computing ... 11

1.1.4 Definition of algorithms ... 12

2 How pupils learn and how we teach ... 14

2.1 Motivation to learn ... 15

2.2 Learning types ... 17

2.2.1 The Activist ... 18

2.2.2 The Reflector ... 18

2.2.3 The Theorist ... 18

2.2.4 The Pragmatist ... 19

2.3 Learning disorders ... 19

2.4 Basic Learning theories ... 20

2.4.1 Behaviourism ... 20

2.4.2 Cognitivism ... 22

2.5 Link between age and learning ... 23

3 The origin of Informatics ... 24

3.1 The Development of the term “Informatics” ... 26

3.2 Informatics in Use ... 27

3.3 Informatics in Austrian Schools ... 28

3.4 Austria’s educational system in international comparison ... 29

4 Computing in school ... 31

4.1 Computing – hype or means to an end ... 32

4.2 Computer based learning ... 36

4.3 Advantages and Disadvantages of e-learning ... 37

4.4 Multimedia-based learning ... 39

5 Open Learning, the ultimate solution? ... 40

5.1 Algorithm in real life situations ... 42

5.1.1 Logic and logical thinking ... 42

5.2 Algorithm in class 45

5.3 Algorithm in computing 46

5.4 Objectives of didactics in computing 46

5.5 Didactical principles 47

5.6 Allow pupils to invent their own methods 48

5.7 Trial & Error Method 48

6 Hands-on programming 49

6.1 Logo 51

6.1.1 LOGO for learning 51

6.1.2 The Turtle 53

6.1.3 Commands in logo 55

6.2 Python 59

6.2.1 Using Python in school 60

6.2.2 Basic Python Operators 62

Arithmetic Operators 62

Comparison Operators 63

6.3 Scratch 64

6.3.1 Technical Details 64

6.3.2 Noteworthy use of Scratch 65

6.3.3 Smalltalk 66

6.3.4 How to get started in Scratch 67

6.3.5 Programming Environment 68

6.3.6 An example for Scratch in use 73

7 Annex 76

7.1 Abstract English 76

7.2 Abstract German 77

7.3 Curriculum Vitae 78

8 Bibliography 80

Foreword

This paper deals with teachers, students, children, learners, teens and parents. At the beginning I would like to point out that these terms refer to both – males and females. To simplify matters I am using the general expression he, his or him.

1 Introduction

This paper is dedicated to my computing, didactics, pedagogics and multimedia studies. It is thought of being a summary of all elements taught to me during the past years. All discussed fields accompanied me through my educational life and what better place to summarizes all my knowledge than the diploma theses.
In particular this paper discusses learning and teaching in general and specializes more and more on the algorithm topic.

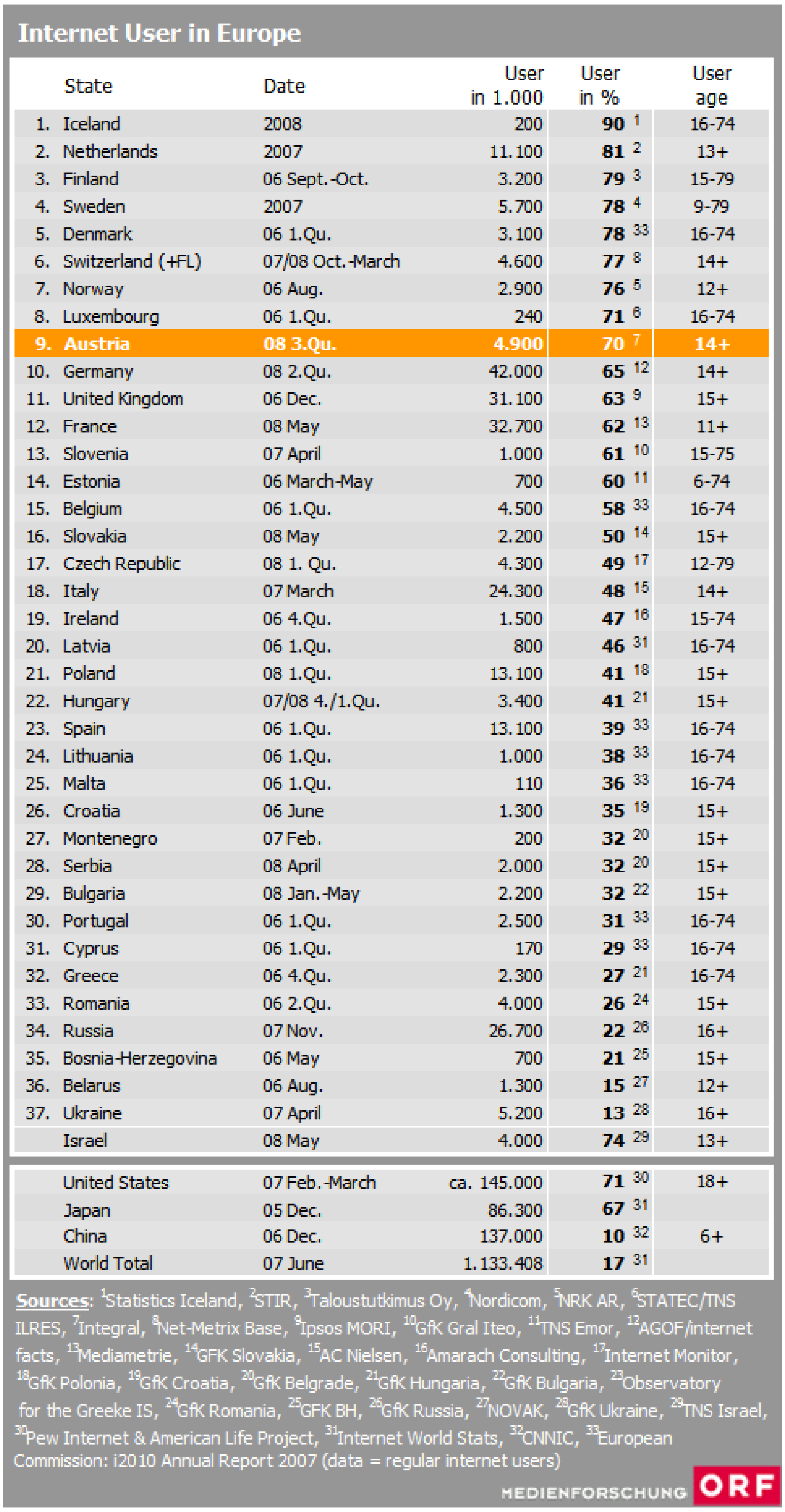
Learning in general is easy - children learn as soon as they see the light of the day and this process keeps on till the end of days (life long learning). So one could say that learning happens automatically – so why send children in school? Isn't it school which makes it hard to learn for children as the "have to" process starts? Children have to go to school, they have to sit there and listen, and they have to do their homework and so on. In the end it all comes down to how children learn and how we teach – choosing the right techniques and the right methods is not easy as there are many different learning types around and not every method is accepted by every child. So is open learning the right answer to this problem or is finding the right method a never-ending challenge for teachers?

Let me give an example – when children grow up bilingually there is no question that they speak both languages without any problems even though they are not taught in a classical way with vocabulary learning and grammar lessons – they just adapt the languages and speak them perfectly well. Then what happens in school? First children are pressed into a system where they have to speak one language and the other language is disregarded and also school starts to concentrate more on the analytical processes of information.

These processes might not match with the child's way of handling information and all of a sudden the child has problems with learning and starts struggling with its inner logic. Especially when it comes to computing in school a lot of didactical questions arise but the world cannot be imaged without computers so it is absolutely necessary that the integration process of computers and informatics starts at early stages in school life.

Also teachers must have the foresight and commitment of using ICT (Information and Communication Technology) in school as the future will not happen without computer and the sooner children start to get used to working in the ICT environment the easier it will be for them to perform computer actions in the future (might it be programming or just using it for every day life actions).
Here it is important that the teacher has the right way of using didactical and pedagogical methods to prevent children from capitulation. But what is the meaning of these important terms which are often used but also often not even understood?

Before I begin to address these issues let me show a statistic table regarding internet usage to underline the significance of computing and information processing these days:



Internet User in Europe

	State	Date	User in 1.000	User in %	User age
1.	Iceland	2008	200	**90** [1]	16-74
2.	Netherlands	2007	11.100	**81** [2]	13+
3.	Finland	06 Sept.-Oct.	3.200	**79** [3]	15-79
4.	Sweden	2007	5.700	**78** [4]	9-79
5.	Denmark	06 1.Qu.	3.100	**78** [33]	16-74
6.	Switzerland (+FL)	07/08 Oct.-March	4.600	**77** [8]	14+
7.	Norway	06 Aug.	2.900	**76** [5]	12+
8.	Luxembourg	06 1.Qu.	240	**71** [6]	16-74
9.	**Austria**	**08 3.Qu.**	**4.900**	**70** [7]	**14+**
10.	Germany	08 2.Qu.	42.000	**65** [12]	14+
11.	United Kingdom	06 Dec.	31.100	**63** [9]	15+
12.	France	08 May	32.700	**62** [13]	11+
13.	Slovenia	07 April	1.000	**61** [10]	15-75
14.	Estonia	06 March-May	700	**60** [11]	6-74
15.	Belgium	06 1.Qu.	4.500	**58** [33]	16-74
16.	Slovakia	08 May	2.200	**50** [14]	15+
17.	Czech Republic	08 1. Qu.	4.300	**49** [17]	12-79
18.	Italy	07 March	24.300	**48** [15]	14+
19.	Ireland	06 4.Qu.	1.500	**47** [16]	15-74
20.	Latvia	06 1.Qu.	800	**46** [31]	16-74
21.	Poland	08 1.Qu.	13.100	**41** [18]	15+
22.	Hungary	07/08 4./1.Qu.	3.400	**41** [21]	15+
23.	Spain	06 1.Qu.	13.100	**39** [33]	16-74
24.	Lithuania	06 1.Qu.	1.000	**38** [33]	16-74
25.	Malta	06 1.Qu.	110	**36** [33]	16-74
26.	Croatia	06 June	1.300	**35** [19]	15+
27.	Montenegro	07 Feb.	200	**32** [20]	15+
28.	Serbia	08 April	2.000	**32** [20]	15+
29.	Bulgaria	08 Jan.-May	2.200	**32** [22]	15+
30.	Portugal	06 1.Qu.	2.500	**31** [33]	16-74
31.	Cyprus	06 1.Qu.	170	**29** [33]	16-74
32.	Greece	06 4.Qu.	2.300	**27** [21]	16-74
33.	Romania	06 2.Qu.	4.000	**26** [24]	15+
34.	Russia	07 Nov.	26.700	**22** [26]	16+
35.	Bosnia-Herzegovina	06 May	700	**21** [25]	15+
36.	Belarus	06 Aug.	1.300	**15** [27]	12+
37.	Ukraine	07 April	5.200	**13** [28]	16+
	Israel	08 May	4.000	**74** [29]	13+
	United States	07 Feb.-March	ca. 145.000	**71** [30]	18+
	Japan	05 Dec.	86.300	**67** [31]	
	China	06 Dec.	137.000	**10** [32]	6+
	World Total	07 June	1.133.408	**17** [31]	

Sources: [1]Statistics Iceland, [2]STIR, [3]Taloustutkimus Oy, [4]Nordicom, [5]NRK AR, [6]STATEC/TNS ILRES, [7]Integral, [8]Net-Metrix Base, [9]Ipsos MORI, [10]GfK Gral Iteo, [11]TNS Emor, [12]AGOF/internet facts, [13]Mediametrie, [14]GFK Slovakia, [15]AC Nielsen, [16]Amarach Consulting, [17]Internet Monitor, [18]GfK Polonia, [19]GfK Croatia, [20]GfK Belgrade, [21]GfK Hungaria, [22]GfK Bulgaria, [23]Observatory for the Greeke IS, [24]GfK Romania, [25]GFK BH, [26]GfK Russia, [27]NOVAK, [28]GfK Ukraine, [29]TNS Israel, [30]Pew Internet & American Life Project, [31]Internet World Stats, [32]CNNIC, [33]European Commission: i2010 Annual Report 2007 (data = regular internet users)

MEDIENFORSCHUNG ORF

Figure 1: Internet User in Europe
[http://mediaresearch.orf.at/index2.htm?international/international_nutzer.htm (30.01.2009)]

1.1 Definition of didactics in general and didactics in computing

The term didactics is very complicated to define as many different definitions are floating around but what all of these resources have in common is – that didactics has the objective of “WHAT should be taught and HOW should content be taught”? Didactic works with the question of learning aims and learning context.

In combing through different reference books I found some definitions which can be summarised in one simple sentence: Didactics is the science of teaching and learning.

1.1.1 Didactics in Computing

Technical didactics is a special kind of didactics – it establishes a relationship between a special branch of science and real life. It makes the gained results for school or more general for advanced training, continued education and life long learning of children and grown ups available.

The following tasks belong to this process:

- definition of aims
- development of concepts of methods and organisation form lessons
- determination of which ideas, methods and findings should be communicated during lessons
- matching of lesson content to curriculums and their continuous update regarding new findings in science and didactics

[cf. Sigrid Schubert/Andreas Schwill, Didaktik der Informatik (Heidelberg. Berlin: Spektrum Akademischer Verlag, 2004), page 17]

1.1.2 Definition of pedagogy

Pedagogy is the theoretical and practical science when it comes to questions regarding education. It is a more or less fuzzy term which defines a scientific domain and lays its focus especially on questions of development and reasons of aims in education and school.

Pedagogic is the lesson of education. It has the task to offer help for learning and to help to develop an autonomous personality.

[c.f. Gudjons, Herbert, ^{6}1999: Pädagogisches Grundwissen. Bad Heilbrunn: Klinkhardt, page 25-30]

1.1.3 Definition of computing

Computing can be defined as the study of computers and their applications. It is a broad field of science which lays focus on data processing but also on data handling and application programs.

Informatics (a combination of the expression “information” and “automatic”, derived from French) is the science of the systematic and automatic processing of information – hence the “science of the computer”. The English speaking world prefers the term “computer science”, the expression “informatics” is rarely used. A landmark for the acceptance of the educational value of informatics in German-speaking countries in the mid 1980s was the claim that it constitutes a fourth cultural technique in addition to reading, writing and calculating. As Klaus Haefner’s sensational book “Die neue Bildungskrise” (1982), (“The New Educational Crisis”) had attracted considerable attention among both those in charge of education and among employers and employed Informatics became a subject on its own and was made compulsory in the fifth form of Secondary Academic Schools under Minister of Education Dr. Helmut Zilk in the school year 1985/86. Moreover, informatics was included as a new subject in the curriculum of the Pre-vocational year.

[cf. Reiter, Anton 2005, 20 years of Informatics Instruction in Austrian Schools and the Use of ICT in Class, Vienna: CDA Verlags- und HandelsgesmbH, page 6]

1.1.4 Definitions of algorithms

An algorithm is a procedure or formula for solving a problem and accomplishing a procedure, as in a computer program. An exact characterization of an algorithm is necessary to draft a problem solving method which can be executed via a computer. This exact design of an algorithm is called a program. Languages which are used to phrase these algorithms and programs are called programming languages. [cf. Sigrid Schubert/Andreas Schwill, Didaktik der Informatik, Heidelberg. Berlin: Spektrum Akademischer Verlag, 2004, page 5]

> "An algorithm is a procedure to accomplish a specific task. An algorithm has to solve a general, well-specified problem. An algorithmic problem is specified by describing the complete set of instances it must work on and what properties the output must have as a result of running on one of these instances. The distinction between a problem and an instance of a problem is fundamental. An algorithm is a procedure that takes any of the possible input instances and transforms it to the desired output."
>
> [Skiena, Steven S. 1998: The Algorithm Design Manual. USA: Springer-Verlag, page 3]

A very simplified way of explaining an algorithm is to imagine a kitchen, containing a supply of ingredients, an array of baking utensils, an oven, and a (human) baker. Baking a delicious raisin cake is a process that is carried out from the ingredients, by the baker, with the aid of the oven, and, most significantly, according to the recipe. The ingredients are the input to the process, the cake is its output, and the recipe is the algorithm. In other words, the algorithm describes the activities that form the process. The recipes or algorithms are generally called software, whereas utensils and oven represent what is generally known as hardware. The baker, in this case, can be considered a part of the hardware.
[cf. Harel, David, ^{2}1998: Algorithmics, The Spirit of Computing, England: Addison-Wesley Publishing Company, page 4]

Flowcharts are often used to represent algorithm – an example:

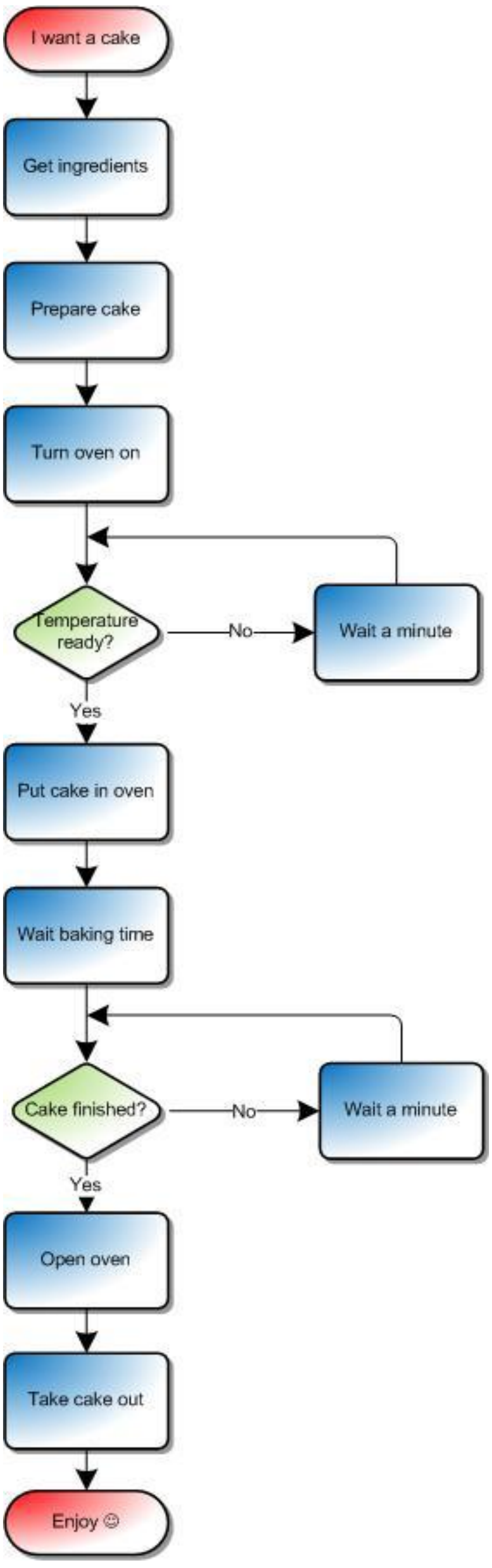


Figure 2: Example Flow-Chart Diagram

2 How pupils learn and how we teach

Each person has personal experiences with school, teaching, education, teachers and learning. Often there are different premises on both sides - teachers like to teach how they learned during their education – but children can not follow these methods because they prefer a complete different learning style and so the difficulties are predictable. Also it seems to be typical that teachers focus on cognitive abilities and fill children with facts – here they tend to forget that this quantitative input does not likely lead to a knowledge output from the learner.

As we know from the science of developmental psychology learning improves when the learner becomes an active participant in the educational process. When selecting among several teaching methods, it is best to choose a method that allows the child to be involved heavily. Also it is a good hint to vary the different teaching methods so it can be assured that all children get addressed in their different learning style. Teachers should also lay an eye on environmental conditions which support open exchange, sharing of options, and problem solving strategies.
[cf. Meyer, Hilbert ^{6}1994: UnterrichtsMethoden. II: Praxisband. Frankfurt am Main: Cornelsen, page 130-135]

Mitchel Resnick of the Media Laboratory Massachusetts Institute of Technology wrote a report regarding “Rethinking Learning in the Digital Age” and has his own opinion of the need of rethinking how children learn.

> He says: *“We need to fundamentally reorganise classrooms. Instead of a centralized-control model (with a teacher delivering information to a roomful of students), we should take a more entrepreneurial approach to learning. Students can become more active and independent learners, with the teacher serving as a consultant, not chief executive. Instead of diving up the curriculum into separate disciplines (math, science, social studies, language), we should focus on themes and projects that cut across the disciplines, taking advantage of the rich connections among different domains if knowledge. Instead of dividing students according to age, we should encourage students of all ages to work together on projects, enabling them to learn from one another (and to*

learn by teaching one another). Instead of dividing the school day into hour-long slices, we should let students work on projects for extended periods of time, enabling them to follow through more deeply and meaningfully on the ideas that arise in the course of their work."

[Report, Chapter 3: Rethinking Learning in the Digital Age, Mitchel Resnick, Media Laboratory Massachusetts Institute of Technology, 2002 (http://web.media.mit.edu/~mres/ 30.01.2009)]

2.1 Motivation to learn

The best didactical approaches and the best use of methods can fail if children are not motivated to learn so it is an important task for the teacher to find ways to motivate the learner.

The relation between motivation and learning is always a two-way procedure: motivation is the precondition as well as the result of learning processes. Success in learning can be more than doubled if motivation is reinforced.

[cf. Peter Hubwieser, Didaktik der Informatik Berlin Heidelberg: Springer-Verlag, 2000, 2001, 2004, page16]

The Centre for Educational Research and Innovation suggests three principles that should govern the design and evaluation of educational practices, including computer software of education.

- the context in which new skills are taught and practiced should match the students worldview sufficiently but should contain enough originality to present a challenge. The student must be able to recognise success and progress.

- If necessary, the teacher or the instructional computer program must help the student to identify individual successes and to recognise progress. For example, a student may not easily recognise that a revised essay is better than the earlier version or that he is fast or more accurate in solving certain mathematics problems then he was a few days ago.

- It may be necessary to vary the size of challenges posed to different students, since part of any student's experience is a sense of general likelihood of success and failure.

[cf. Centre for educational research and innovation, Information Technologies and basic learning (OECD, 1987), page 25]

Tips for improving motivation in practise:

- mind expectations of learner
- find closeness to the field of experience of the learner
- offer elements of surprise
- do not wake to high expectations
- try to motivate more in an intrinsic than in an extrinsic way
- communicate to fulfil tasks with great pleasure
- mind spontaneous interests
- create moments where decision making is necessary
- mind environmental aspects (air, temperature, light, social climate)

[cf. Peter Hubwieser, Didaktik der Informatik, Berlin Heidelberg: Springer-Verlag, 2000, 2001, 2004, page 16]

"The best way to create interest in a subject is to render it worth knowing, which means to make the knowledge gained usable in one's thinking beyond the situation in which learning has occurred."

Bruner (1960)

2.2 Learning types

When it comes to learning – there are many different learning types to consider. This makes it very hard for teachers in classes as they have to plan their lessons they way that every pupil can follow the teaching and can get the best out of it.

There are a lot of different scientists who are investigating the topic of the different learning types. Honey & Mumford identify in their book "The Manual of Learning Styles: Peter Honey and Alan Mumford" the 4 different learning types/styles:

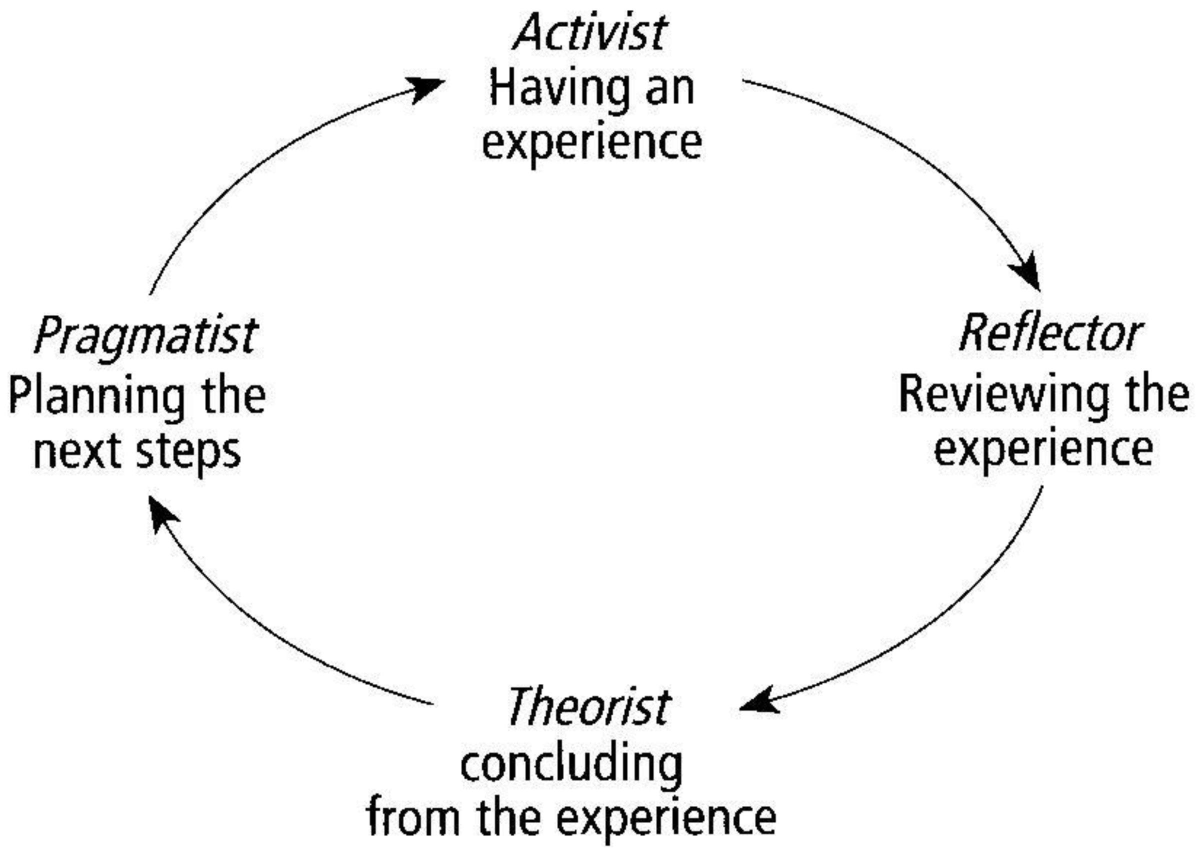


Figure 3: Learning types circle

[http://www.emeraldinsight.com/fig/0370270801002.png (08.04.2009)]

1) Learningtype 1: Activist – what is new? I'm game for anything.
2) Learningtype 2: Reflector – I'd like time to think about this.
3) Learningtype 3: Theorist – How does this match to that?
4) Learningtype 4: Pragmatist – How can I relate this to practise?

2.2.1 The Activist

- Activists enjoy anything new
- They tend to handle problems by brainstorming
- They get bored with longer-term consolidation and implementation
- Activists learn best from fresh experiences
- Activists learn least well from inactive situations where do nothing like reading, watching or listening to lectures, is happening - especially those on theory and history

2.2.2 The Reflector

- Reflectors like to stand back to weigh experiences and emblaze them from many different perspectives
- To be cautious is their philosophy
- They like to observe other people in action and they prefer to stand back in meetings and discussions
- Reflector learn best from activities where they are allowed to stay in the background to watch and listen
- Reflectors learn least well when they are pushed into things

2.2.3 The Theorist

- Theorists like to analyse, display and illustrate
- They raise rationality and logic above all other features
- They are dedicated to rational objectivity and they like to solve tasks in a detached, and analytical way
- Theorists learn best when they are asked to do something without any obvious reason, when activities are happening in an unstructured and indistinct way and when emotion is stressed

2.2.4 The Pragmatist

- Pragmatists are keen testing ideas, techniques and theories to see if they are practical in real life
- They are basically practical and indigenous people. They like making practical decisions and also solving problems
- Pragmatists learn best if there is an relevant connection between the subject and their current task
- Pragmatists learn least well where there are no immediate gratifications from the activity and the learning occurrences or their organisers seem to be far-out from reality

[cf. Lecture notes "Computer Based Learning" Geneen Stubbs, University of Glamorgan 2001, United Kingdom; cf.http://www.mftrou.com/honey-mumford.html 08.04.2009; cf. http://www.ic.polyu.edu.hk/oess/POSH/Student/Learn/Learning_to_learn.htm 08.04.2009]

2.3 Learning disorders

Many times teachers have to deal with learners who have learning disorders. But it is important to differentiate a learning disorder to a learning dislike. Learning disorders are normally diagnosed when a child has low learning results over a long distance – these learning results are substandard to standard. These problems can be divided in area specific and interdisciplinary disorders. To the area specific disorders count orthography, reading and calculation difficulties – especially in this case the acquirement of the basic cultural techniques like reading, writing and calculation is negatively affected. The area of interdisciplinary disorders contains an overall learning disorder, underachievement and learning handicaps. It is characteristic for this kind of disorder that several subjects are affected.

[cf. Frank Dammasch/Dieter Katzenbach (HRsg.), Lernen und Lernstörungen bei Kindern und Jugendlichen. Brandes & Apsel Verlag GmbH, 2004, page 70]

There are several didactical ways – how children should be integrated in class, they should develop ideas which can be for interest for the whole class – like work a topic out and present it to the whole class or design a tricky game which then the class colleagues should solve and many small things more.

Krajewski [cf. Kristin Krajewski, Vorhersage von Rechenschwäche in der Grundschule, Hamburg: Kovac, 2003] found out that early knowledge in number and theory of sets stays in very close combination with good understanding in mathematics. So this shows how important it is, especially in regards to computing and algorithms, to start with getting children to understand numbers and logical sequences early.

2.4 Basic Learning theories

For the design of learning processes the most common theories are the behaviourism and the cognitivism. Both play an important role when it comes to teaching psychology as from their basic statements an approach to our actual attitude in teaching can be drawn.

2.4.1 Behaviourism

This theory indicates that learning occurs through interactions with the environment. Ivan Pavlov (a Russian physiologist) discovered that classical conditioning is a learning process that occurs through relations between an environmental stimulus and a naturally occurring stimulus.

He proved his theory with the use of his dog – in a series of experiments; Pavlov set out to trigger a conditioned response to a previously neutral stimulus (sound of a bell). He decided to use food as the unconditioned stimulus, or the stimulus that evokes a response naturally and automatically. The sound of a bell was chosen to be the neutral stimulus. The dog was first set out to the sound of the bell and then the food was immediately presented. Then after a while only the sounds of the bell stimulated salivary secretion. Salivation in response to the bell was referred to as a conditioned reflex. This terminology was the reason that the entire method was described as conditioning.

[cf. http://www.learningandteaching.info/learning/behaviour.htm (08.04.2009);
cf. http://en.wikipedia.org/wiki/Ivan_Pavlov (08.04.2009);
cf. http://encarta.msn.com/encyclopedia_761578034/ivan_pavlov.html (08.04.2009)]

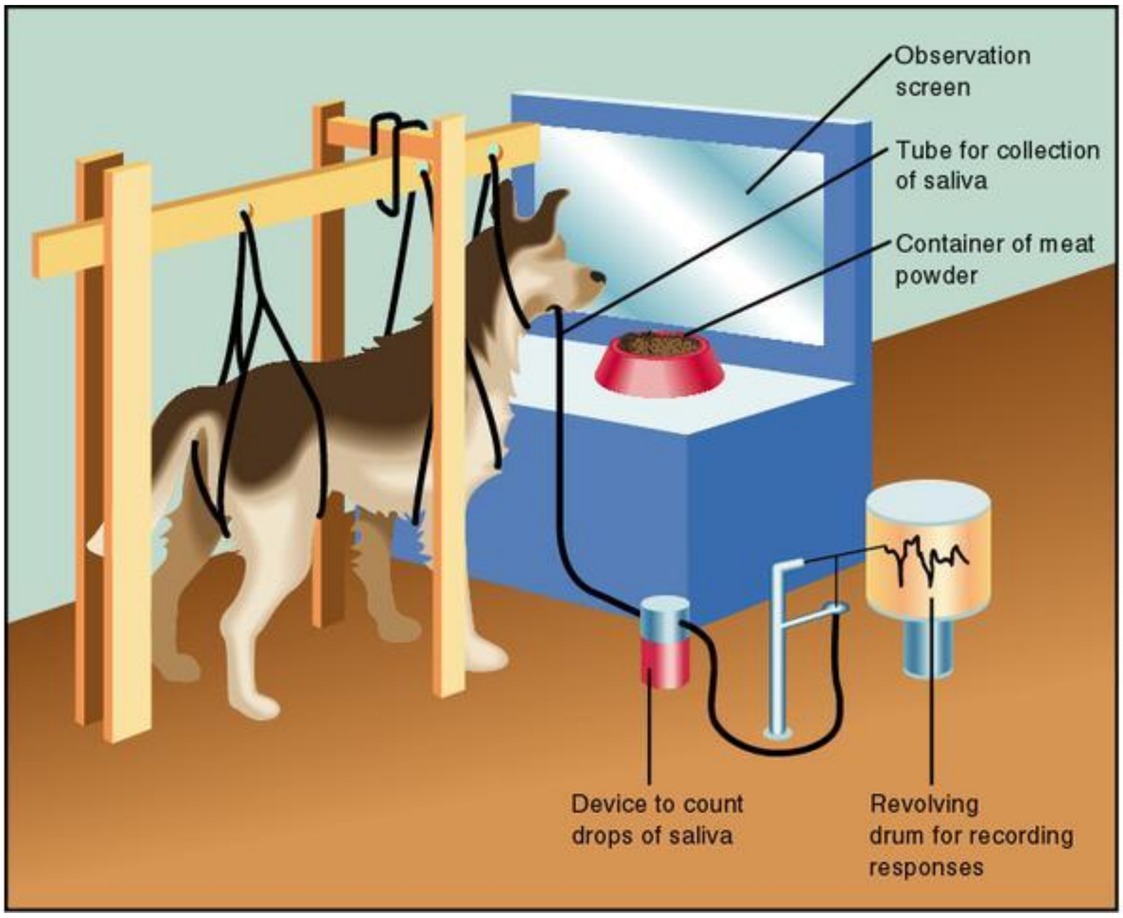

Observation screen
Tube for collection of saliva
Container of meat powder
Device to count drops of saliva
Revolving drum for recording responses

Figure 4: Pavlov's experiment
[http://www.mhhe.com/socscience/intro/ibank/ibank/0075.jpg (08.04.2009)]

Burrhus Frederic Skinner, who was the most radical representative of the behaviourism, adopted the assumptions and transferred them to the teaching experience. He saw in the theory of the conditioning the explanation of all behavior. To make this theory perfect he suggested the programmed instruction where the teacher gets partly replaced by tutorials and instruction machines. Followers of this theory assume that every learning is connected to consequences of behavior and this is the foundation of the operationalisation of learning targets.

[cf. Ludger Humbert, Didaktik der Informatik (Wiesbaden: B.G. Teubner Verlag / GWV Fachverlage GmbH, 2006), page 35]

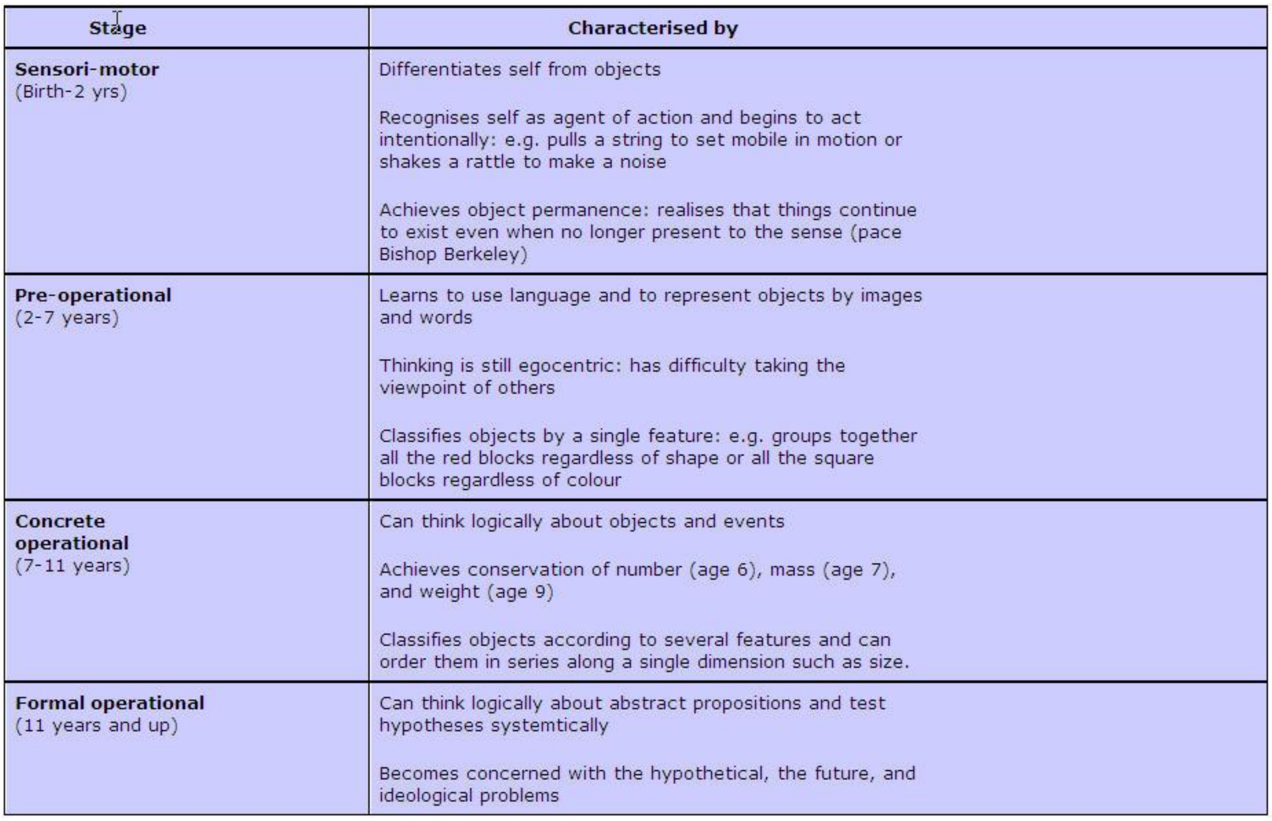
2.4.2 Cognitivism

A cross current to the theories of behaviourism in the USA and Europe emerged learning theories which concentrated more on inner mental activities of the learner. Opening the “black box” of the human mind is important and essential for understanding how people learn. Mental processes such as memory, thinking, judging, knowing, and problem-solving need to be discovered.
A response to behaviourism, people are not “programmed animals” that simply respond to environmental stimuli; people are rational beings that have to be involved in active participation in order to learn, and whose activities are a consequence of thinking. Changes in behaviour are observed, but only as an indication of what is taking place in the learner’s head. Cognitivism compares the human mind with the processes which have to be executed while running a computer program: information comes in, is being processed, and leads to certain outcomes.
[cf. Didaktik der Informatik, Wiesbaden: B.G. Teubner Verlag / GWV Fachverlage GmbH, 2006, page 35]

Piaget was a strong ambassador of the cognitivsm and developed the theory of cognitive development.

> *His view of how children's minds work and develop has been enormously influential, particularly in educational theory. His particular insight was the role of maturation (simply growing up) in children's increasing capacity to understand their world: they cannot undertake certain tasks until they are psychologically mature enough to do so. His research has spawned a great deal more, much of which has undermined the detail of his own, but like many other original investigators, his importance comes from his overall vision.*
> *He proposed that children's thinking does not develop entirely smoothly: instead, there are certain points at which it "takes off" and moves into completely new areas and capabilities. He saw these transitions as taking place at about 18 months, 7 years and 11 or 12 years. This has been taken to mean that before these ages children are not capable (no matter how bright) of understanding things in certain ways, and has been used as the basis for scheduling the school curriculum.*
> [http://www.learningandteaching.info/learning/piaget.htm (15.03.2009)]

Stages of Cognitive Development



Stage	Characterised by
Sensori-motor (Birth-2 yrs)	Differentiates self from objects Recognises self as agent of action and begins to act intentionally: e.g. pulls a string to set mobile in motion or shakes a rattle to make a noise Achieves object permanence: realises that things continue to exist even when no longer present to the sense (pace Bishop Berkeley)
Pre-operational (2-7 years)	Learns to use language and to represent objects by images and words Thinking is still egocentric: has difficulty taking the viewpoint of others Classifies objects by a single feature: e.g. groups together all the red blocks regardless of shape or all the square blocks regardless of colour
Concrete operational (7-11 years)	Can think logically about objects and events Achieves conservation of number (age 6), mass (age 7), and weight (age 9) Classifies objects according to several features and can order them in series along a single dimension such as size.
Formal operational (11 years and up)	Can think logically about abstract propositions and test hypotheses systemtically Becomes concerned with the hypothetical, the future, and ideological problems

Figure 5: Description of the stages of cognitive development
[http://www.learningandteaching.info/learning/piaget.htm (15.03.2009)]

2.5 Link between age and learning

When it comes to age and learning we will see some major differences in adult and in children learning. Compared to children and teenagers, adults have special needs and requirements as learners. For example the motivation to learn – for pupils the compulsory education pushes them into school and studying for adults there are various reasons:

1) Adults seek out learning experiences in order to deal with certain life changing events, e.g. a new job, marriage, divorce, a promotion, being fired, retiring, losing a loved one, moving to a new city.

2) The more life changing events an adult comes across, the more likely he or she is to look for new learning opportunities. Just as stress can increase the need to search for life-changing events, the motivation to find change in commitment to learning experiences increases.

3) Adults seek out learning experiences on their own which are directly related – at least in their opinions – to the life-change events that activated the seeking.

4) Adults are generally willing to take on learning experiences after, before, or even during the actual life change event. Once persuaded that the change is certainty, adults will engage in any learning that guarantees to help them cope with the alteration.

5) Adults who are motivated to seek out a learning experience do so above all because they have a use for the information or expertise being required. Learning is a means to an end, not an end itself.

[cf. 30 things we know for sure about adult learning, Ron and Susan Zemke, Innovation Abstracts Vol VI, No 8, March 9, 1984]

This proves that motivation is crucial factor for learning and that it revokes from different reasons. It is very interesting that when research is done – it seems that there is a strong link between age and learning – most of books and articles concentrate on adult learning, life long learning, which seem to be a big issue but the outcome of all these literature is that learning can happen in each age when some basic principles are respected.

3 The origin of Informatics

Counting is a term which means something for anybody these days but nevertheless it is quite a new achievement for mankind. It was developed autonomous from many different cultures in many different ways. One example is that in former day’s sky observation lead to the calculation of days which then lead the people to the assumption that counting is something godly as the gods are living in the skies – which was a good promotion for counting.

But counting had also a very practical meaning – especially when it came to money, time and geometry (which are all precursors of our understanding of mathematic). Soon people recognized that calculation is not an easy task – so they were looking for a tool, an alternative which should make it simpler to perform calculations.
Some time passed by but then came Mr. Blaise Pascal (1623 - 1662) a French mathematician, physicist and religious philosopher. In 1642 he developed the first mechanical calculator which relieved people from their burden of monotone calculation work. The scientific aspects of calculation where formed in later years.

From the Indian culture emerged the word “compute” some centuries anno Domini. The word describes calculating and counting with the help of numeral symbols.

The new art of accurate calculation was called “Algorismus” – which stands for calculations with numbers which are displayed in radix notation. So one could say that “Algorithms” is a game with signs certain rules.
[cf. Weinhart, Karl (Hrsg.) 1979: Informatik im Unterricht. Eine Handhabung. München, Wien: Oldenburg, page 23 - 25]

The launch of the first functional computer was in 1941 in Germany – Zuse Z3 – developed by Konrad Zuse. The Z3 was the first electromechanical calculator and it was fully automatic programmable, which was the forerunner for all computers to come. Also the military played a significant roll in developing computers as they became a significant roll in technologic development of weapons.
History shows that the development of computers was not to stop – the performance of computers as well as the speed and reliability was increasing and expanding.
[cf. Weinhart, Karl (Hrsg.) 1979: Informatik im Unterricht. Eine Handhabung. München, Wien: Oldenburg, page 28]

3.1 The Development of the term “Informatics”

Informatics is a new scientific discipline which has developed from issues one had with programming and computers since the last 30 years. In the forties it was not so important to lay focus on programming – the focus lay more on improving switching networks, brilliant inventions like storage devices and so on. But for most engineers the development of technical features and doing programming work was too much so many highly gifted people tried to find the way into programming.

One significant person was Mr. J. von Neumann who published his first computer program in his famous Princeton-Report in 1947 “Planning and Coding of Problems for an Electronic Computing Instrument”.

From this time till the sixties nobody was using the term Informatics and hardly anybody was talking about a new scientific discipline all was discussed under the umbrella of the term mathematic.

In the USA around 1960 people started to communicate that the term mathematic should be separated from programming as the “pure” mathematicians didn’t want to be mixed up with the programming ones. So this was the birth of the term – computer science. Some time later the trend spilled over to Great Britain, France and Germany and after the term “informatique” was common in France also Germany accepted the new word for their educational system. The terms technical informatics developed out of some discussion, which means that functional aspects are stronger in this area and practical informatics where more the programming aspects were included.
[cf. Weinhart, Karl (Hrsg.) 1979: Informatik im Unterricht. Eine Handhabung. München, Wien: Oldenburg, page 7-8]

3.2 *Informatics in Use*

Long time in history the computer was just a tool which was able to work with numbers. The user demanded real or complex numbers and sometimes integers from the machines. But thanks to their strong performances the computers can also perform very sophisticated orders like the handling of algorithms. Especially these are competences we like to deal with in Informatics.

Scientist not only gave the arithmetic abilities to the computer they also implemented data storage and command storage which gave the machine the possibility to remodel their own programs.

Nowadays a modern computer is a system which consists of hard- and software, is able to take instructions and execute them (this is also the main difference to a mechanical calculator).
Calculators simply response to immediate input, while the computer can solve more complex tasks (with the help of programs).
[cf. Weinhart, Karl (Hrsg.) 1979: Informatik im Unterricht. Eine Handhabung. München, Wien: Oldenburg, page 30 - 31]

> *Informatics studies the structure, algorithms, behaviour, and interactions of natural and artificial systems that store, process, access and communicate information. It also develops its own conceptual and theoretical foundations and utilizes foundations developed in other fields. Since the advent of computers, individuals and organizations increasingly process information digitally. This has led to the study of informatics that had computational, cognitive and social aspects, including study of the social impact of information technologies.*
> [http://en.wikipedia.org/wiki/Informatics (12.02.2009)]

3.3 Informatics in Austrian Schools

Since the 90ties a change is happening in the Austrian education system one can call it a paradigm shift to school self-governed (schulautonome) solutions. Thanks to this development many initiatives to the expansion of information technology were triggered off. When checking the curriculums of the lower grades (grammar school and secondary modern school) it is to see that none of them have informatics as a compulsory subject. Nevertheless many of these schools offer informatics classes to their students – which can happen thanks to the already mentioned self – governed solutions. But at long sight there can only exist one aim – all pupils must have a adequate, collateralize and revisable informatics knowledge base within their compulsory education.

Some factors make it hard for schools – even if they try to fulfil those aims – first the lack of hardware equipment: it is still a question of money for schools but also for parents to equip their students with the necessary hardware.

Also still a problem is the basic and advanced training for teachers – some of them have no interest in informatics so they will not train and some of them do not have the possibility to participate in those trainings. Some of them are mathematic teachers and they were given the additional task from the school administration to hold the informatics classes – which most likely leads to an overworked teacher who might not be willing to put much effort in training.
Also schools might not have the needed infrastructure, which is needed to have access to the internet.
[cf. Reiter, Anton / Scheidl, Gerhard / Strohmer, Heinz / Tittler, Lydia / Weissenböck, Martin 2003, Schulinformatik in Österreich. Erfahrungen und Beispiel aus dem Unterricht. Wien: Carl Ueberreuter, page 19 - 31]

In some schools information technology (IT) curators are appointed – their tasks are (since BGBL. II Nr. 29/1999):

- application orientated hard- and software assistance
- maintenance of technical and logistical operation abilities
- assistance in preconception and implementation of IT equipment
- network installation of applications and operating systems

- safety, virus and privacy protection
- installation of domain, mail, proxy and web servers as connection to electronic networks
- educational and organisational work
- support of teacher and students in IT related concerns
- assistance in information technology procurement
- management of a specialised library
- creation and of storage of electronic publications
- backup and restoring data
- first level support
- user administration

[cf. Lecture notes, Informationsmanagement in der Schule, Franz Gansterer TU 2008]

Also the awareness of the teacher has to be strengthened to use the computers interdisciplinary. This topic is forced strongly by the government but it too few schools lay effort in these activities. Reason might be that it requires much effort and much preparation time and unfortunately not many teachers see the positive outcome and the advantages of this way of teaching.

3.4 Austria's educational system in international comparison

The Technical University of Dresden conducted an international research regarding which level different educational systems reach compared to others in different countries in July 2007. The study summarizes all fundamental data and innovations of the last few years and reflects the development of the current situation of the educational systems of the respective country.
Furthermore the study response to already implemented and planned educational projects in the field of information and communication technology and critically tackles these results / effects on the public educational infrastructure.

The main goal of this study was to compile a structured overview of informatics education in different countries of the world. For this it was necessary to analyse the existing educational systems, the different curriculums of informatics and information and communication technology and to conduct interviews with experts in different countries.

For Austria's educational system – the results as compared to international standards were:

Austria achieved at the PISA studies 2000 not very satisfying results. Especially in the field of IT standards there was the need of improving quite swiftly.
A workgroup was built to compile specific contents of theses standards but the question is – how will these standards be implemented in Austrians schools as there dominates self-governed solutions and no standardized curriculums exist.

Another problem in Austria is the whole educational system which is much diversified and selects pupils in very young years so the chance of a common long learning is assigned. Especially this issue is still under heavily discussion but comparative studies show that countries which allow pupils to learn in common for a long time and get selected later on – show better outputs in general.

In difference to other countries Austrian students are using the computer very much in school (50%) and also regularly at home (80%).

In average 4-5 students have to share a computer in class which explains the regular use of the computers in school. This enhances the self-employed work on the computer as well as the students own initiative as the availability of a computer is given.

Research on the internet, develop presentations or to do homework can be made directly in school and has not to be done at home.
Already 68,1 % of Austria's schools have a broadband internet connection which also boots the use of the computers in school. In 68,2% of these schools an internal networking is possible – which allows internal communication, makes it easier to work in groups and supports the idea of working with the computer interdisciplinary.
[cf. http://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_informatik/smt/dil/ib/laendervergleich/Oesterreich.pdf (12.02.2009);
cf. http://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_informatik/smt/dil/ib/laendervergleich/oesterreich/oestereich (12.02.2009)]

4 Computing in school

Eric Rusten the Senior Program Officer for the Learning Technologies (LTNet) project in Brazil writes in his report “Using Computers in School”:

> *“Many school computer programs focus on teaching students about computers, training them to use basic applications such as word processors, spreadsheets, databases, and presentation programs. This approach to computer use in education is fine as long as it is just the first step in enabling students to use computers to enhance all other areas of learning. Unfortunately, many programs never go beyond these mundane training activities.*
>
> *Learning the “how” of computer use is an important and necessary step for teachers and students, but it is an insufficient reason for introducing computers to schools. Experience shows that mastery of computer skills is best achieved by using these tools to accomplish other educational activities. In other words, learning with computers is the best way to learn to use them. In addition, the “learning about” approach to computer use actually can inhibit the integration of computer technologies into other spheres of learning. Under such systems, teachers, administrators, and students can create artificial barriers between computers and core disciplines.”*
>
> [http://learnlink.aed.org/Publications/Sourcebook/chapter4/Computers_in_Schools_modelofuse.pdf, page 215 (13.02.2009)]

If teachers are ready to introduce computers to classes – there are some key issues which should be brought into consideration before hand.

Waltraud Milalkovits made her first experiences in introducing computers to her class (foundation stage) and she cites some interesting didactical principles:

- very important to mind is the individual developmental stage of the pupils
- essential is also the way pupils get taught and how far the willingness to learn of the child is developed

- the complete personality of the child must be taken into consideration: strengthening of the self-esteem as well as the acceptance of the others should be put into practice in a special way (concentration on education)
- the variety of the pupils regarding their learning aptitude and their willingness to learn is to accommodate (differentiate and enhance)
- the computer should be a technical ad to realize these values
- the computer trains the fine motor skills
- mutual assistance and the communication of the children should stand in the foreground (social aspect)
- different working- and learning techniques are to be imparted

In general she says that the children should be empowered to find their ways in the future working environments, which are strongly affected by computes. [cf. Reiter, Anton / Scheidl, Gerhard / Strohmer, Heinz / Tittler, Lydia / Weissenböck, Martin 2003, Schulinformatik in Österreich. Erfahrungen und Beispiel aus dem Unterricht. Wien: Carl Ueberreuter, page 97 - 98]

Considering all of these facts and principles there should still be on thing in a teachers mind – like Piaget (1970) asserted:

„Each time that one prematurely teaches a child something he could have discovered for himself, that child is kept from inventing it and consequently from understanding it completely. “

4.1 Computing – hype or means to an end

Computing became an integral part of school curricula in the last years, but the development of the subject matter computing has not really moved on. In the 1980ies the PCs were springing out like mushrooms and became familiar in schools, offices and private households. So there was the need of getting acquaint with them and of getting key qualifications to use these machines.

When doing research about the topic computing in school one stumbles over a lot of critical educational issues – positive or negative – so let me point out the most important ones:

- **Learner-centred education**
  Even computers exist in the classroom, teachers should not forget the students – they should engage them in experimenting, giving them tasks to construct their knowledge and encourage them to raise questions which make them understand.

- **Teachers need to enable reflective and creative learning**
  Teachers should create learning environments to help their students to gain information and experiences, also students should be able to find their own answers and obtain knowledge in their own way.

- ***Lifelong learning***
  Learning starts at the beginning of life - lasts while school education and continues to take place throughout life.

- ***Active inquiry, research, and analysis***
  Students must learn to deal with the information from the computer/internet carefully. They need to filter, structure and analyse the gained information – also they need to verify what is right and what is wrong – with these judgements they can make the right decision how to proceed with this information.

- ***Collaborative, project-based learning***
  Students must learn to work in groups and in group projects also in terms of interdisciplinary subjects. So they will have the chance to see and to solve problems in computer and in real word.

- ***Lack of technological literacy***
  In real life hardly anybody can imagine to work without computers – so not to teach children using them properly can come to a bad end in terms of their economical future.

- ***Educational/real world relevance:***

  Teachers must prepare students for real life – so they have to provide them with skills and experiences which they will need in life after school.

- ***Individualized instruction:***

  This is a big advantage that computers have – they allow students to learn in their own speed, their own level and in their own learning style. Learning programs can be designed to meet the individual need of the student. Also online group and project work often shows that even students who have some difficulties in learning participate with more effort in these learning environments.

[cf.http://learnlink.aed.org/Publications/Sourcebook/chapter4/Computers_in_Schools_modelofuse.pdf, page 209 (13.02.2009)]

In the report mentioned above the researchers also show a look in the future and show trends that will most likely come out of computers in school and Information and Communication Technologies:

- **Use of web-based/e-learning as a possible platform for instructional delivery**

  Many users are looking more and more for web-based instructions that are simple and easy to navigate. So these are gaining popularity. These platforms are also necessary to have in areas where there is not enough training for teachers and or where access to training is insufficient.

- **Appearance of educational portals within regions and countries**

  To improve teacher practice and to reduce feelings of teacher isolation - portals could be the answer. They are ever more rising and serving as an excellent resource and as perfect communication networks.

- **Expansion of distance education programs**

  In some countries there is a lack of resources for educational institutions. There distance education can be the solution to educate learners – especially in these areas there is a chance that online e-learning platforms and video based learning programs will increase and gain popularity.

- **Sharing resources online**

  When teachers and students are able to share resources online, the level of both teacher satisfaction and the learning successes of the students will be likely to rise. Such kind of learning is especially useful for schools that do not have the resources to create all of their own materials for teaching on their own.

- **Increased inter-school interactions and collaborations**

  Very successful and effective for learning are online projects amongst students from different schools and distant countries. Thanks to the broad range of projects such collaborations are accessible for students of all ages and all subject matters.

- **Use of strategic plans for upgrading, replacing, and expanding computer/ICT equipment**

  Many lessons and evaluation reports took existing ICT projects in schools as a kind of role model. They showed the need for strategies to include funding especially for the upgrading and replacement of out of date technical equipment.

- **Increased pressure on educational policymakers**

  It is a fact that teacher have to ensure that funding for training in effective technology integration in class is increased - policymakers must be prepared and informed of ICTs' potential for escalating learning in the knowledge wealth. It is a common request that educational policymakers put ICTs at the top of their agendas. As the use of computers is continuing and expanding.

[cf. http://learnlink.aed.org/Publications/Sourcebook/chapter4/Computers_in_Schools_modelofuse.pdf, page 244 (13.02.2009)]

4.2 Computer based learning

From the late 1980´s to the mid 1990´s computer assisted learning was applied as Computer Based Training (CBT) with multimedia CDs and with info-and edutainment components that embedded the subject matter. In the following years CBT was replaced by Web Based Training (WBT), which is nowadays widely known by the term e-learning.

E-learning arrangements and mixed forms with phases of tutors being present ("Blended Learning") can, with an adequate structure of the course and with enough care in the online phases, contribute to a more flexible learning process and encourage self-determined learning.

As a matter of course, school educators have the possibility of offering standardised learning aids (the so called e-content might embrace simple text elements, scripts, large scale multimedia presentations and complex learning modules) on e-learning or content management platforms. A repeatedly used argument on behalf of e-learning is the individualisation of the learning process, which has become independent of place and time.

However, far too often one fact is ignored: That learning (reading) on the screen is tremendously exhausting and instead the pupils prefer printed copies of their course records.

[Reiter, Anton 2005, 20 years of Informatics Instruction in Austrian Schools and the Use of ICT in Class, Vienna: CDA Verlags- und HandelsgesmbH, page 24]

Most education-reform initiatives assume that learning takes only place between the ages of 6 and 18, between 08.00 am and 03.00 pm – that is, when children normally are in school. But schools are just a component of a broader learning ecosystem. In the digital age, learning must and can become a daylong and lifelong experience. National education initiatives should intend to improve learning occasions not only in schools, but also at home, in community centres, workplaces and in museums. In the coming years, the Internet will open up many innovative learning opportunities and it will facilitate new types of "knowledge-building" communities. In which children (and adults) work together in projects and learn from one another around the globe.

[cf. Report, Chapter 3: Rethinking Learning in the Digital Age, Mitchel Resnick, Media Laboratory Massachusetts Institute of Technology, 2002 (http://web.media.mit.edu/~mres/ 08.04.2009)]

4.3 Advantages and Disadvantages of e-learning

Many existing e-learning environments are applications in which teachers upload study material and students can afterwards download those contents. In e-learning I see the chance that many didactic principles that could support and enhance these tools can be included.

The biggest advantage of e-learning can be written down as a statement:

Anywhere, anytime, anyone.

Thanks to the World Wide Web the learner can access e-learning offers on the whole word, 24 hours a day, seven days a week (24/7). Technical difficulties, such as access, log in problems, infrastructure, and bandwidth are sometimes still an issue when it comes to e-learning but these are getting less and less a problem.

But there are more advantages which should be stated here:

Because CBL allows the learner to set his/her individual learning speed, the times of learning – the learning process can be more individualised by the learner. If the program is designed well the learner could chose the level of difficulty – decides when and how to get help and how often he/she wants to make the exercise.
The more free the learning structures of the program the more learning expertise and self-directed motivation must come from the learner.

Immediate feedback to the learner is a basic prerequisite for self controlling the process of learning. In some programs learners should be offered a system which informs the user about his/her performance. For example which questions were solved correctly, which were not answered, for which question does the user need more attempts and so on.

This possibility of self control helps the user to work individually with the software and offers him/her a new way of learning, experiencing not only new tasks and learning context.

They also offer a way to get to know the personal learning style and the own personality when it comes to self-directed and autonomous working.
Learning improves when the learner is an active and dynamic participant in the learning software.

Also thanks to the 24/7 access the learner can learn in their own speed, which is very motivating for users of e-learning offers.

Another advantage is that the learner can access the information and content from home – which can be a cost saving issue as the learner does not need to get somewhere to visit school, training or university.

The up to date information is also a big advantage for the learner, if the teacher keeps content fresh and consistent. This is thanks to online e-learning programs an easy issue.

E-Learning accommodates different learning styles and encourages learning through a broad range of activities that relate to different learning styles.

The learner can also access the information as many times as possible and can also customise the content at his own pace, which makes it easier to study and more stress-free.

Thanks to the possibilities that e-learning is offering to the learner, they are getting encouraged to take responsibility for their own learning and this succeeds in establishing self-knowledge and self-confidence.

One disadvantage of e-learning can be that learners need to have access to a computer as well as the Internet. They also should have computer application skills especially with programs like the internet itself, word processing programs and basic knowledge about handling file transfers and so on.

The learner must be well motivated because they learn at their own responsibility. If they do not see the need to do their work at their own will, they might fall behind

especially when they do not have the traditional structures of class and the leading hand of the teacher.
Also learners might miss the direct social contact they would have in class – which also can lead in
Slow and breaking internet connection and old very slow PCs can make online learning very frustrating.
[cf. http://cai.au.edu/concept/benefit.html (10.02.2009);
cf. http://ezinearticles.com/?The-Advantages-of-eLearning&id=603386 (10.02.2009);
cf. http://www.newman.ac.uk/Students_Websites/~m.m.friel/dis.htm (10.02.2009);
cf. http://www.1stopbiztro.com/_mgxroot/page_10752.html (10.02.2009);
cf. http://www.about-elearning.com/e-learning-advantages-and-disadvantages.html (10.02.2009)]

4.4 Multimedia-based learning

In educational science teachers and students are looking for effective and efficient learning and teaching. A great part of these two issues is based on the use of media such as books or computers. A big assignment in the field of educational science is how to optimise this medial information to make them useable in lessons. The most known modes for present such information are written and spoken texts as well as static and dynamic pictures. When checking the existing medial information the result is, that they combine normally text with pictures. When we speak about multimedia learning we are speaking about learning with pictures and texts.
[cf. Stiller, Klaus Dieter 2007. Computerised Multimedia Learning. Modes of Text Presenation and Access to Text. Hamburg: Verlag Dr. Kovač, page 14]

Each day the learning programs become more and more sophisticated and programmers include video, animation and sound to attract pupils and all their individual learning styles.
When speaking about types of multimedia learning we find a huge variety of learning programs, which offer the possibility to interact with multimedia and gain knowledge out of it.

Some examples we can find in CBL (Computer Based Learning) and CBT (Computer Based Training) programs - many of them are based on the programming language Authorware. Authorware is a cross-platform, object orientated, icon-based authoring environment. It indicates hypertext and media capabilities, data measurement functions and media integration controls. Authorware can be used to create, deliver and maintain interactive applications (computer based training, education courseware,…). Authorware uses a flowline on which functional icons can be placed – it offers straight-forward scripting language and allows use of DLLs. However, many functions are already built into the icon based architecture.

[cf. Lecture notes “Computer Based Learning” Geneen Stubbs, University of Glamorgan 2001 United Kingdom]

5 Open Learning, the ultimate solution?

The concept of open learning has its origins in educational and pedagogical discussions concerning breaking up restrictions of the traditional education system. It is used in a variety of contexts and its meaning differs according to different perspectives of description. One may distinguish between an educational, an economical, a technological, a psychological, and a pedagogical-didactical perspective.

Viewed from a perspective of pedagogies and didactics, the concept of open learning refers to innovations in education aiming at delegating more initiative and responsibility concerning learning to the individual learner, and freeing the learning from restrictions as they are characteristic of the formal education approach. It is one of the central goals of pedagogic and didactic, to develop appropriate models and approaches that will meet the demands of society for modern forms of education and training, by using information technology-based learning environments.

The meaning of the concept “Open Learning” is not limited to “freedom” to arrange the learning situations as well as the personal way of learning. This also involves the fact that learning in instructional settings has to be initiated, maintained, and promoted by instruction and will not be successful without a considerable amount of interaction between the learner and the instructor. With respect to open learning in information technology-based learning systems, both the pedagogic-didactic and the psychological position holds, that open learning is not synonymous with extreme *openness* which may hinder learning activity and self-regulated learning.

Open learning systems must also provide instructional support at the request of the individual learner to enable effective self-regulated leaning. They must provide for individualization that the adaption of instruction to the learner’s needs and level of understanding.
The potential of a system to interact with the learner contributes to a high degree both to self-regulation in learning and to individualization of instruction. An optimal balance between high system regulation and extreme openness, an appropriate degree of individualization, as well as the potential of a system to make possible a high amount of learner-system interaction seems to be best suited for an effective open learning environment.
[cf. Tergan, Sigmar-Olaf Dr. / Sparkes, John J. Prof / Hitchcock, Cheryl / et al.: Open Learning and Distance Education with Computer Support, Nürnberg: BW Bildung und Wissen Verlag und Software GmbH, page 106 - 109]

Scientists say that children can not discover mathematics through the environment alone. Concepts need to be developed from practical to theoretical through the medium of language, and for this the teacher is an indispensable element. The teacher has to show things, talk about and do something with them, to introduce the language connected with these things together with the mathematical ideas involved. Procedures and techniques should be demonstrated, mathematical terms introduced and explained, before children are given tasks to do.

With lower and middle infants a foundation for the development of the concept of number mass, length, capacity, time and surface is provided but individual work can not begin effectively with any of these topics until essential words, key phrases and procedures from comparing attribute have been introduced.
General terms like “bigger”, can be developed in the appropriate circumstances to mean “longer”, “wider”, “thicker”, “taller”, etc. if we are dealing with length. In other circumstances it might be developed into “holds more” or “covers more”. Teachers can illustrate the meaning by doing things, talking about what is done and seen, and also by arranging activities for individual children to engage in.
There is a need, at the early stages especially, for more “talking time” with groups of children.
[cf. Thyer, Dennis / Maggs, John ^{2}1981: Teaching mathematics to young children. Great Britain: Holt, page ix - 1]

So teachers must address these issues when it comes to teaching mathematics, but also they have to show children that it comes down to understanding the algorithm to follow sequences of either in real life actions or when it comes to basic arithmetical operations e.g. the addition.

5.1 Algorithm in real life situations

Life could not be lived without algorithms – each action preformed in real life is a sequence and has to be followed to result in success. And here is the best point where teachers should tie in with teaching algorithm.
The didactic approach as discussed earlier is that children learn best when they can identify learning matters with their environment.

5.1.1 Logic and logical thinking

Thinking in an algorithmic structure has a lot to do with the ability to think in a logic way. There are as many ways of thinking like persons. There is no thought, kindness or insult two persons would react the same way to. A common problem of the term logic is to explain how the logical illustration of logic can be started before having defined it.

In a lot of everyday life situations the ability to think in a logical way is of great significance. This declares and points out that logical thinking works by thinking in terms of reasons and consequences, which then explains that we are speaking about sequential thinking here. Logical thinking is a very important ability and like all other proficiency, it must be trained and learned in a recurrent way.

Logical thinking is not only of great importance in our everyday lives, but is also an ability which is crucial for pupils in school. Basically all what is happening in learning situations is constrained to logical thinking. It is therefore of the highest significance that parents/teachers should teach their children this very important skill as early in life as possible.

Logical thinking is in a way the opposite of thinking with the humans' short-term memory. Short-term memory is the ability which enables one to recall the direct past. Also logical thinking allows one to keep the immediate future in view. So these two skills are closely connected and can not exist without the other. A person who has a poor short-term memory will therefore naturally have a weak skill to think logically because the one can not work without the other.

[cf. http://www.audiblox2000.com/logical-thinking.htm Logical Thinking: An Indispensable Skill by Dr Jan Strydom (MA, HED, DEd) (17.03.2009); cf. http://www.learninginfo.org/logical-thinking-2.htm (17.03.2009)]

Some examples to practise logical thinking:

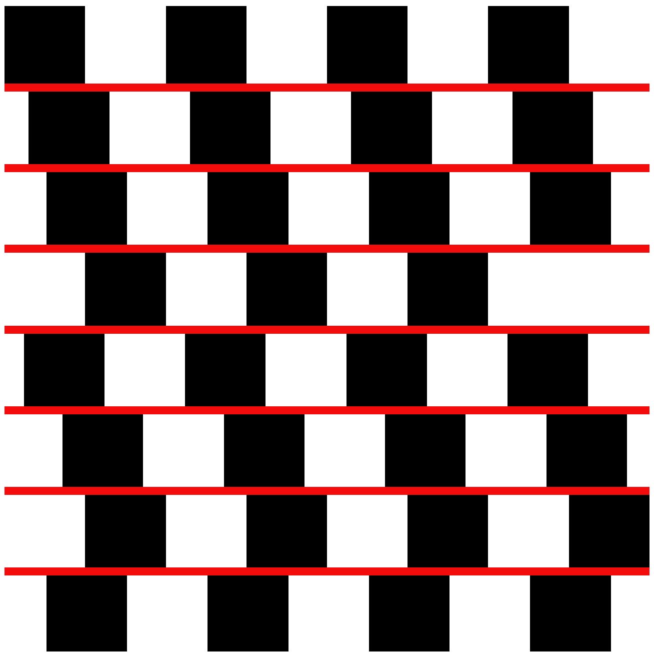


Figure 6: The parallel
[http://www.oppisworld.de/extras/illusionen/ (17.03.2009)]

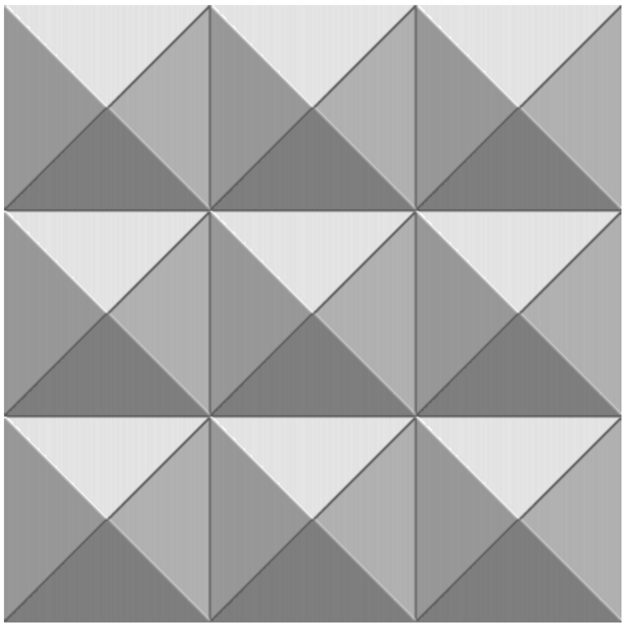


Figure 7: Up and Down

[http://www.oppisworld.de/extras/illusionen/ (17.03.2009)]

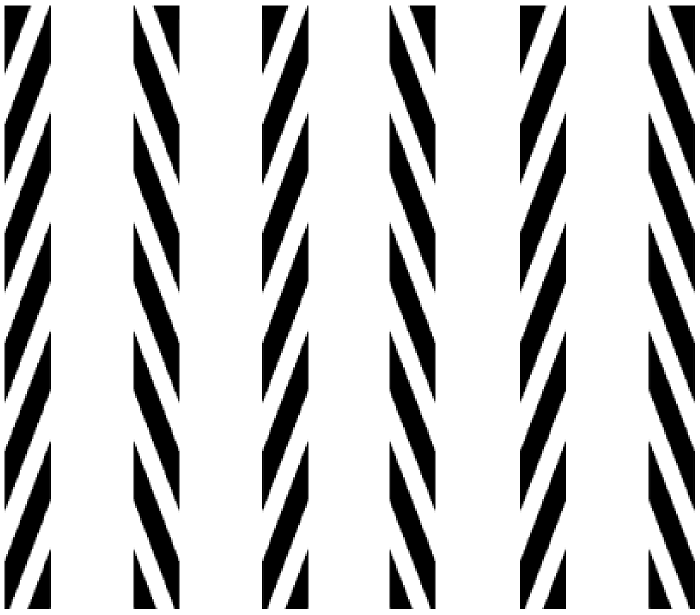


Figure 8: Columns
[http://www.oppisworld.de/extras/illusionen/ (17.03.2009)]

5.2 Algorithm in class

The foundation for all areas of informatics and its uses, is the algorithmic thinking. While solving different tasks the competence to formulate solutions in an algorithmic way should gradually be expanded. To introduce the concept of algorithm teacher should choose examples, which relate to the real life of the pupils (no pure mathematical examples in the beginning). In nearly every area of life we have to deal with instructions, regulations or recipes, which we have to or should fulfil.

For example:
The package instructions on a medicine say: If not differently prescribed from the doctor use 3-4 time 15-20 drops daily with hot sugar water. Children are just to take half of the amount.

If we look on such instructions closely, characteristics of an algorithm can be worked out. An algorithm is describing a procedure, which should be executed by a human being or a machine. This description is done with the help of special elemental steps, which have to be executed after another or next to each other. These steps must be understood and known by the executer. The description must be sufficient, must end on a certain level but on this level there must be an exact understanding.

The characterisation of an algorithm must be finite otherwise the message to the executor would last infinitely. For the execution of the algorithm there must be no time limits – there is no need that the algorithm ends in five hours but it is a must that he has to end (terminating algorithm). Furthermore the algorithm should be so general that he can be aimed at several similar problems.

In class the teacher should take care that he uses the mother tongue of the pupils for the descriptions and should work out the necessary language constructs. It must be pointed out that while executing algorithms, objects (data) are processed and that for this instructions, requests, distinctions of cases, sequences, and many more are needed.
[cf. Weinhart, Karl (Hrsg.) 1979: Informatik im Unterricht. Eine Handhabung. München, Wien: Oldenburg, page 41 - 43]

5.3 Algorithm in computing

The idea of algorithm stands behind every computer program – the program has to follow sequences in order to be executed. In computing describing an algorithm requires a notation for expressing a sequence of steps to be performed.

[cf. Skiena, Steven S. 1998: The Algorithm Design Manual. USA: Springer-Verlag, page 9]

Seymor Papert (the inventor of the Logo language) says: "*The great thing about the computer is that it can bring together in one experience the aesthetic, the personal, the analytic, the mathematical, the abstract – all these things happen in one combined integrated experience. The child which is making stuff on that screen is following an aesthetic way, is doing something socially and culturally meaningful – personally involved as well as mathematically rich. These are not separated – it is not as if the computer now does one thing and now does the other thing. Like real life situations like the babies relationship to the toys or to the mother. It has all these dimensions integrated into one experience and this is where it leads to a fundamental change in the nature of the learning."*

[Interview with Seymour Papert made in 1983 by the BBC and the (UK) Open University, http://technologyhistory.blogspot.com/2007/05/logo-and-seymour-papert.html (15.03.2009)]

5.4 Objectives of didactics in computing

Rombach´s definition for methods (greek méthodos= way, gateway) is generally "an selectable approach which in view of a certain goal is assigned to be a possibility beyond many possibilities."

[cf. Wörterbuch der Pädagogik Band 2, Heinrich Rombach, Herder Freiburg, 1977, page 280]

This definition means that there is not one right method but there are many different methods. It is up to the teacher to choose a method or a method mix and executes this on the learner. The creative task of the teacher is to find and invent approaches which motivate children to deal with actual situations. The methods have to be so pleasure orientated so that the children do not only start with the game-, work-, or research process but hang on through the process. Also have the chosen methods to fulfil a certain educational goal.

The principle of autonomy is one of the most important didactical methods! The learner must be offered various situations where they can execute their learning by doing needs. Teachers should use media which offers many learning possibilities and which accounts the children to explore, discover and experiment.

[cf. Meyer, Hilbert ^{6}1994: UnterrichtsMethoden. II: Praxisband. Frankfurt am Main: Cornelsen, page 34 - 37 & 417 - 418]

5.5 Didactical principles

Generally speaking there are a lot of didactical principles floating around – but especially in terms of mathematic (which is one of the hardest and mostly unpopular subjects) it is absolutely necessary to find the right didactical principles.

When we think about all different levels of teaching mathematics, from and educational point of view, the elementary one is the most complicated one to teach. The major reason is that teachers do not know as much as necessary about how young children gain their knowledge. A little more is known about how older children learn, but they are no longer as much dependent on the educator. It has be already observed that all people, children in particular, must be accurately grounded in real life experiences before they can value an abstraction. Children's interest differs very much when it comes to learning, it differs in the degree of learning concrete material, and in the ages at which they can grasp even simple abstractions, whether these differences are genetic or conditioned. The elementary school teacher must be committed, if he is to avoid monotonous drill, to allocate his efforts and the available time between teaching skills and communicate understanding – neither task is easy.

[cf. Kline, Morris 1977, Why The Professor Can´t Teach, Mathematics and the Dilemma of Education, New York: St. Martin´s Press, page 183 - 188]

Educational researchers emphasize the importance of linking educational materials and curricular programs to students´ existing knowledge and experiences.

> "The challenge to educators at all levels is to develop engaging assignments and curriculums that can appeal to a variety of students with different learning styles, interests, socio-cultural backgrounds, and abilities, while maintaining

the rigor of the discipline. Putting the concepts of computing in appealing contexts and building on existing competence can both reduce entry barriers and level the playing field for those with limited experience."

[http://info.scratch.mit.edu/Educators "Scratch from the National Center for Women and IT" (09.04.2009)]

5.6 *Allow pupils to invent their own methods*

Open learning is the latest catchword in schools. But by doing research I found out that open learning is already existing for ages (Maria Montessori) it just became a new hype in regards to open source software. Especially in kindergarten and elementary school we can find the concepts of open learning – later in school it fades away – most likely because it is a hard job for teachers to integrate open learning into lesson planning but that is a shame because open learning helps the children to develop their own methods and supports their understanding and problem solving strategies.

[cf. Badegruber, Bernd ^{7}1997: Offenes Lernen. 28 Schritte vom gelenkten zum offenen Lernen. Linz: Veritas, page 5 - 18]

5.7 *Trial & Error Method*

One of the very natural accesses to problem solving is the trial and error method. Eduard L. Thorndike (1874 - 1949) Psychologist who, amongst other things, studied animal intelligence and applied animal to human educational experience – characterised trial learning that the pupil has a problem or a need (initial situation) which he tries to solve (learning result) by trying different attitudes respectively by trial and error. He tries to purpose an aim by choosing a suitable behaviour out of many possible reactions till the problem is solved, which can take a long time. A trial is defined by the time which was needed to achieve the aim (or by the number of mistakes). The random and unrewarded behaviour as well as the rewarded behaviour which leads to the desired results serve the problem solving strategies of the learner. Thorndike came to the conclusion that behaviours which lead to a successful and satisfactory result are easily kept in mind of the learner.

Trial and error learning happens when the learner solves a problem (with a definite objective) by accident or through systematic behaviours – whereas the successful behaviour leads into a feeling of success. This successful behaviour will be available for the learner in the future if he uses the learned behaviours adequately.

[cf. Schwendenwein, Werner ^{6}1998 Theorie des Unterrichtens und Prüfens, WUV Studienbücher Sozialwissenschaften 4 Wien: WUV Universitätsverlag page 91 - 93]

Another or rather an additional possible assess for pupils to get into contact with algorithms, programming and so on is to allow them to play around with different applications.

6 Hands-on programming

One of the didactical principles is the hands-on / experimental approach. In my opinion this is one of the most important and natural approach. Starting in early childhood – children learn via trial and error principles. Lotte Schenk-Danzinger is writing in her book “Entwicklung Sozialisation Erziehung” that the first experimental movements from children will lead them for example to the ability to grip. In the age of the fourth month children move their fingers at eye level and follow these movements with their eyes. Here the first beginning of sensomotorical coordination can be found and out of this experimental behaviour – the child develops the grip. At the age of two the concentration of the child changes from the movement of the object to the observation of the object. The child focuses its curiosity on objects, and tries to understand by actually touching the object. This evolution accompanies the children – they continue this as they explore relations of elements till they step into the constructive phase of their life. This phase gives them the possibility to construct something – not just in mind – but with their hands. In the beginning this might be done with sand, then when the child gets older it uses building blocks – in the age of five with modelling material, via paintings and mosaic games. The first planned products emerge from the time when children explore their ego and learned to lift is identity from the surroundings, followed their first plans and let discover first deliberate objectives.

[cf. Schenk – Danzinger, Lotte ^{2}1990: Entwicklung Sozialisation Erziehung. Von der Geburt bis zur Schulfähigkeit. Stuttgart: Klett-Cotta, page 194 - 200]

These thoughts and developments during childhood were taken into account and considerably influenced the development of programming languages for children. All these principles can be found in the language LOGO made famous by Seymour Papert. This programming language was initially designed to introduce children to programming concepts, to help them to enlarge their thinking skills which then could be aligned to other contexts.

The motto which has been developed for LOG is "A language for learning." It was intended that there is a double meaning hidden behind this slogan. First Logo is a language for learning to use a computer, or to program a computer. Second LOGO is a language which helps to make informatics as simple as possible and easily understandable for the learner.

The purpose of LOGO was that it should be easy to learn, easy to use and easy to read, but it should be also a very powerful tool which also should be able to handle complex problems and situations.
[http://www.engin.umd.umich.edu/CIS/course.des/cis400/logo/logo.html (08.03.2009)]

Seymor Papert says: *"I tend more and more to avoid the word education. It makes one think of doing something to the child, educating the child. The total learning environment in which the child grows up – includes school of course includes the deliberate so called educational acts that society does on the child but the most important parts are in relationships, in play, in social forms, in art, in sensibility to aesthetic all this is part of the developing an individual and the computer enters into all of that*."
[Interview with Seymour Papert made in 1983 by the BBC and the (UK) Open University, http://technologyhistory.blogspot.com/2007/05/logo-and-seymour-papert.html (15.03.2009)]

6.1 Logo

Logo is a high-level language which conveys that there are also low-level languages, also called “machine languages” or “assembly languages”. Loosely speaking, computers can only process programs which are written in low-level languages. Consequently, programs written in a high-level language have to be processed before they can run. This additional processing takes some time, which some see as a small disadvantage of high-level languages.

But the advantages of those high-level languages and programs are huge – find some of the advantages when it comes to programs listed here:

- it is much easier to write programs in a high-level language
- less time is needed to write
- they are shorter
- they are easier to read
- they are more likely to be correct
- they are portable (they can run on different kinds of computers with few or no modifications)

Due to these advantages, almost all programs are written in high-level languages.

Low-level programs can only run on one kind of computer and have to be rewritten to run on another. Low-level languages are only used for view specific applications.
[cf. Downey, Allan B. / Gay Guido 2003, How to think like a Computer Scientist. Italy: Green Tea Press, page 2]

6.1.1 LOGO for learning

LOGO was designed for learning and at this time it was unique as no other language was designed for this purpose.
The computer offers a chance to reorganise the ways children are learning. Instead of being given knowledge by hierarchies of experts pupils can get it for themselves, building it from scratch by using the computer as a work bench.

By designing LOGO, the team tried to achieve that it should be easily accessible (easy to get into it) but it should not be a toy language. It is not like LOGO is easy – it is easy to get into but as soon as you are into it you can you can progress to the most sophisticated ideas in the world of programming.

Juggling is a complex skill but made out of simple acts. Throw a ball – catch it. This is Papert's favourite's metaphor of thinking about programming.
For him programming in the language of LOGO gives children a metaphor to think about combining the simple to make the complex. It helps them to learn to juggle, it helps them to enjoy the movements of an expert juggler and when you use have used mathematical principles as a key to enjoyable physical activities your feeling to mathematics is likely warmer, more personal and more engaged.

Seymore developed the turtle and says: *"The essential point about the turtle is its role as a transitional object.*
Transitional between the body, the self and abstract mathematical ideas. Children can identify themselves with the turtle, they can move their body to guess how to command the turtle.
So it's related to you to the body to the human and to mathematic as it can capture some extremely powerful some physical and geometrical ideas."
[Interview with Seymour Papert made in 1983 by the BBC and the (UK) Open University, http://technologyhistory.blogspot.com/2007/05/logo-and-seymour-papert.html (15.03.2009)]

LOGO has its critics as some teacher thinks it is too much freedom and children get swamped with information – so they are looking for a more structured way to use logo.
Comment of a teacher: *"Children should be given to opportunity to choose the direction in which they would like to go is very exciting. So much in the educational world is teacher directed – they structure the children's learning as they think they need to learn certain aspects before they can take the next step. There should be a place for structuring learning but especially young children should be given the experience of actually exploring and discovering things for themselves."*
[Interview with Seymour Papert made in 1983 by the BBC and the (UK) Open University, http://technologyhistory.blogspot.com/2007/05/logo-and-seymour-papert.html (15.03.2009)]

6.1.2 The Turtle

When it comes to LOGO the turtle plays a role of outstanding importance. It is not only used as a drawing instrument – it is also used as an object children can identify themselves with.

The turtle transfers the children’s commands – either to the computer screen or to real paper (when it is used as a little robot in a turtle shape). On the computer screen the turtle moves when commands are typed into the LOGO editor. LOGO has its own language and by using this language the children can get the turtle to be in motion (turtle talk). The turtle starts to move and leaves a line behind while it is going.

If the child types in the command "Forward 60" it causes the turtle to move straight ahead in a certain distance. "Forward 120" will make the turtle move in the same direction but twice as far. Soon the children will understand the idea that the numbers stand for the distance the turtle moves. These moves should be thought of as turtle steps. If the turtle should change direction and move in a different direction a command like “Left 90” should be written into the editor. The turtle keeps in its original place but turns 90 degrees to the left and is ready to start from this source.

With this little knowledge and these basic commands it should be already easy for the child to draw easy shapes like a box. Then with some more playing around and some more practical experience the child will soon be able to draw circles, spirals and even more complicated shapes and figures.

Seymour Papert gave good advice which should be followed by children who have difficulties in understanding the concept which stands behind the turtle: “*Put yourself in the place of the turtle. Imagine yourself moving in the outline of a box or a circle or a spiral or whatever it may be.”*

[cf. http://www.bfoit.org/itp/IntroCmds.html (15.03.2009)]

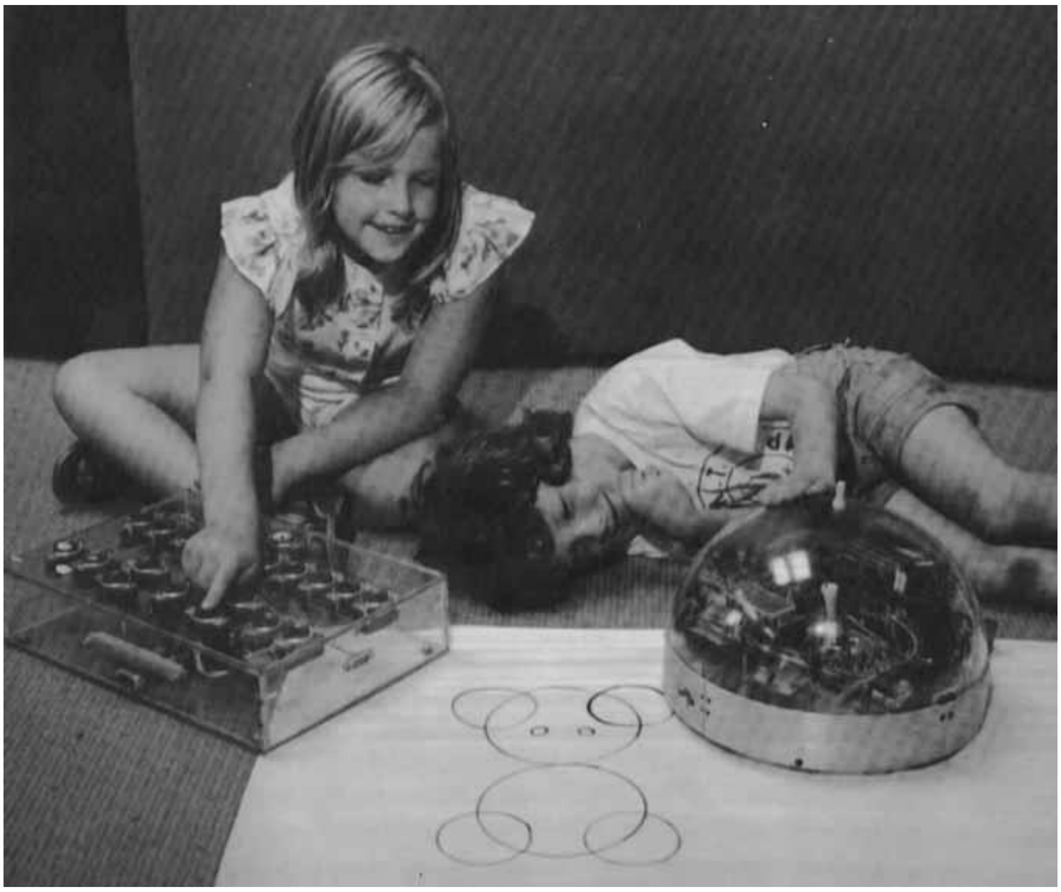


Figure 9: The original turtle from "The Children's Machine"
[http://www.bfoit.org/itp/images/mindstorms_turtle.jpg (15.03.2009)]

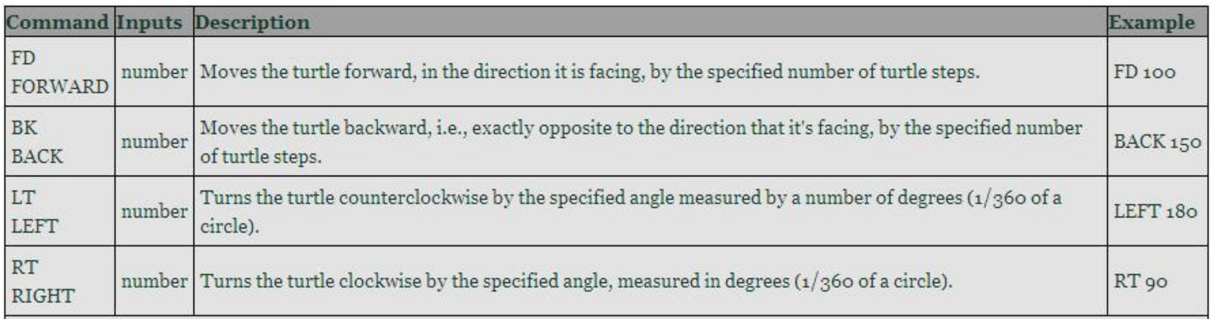
How does the turtle help children with maths?

Seymoure Papert says that the turtle does provide a mathematical environment for the children. Children experiment with the turtles they try different shapes and they try different commands – they discuss in groups how to proceed and they will find a solution in the end. So for example the children learned about the properties and angels of mathematical objects without even knowing them.
While the children are working with the turtles the only talk about mathematical things, which help them understand this area. The verbalisation in mathematics is very important. Also the playing around with mathematical problems and the talking about them makes it easier for children to get into the difficulty of the subject area.

[cf. Interview with Seymour Papert made in 1983 by the BBC and the (UK) Open University, http://technologyhistory.blogspot.com/2007/05/logo-and-seymour-papert.html (15.03.2009)]

6.1.3 Commands in logo

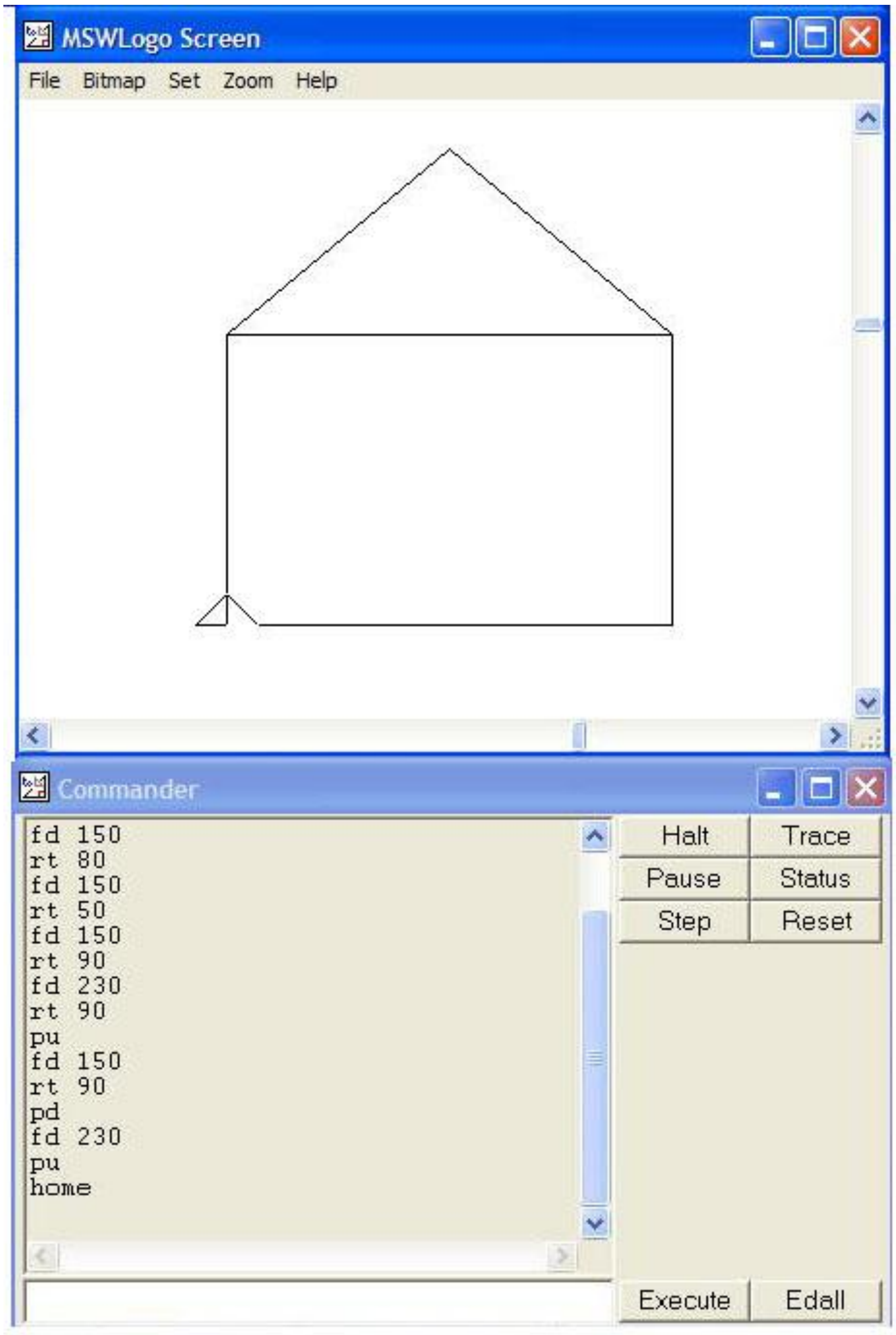
Commands in logo which let the turtles move are closely related to the way we move in real life. It's easy to try this in real life. For example teachers take the children to the outside – there one child is the turtle and the others give commands – 6 step forward – then turn left,….so they can experience and design shapes in real life.



Command	Inputs	Description	Example
FD FORWARD	number	Moves the turtle forward, in the direction it is facing, by the specified number of turtle steps.	FD 100
BK BACK	number	Moves the turtle backward, i.e., exactly opposite to the direction that it's facing, by the specified number of turtle steps.	BACK 150
LT LEFT	number	Turns the turtle counterclockwise by the specified angle measured by a number of degrees (1/360 of a circle).	LEFT 180
RT RIGHT	number	Turns the turtle clockwise by the specified angle, measured in degrees (1/360 of a circle).	RT 90

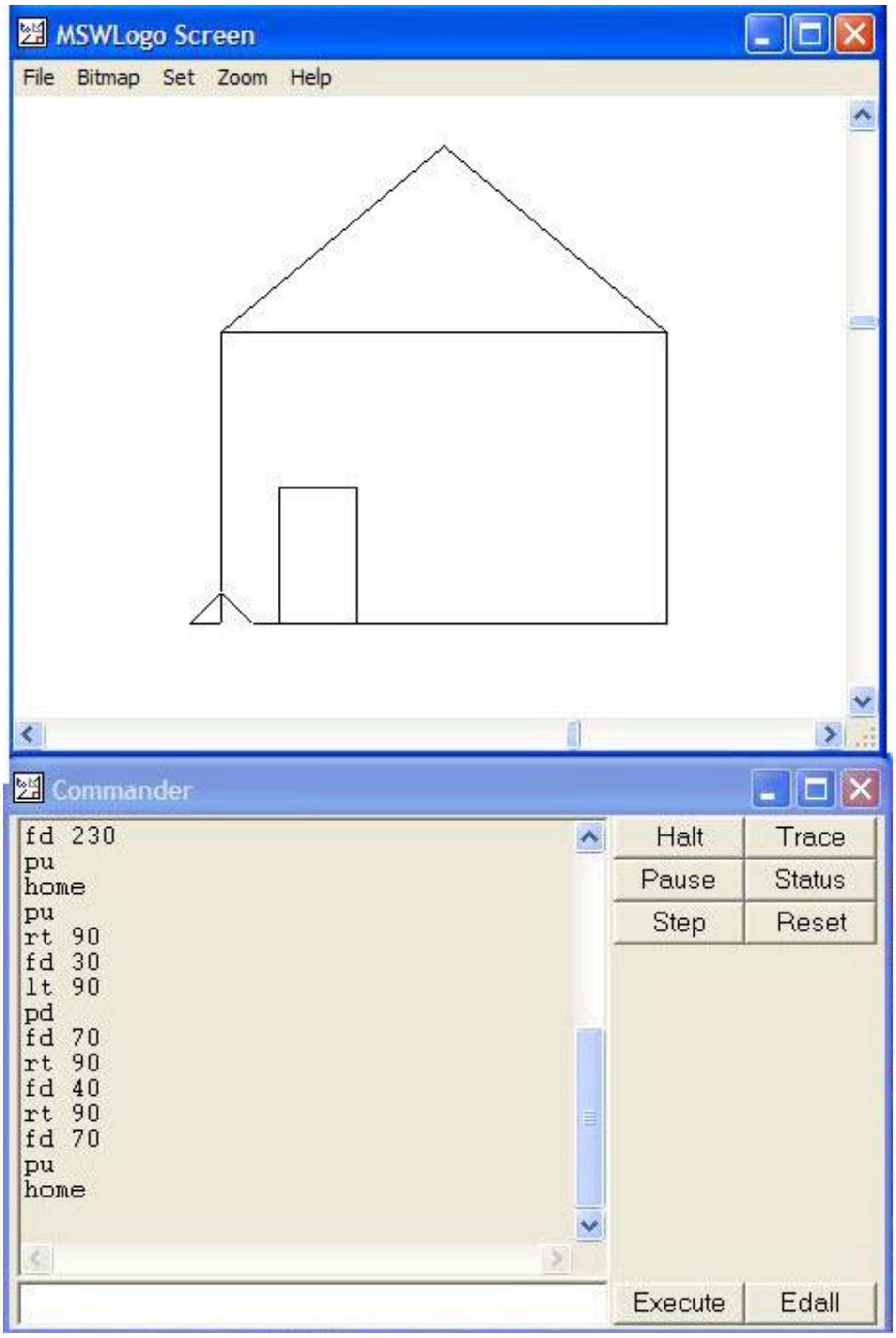
Figure 10: Basic commands in logo
[http://www.bfoit.org/itp/IntroCmds.html (15.03.2009)]

Here is an example of how easy it is – by using the basic commands – to draw a house:



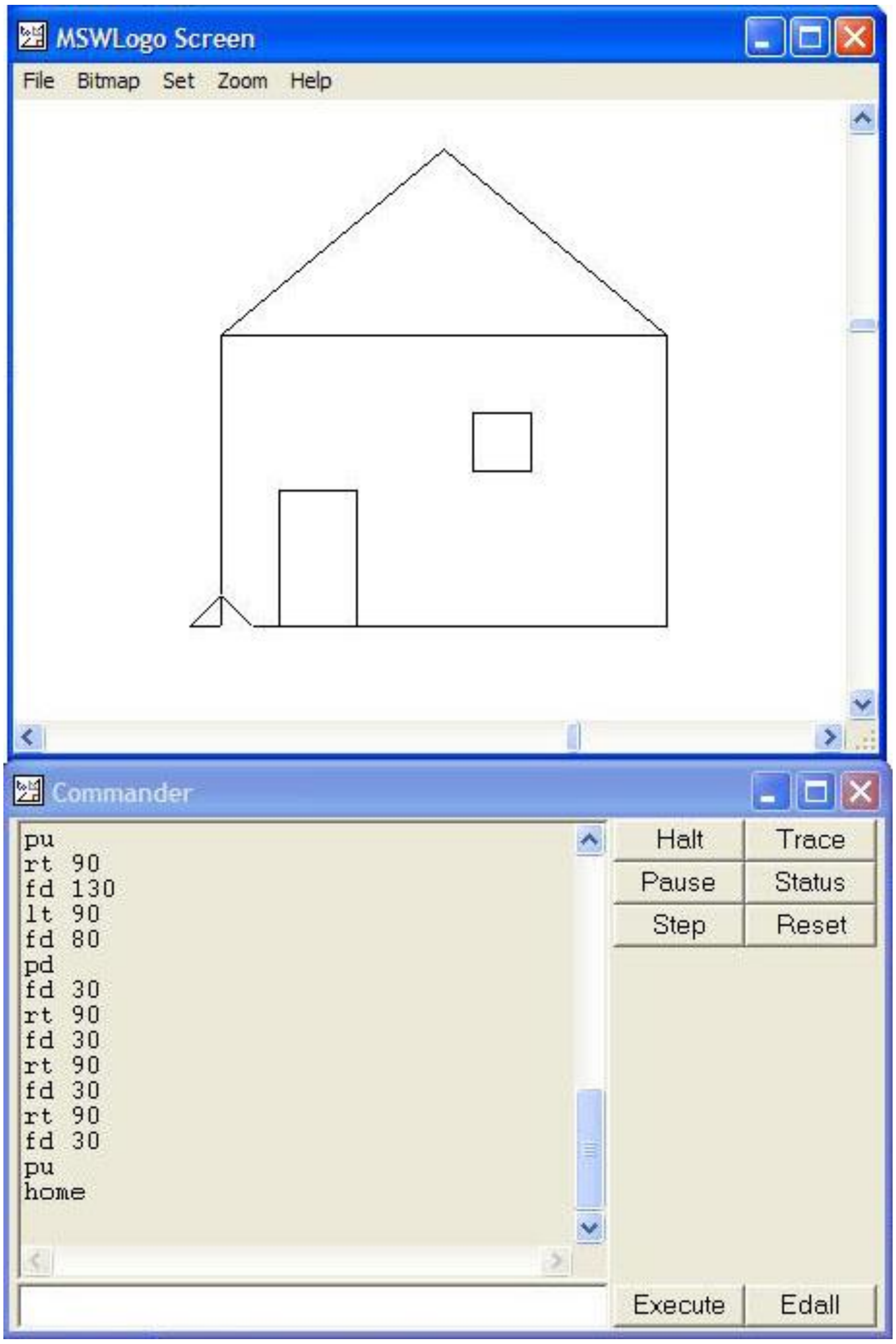
```
fd 150
rt 80
fd 150
rt 50
fd 150
rt 90
fd 230
rt 90
pu
fd 150
rt 90
pd
fd 230
pu
home
```

Figure 11: Layout of the house



```
fd 230
pu
home
pu
rt 90
fd 30
lt 90
pd
fd 70
rt 90
fd 40
rt 90
fd 70
pu
home
```

Figure 12: Adding a door to the layout of the house



```
pu
rt 90
fd 130
lt 90
fd 80
pd
fd 30
rt 90
fd 30
rt 90
fd 30
rt 90
fd 30
pu
home
```

Figure 13: Complete the house with adding a window

6.2 Python

Some time later a supposed shift of paradigms occurred in programming: the so called procedural (imperative) method (by using Pascal or LOGO) was replaced by object-orientated programming (with Java, C++ or Python) which interprets the world as a blue-print simulation of communicating objects. Apart from data, objects always imply methods and they fuse into each one another. Thus the system “school library” consists of shelves, books and people who borrow those books. Book-objects have a number of attributes, like title, author, name of the person who has borrowed the book etc. These characteristics can be manipulated by functions (operations, methods), if for example the title, the author and the name of the person who has borrowed the book is registered. Object-orientated thinking and modelling requires a process of reorientation among pupils, who often (still) have difficulties with abstract structuring.

[cf. Reiter, Anton 2005, 20 years of Informatics Instruction in Austrian Schools and the Use of ICT in Class, Vienna: CDA Verlags- und HandelsgesmbH, page 32]

> “Python for example is a dynamic object-oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days. Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code.”
>
> [http://www.python.org/ (15.03.2009)]

The main and outstanding advantages of Python are:

- no compilation steps
- a very fast edit-test-debug cycle
- debugging programs is very easy
- no segmentation faults caused by a bug or bad input
- the interpreter raises an exception when discovering errors
- by missing the exception, the interpreter prints a stack trace

- a source level debugger allows the check of local and global variables, the evaluation of arbitrary expressions, the setting of breakpoints, the stepping through the code a line at a time, and many more.
- the debugger is written in the Python language

The trick to debug a program in a rapid way is to add a small number of print statements to the source code: the fast edit-test-debug cycle makes this straightforward approach very valuable.
Because of these advantages and the increased productivity Python provides, programmers often fall in love with it.
[cf. http://www.python.org/doc/essays/blurb.html (15.03.2009)]

6.2.1 Using Python in school

Nowadays Python is used in schools very often and becomes more and more popular. While doing research for this thesis I found and interview between Frank Willson (Journalist) and Jeff Elkner who is a computer programming teacher at Yorktown High School in Arlington, Virginia, USA.

Jeff is using Python in lessons as a first teaching language. He also took along a student (Lex) who is using Python in class – let me point out some interesting questions which were asked and remarkable answers that were given.

Frank:

> *"I'd like to ask Lex a couple of questions. How do you feel about Python? I know you can't compare it to anything else, but did you find over the year that you were able to learn programming pretty well? And did you enjoy computer science?"*

Lex:

> *"I have programmed with several other languages. I started with PHP, which is a Web scripting language. Then I started playing around with Perl, and soon after that, I was introduced to Python. At first, I didn't really like Python because I thought it was too simple.*

But when I rewrote several Perl programs in Python, I saw how I could write better and more readable code. I also noticed how much more concise Python code is and how quickly I could write large programs.”

Frank:

“Jeff, what aspects of computer science are you able to teach more clearly now because of Python?”

Jeff:

“It's still a little early to tell, but we've completed in half a year what we used to do in a year. I have been able to teach procedural programming techniques, introduce students to functions, and get them to use them with a good degree of facility. We're going to be starting object-oriented programming in the next couple of weeks.”

Frank:

“So, the fact that Python is fairly abstract and hides a lot of the details allows people to get real stuff done, instead of getting bogged down in implementation?”

Jeff:

“Absolutely.

For example, string handling is a nightmare in C++, while reading a string and writing it out to file in Python is such a pleasure. I'm having so much fun with it. It's certainly making programming a lot more fun to teach.”

Frank:

“Is Python a language that you'll continue to use even after learning C++?”

Lex:

“Yes. For example, we're writing a large program called Student Portfolio [summarized by Lex as an appendix to Jeff's Python Conference proceedings], and using Python over C++ is a great advantage. I will always use Python for web applications.”

Frank:

“Jeff, one of the issues that you talked about at the Python Conference was the need for teaching materials. Can you talk a little bit about that? What does Python need in order to be useful to teachers who are now using standard languages like Pascal and C++?”

Jeff:

“They need textbooks geared toward high school students, combined with sample lesson plans and work sheets--the types of resource materials that exist for all of the other programs that we have at the high school.”

[http://oreilly.com/pub/a/oreilly/frank/elkner_0300.html (15.03.2009)]

6.2.2 Basic Python Operators

Arithmetic Operators

Assume variable a holds 10 and variable b holds 20 then:

Operator	Description	Example
+	Addition - Adds values on either side of the operator	a + b will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	a - b will give -10
*	Multiplication - Multiplies values on either side of the operator	a * b will give 200
/	Division - Divides left hand operand by right hand operand	b / a will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	b % a will give 0
**	Exponent - Performs exponential (power) calculation on operators	a**b will give 10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	9//2 is equal to 4 and 9.0//2.0 is equal to 4.0

[http://www.tutorialspoint.com/python/python_basic_operators.htm (15.03.2009)]

Comparison Operators

Assume variable a holds 10 and variable b holds 20 then:

Operator	Description	Example
==	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(a == b) is not true.
!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(a != b) is true.
<>	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) is true.

[http://www.tutorialspoint.com/python/python_basic_operators.htm (15.03.2009)]

6.3 Scratch

Originally the visual programming (mini) language Scratch was developed for students to generate 21st century skills. Its idea is to be used to develop games, animations and other programs by "clicking together" programming designs represented as building blocks.

Thanks to the look and feel of this programming language and to the "easy to use" approach - Scratch is a very appealing language and is more and more used in computer club houses, high schools and also in basic programming courses in college. Some say that Mini languages allow an insight into programming and support the teachers by teaching the way of algorithmic thinking for general computer science in an insightful, straightforward but powerful way.

Scratch distinguishes the practical learning of the fundamental CS concepts and at the same time backs up the idea of the encouragement in CS classes.

[cf. http://www.funlearning.de/ (30.03.2009)]

Scratch programming language is very close to LOGO and LEGO/LOGO, but takes benefits of the new, improved computational ideas. It extends the range of what kids can design and learn (raising the ceiling) but at the same time Scratch makes it easier to get started with programming (lowering the floor).

[cf. http://www.squeak.org/ (22.03.2009)]

6.3.1 Technical Details

Computer OS: Windows XP or later, including Vista; Mac OS X 10.4 or later

Display: 1024 x 768 or larger, thousands or millions of colors (16-bit color or greater).

Disk: At least 120 megabytes of free space to install

Memory: Most computers have enough memory to run Scratch. Older computers may run Scratch slowly.

Sound: To take advantage of sound output and input, speakers (or headphones) and a microphone are needed.

[Scratch Version 1.3 Reference Guide: http://info.scratch.mit.edu/Support (08.04.2009)]

6.3.2 Noteworthy use of Scratch

At a recent conference at MIT, students and educators came together from around the world to discuss the uses, benefits and possible improvements for Scratch.

Mitchel Resnick from Learning Research, MIT Media Lab says: "*We developed Scratch to allow kids and actually people of all ages to create their own interactive stories, games and animations online. We do not think kids should only use computers to click on websites and play online games and chat but really express their own ideas online."*

And that's what users are doing – since the launch of Scratch about a year ago there have been 350.000 downloads of the free development software and nearly 200.000 projects uploaded. The projects are uploaded to Scratch´s website where users can view then, trade suggestions or even download them and tweak the code. Users seem to like the graphical interface of Scratch.

[cf. http://info.scratch.mit.edu/Support/Videos "Kids Programming with Scratch" Video (08.04.2009)]

Lilya a 13 year old student from Russia says: *"Well I like the fact that you can actually see what you are doing while you are programming. So you can just snap blocks and there is not a piece of text that you can not understand. You can play and experiment with the blocks."* Lilya has developed more that 200 projects in Scratch - partly because the program is easier to use than other software. Jorem also a 13 year old student but from Belgium has the opinion that: *"Other programs have a steep learning curve – you have to learn everything at the same time. But with Scratch it is really easy to drag a few blocks and see how it works."*

[cf. http://info.scratch.mit.edu/Support/Videos "Kids Programming with Scratch" Video (08.04.2009)]

Scratch can be especially useful in computer classes but it is not limited to those courses. Karen Randall a "regular classroom teacher" from Minnesota explains in the interview that in her school teachers use Scratch interdisciplinary: *"I was at one of the places where Scratch was beta tested so we started using it 3 years ago and we integrated Scratch in many projects. We have done things like – in a Unit on history of creation the kids studied creation stories from around the world and then animated a creation story of their choice."*

[cf. http://info.scratch.mit.edu/Support/Videos "Kids Programming with Scratch" Video (08.04.2009)]

When children start to get into Scratch they experience many different things. On the one hand they start to get to know mathematical and computational ideas which are built into Scratch. They are experiencing that by creating programs in Scratch and by learning core computational concepts such as iteration and conditionals. On the other hand pupils start to get an understanding of important mathematical fundamental terms such as coordinates, variables, and random numbers.

The important thing is that students start to learn about these concepts in a motivating and meaningful context. When children have to learn about variables in traditional mathematical classes, the usually can create just little personal associations to the concepts and they can not link the content to their personal life experience. But when they learn for example about variables while using them in Scratch, they can apply them straight away in very meaningful ways: to control the look and feel of an animation, or to track of the score in a game they are developing.

As children are working on Scratch projects, they also start to understand the process of design. To reach the aim – which in their case is to get a program running - they have to follow a special algorithm:
Start with an idea – creation of a working prototype - experimenting with it - debugging it when things go wrong - getting feedback from others - revision and redesign.
It’s a constant spiral: first the child has to get an idea, then the start of creation of a project follows, which then leads to new ideas, which then leads to new projects and so on.
[cf. http://info.scratch.mit.edu/About_Scratch Pdf: “Learning with Scratch” (09.04.2009)]

6.3.3 Smalltalk

Smalltalk is an interpretive language that uses an intermediate compiler. Smalltalk source code is compiled into byte codes and made a part of the Smalltalk virtual image. During program execution, the Smalltalk interpreter translates byte codes into appropriate actions.

Smalltalk source code for all functionality in the language is available for viewing or modification by the user. The only exception is a group of low-level operations known as primitives, which are usually implemented in assembly language.

[cf. Pinson, Lewis J. / Wiener, Richard S. 1988 An introduction to Object-Oriented Programming and Smalltalk USA: Addison-Wesley Publishing Company Inc., page 33 - 34]

Smalltalk is not small! The size, completeness and variety of functionality provided in the Smalltalk system image is at the same time one of its strong features and one of its drawbacks. Smalltalk is not just another language, but a complete environment. Of all the object-oriented languages, Smalltalk is most true to the object-oriented paradigm. It supports all the features and properties for object-oriented problem solving. As an interpretive language it provides rapid testing of incremental changes to the image. Many problems are solved by using (and/or modifying) existing classes and methods in the system image.

[cf. Pinson, Lewis J. / Wiener, Richard S. 1988 An introduction to Object-Oriented Programming and Smalltalk USA: Addison-Wesley Publishing Company, Inc., page 33 - 34]

6.3.4 How to get started in Scratch

As already mentioned before, Scratch is a programming environment, which is based on Small Talk. The environment offers a visual representation of the commands by displaying them as colourful building blocks. Projects in Scratch are made of objects, which are called sprites. They can be changed by adapting a different costume to them. The great thing about a sprite is that it can look like a person, a car or a sort of animal or anything the pupil can imagine. Sprites can be visualized as a costume: to import sprites to Scratch – pupils can draw an image in for example paint editor, they can import an image from any external devices (USB sticks, CD-ROM,…) or they can drag in an image from a website. A sprite can be given instructions – it can be told to move, to play music or to interact with other sprites. To tell a sprite what to do – graphic blocks must be clicked together into stacks – called scripts. By double clicking on a sprit, Scratch runs the blocks from the top to the script to the bottom.

[cf. Scratch Version 1.3 Reference Guide: http://info.scratch.mit.edu/Support (08.04.2009)]

6.3.5 Programming Environment

6.3.5.1 Scratch Interface

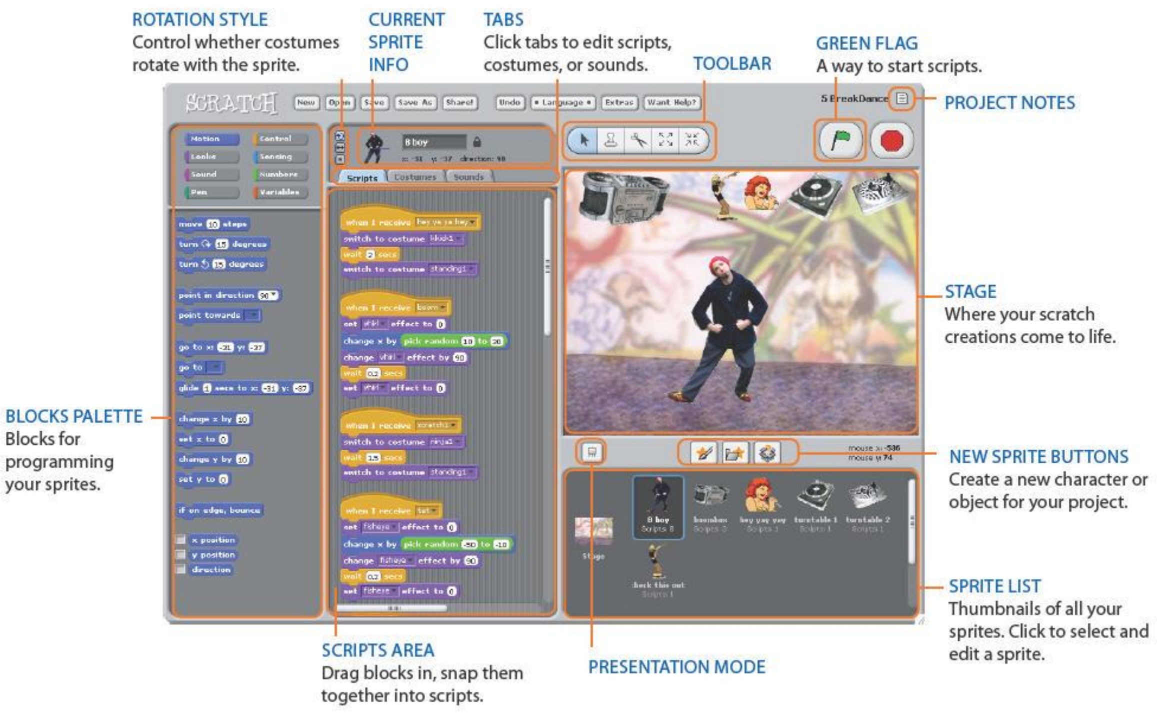
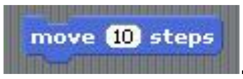
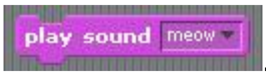
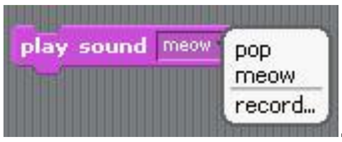

ROTATION STYLE
Control whether costumes rotate with the sprite.
CURRENT SPRITE INFO
TABS
Click tabs to edit scripts, costumes, or sounds.
TOOLBAR
GREEN FLAG
A way to start scripts.
PROJECT NOTES
STAGE
Where your scratch creations come to life.
NEW SPRITE BUTTONS
Create a new character or object for your project.
SPRITE LIST
Thumbnails of all your sprites. Click to select and edit a sprite.
PRESENTATION MODE
SCRIPTS AREA
Drag blocks in, snap them together into scripts.
BLOCKS PALETTE
Blocks for programming your sprites.

Figure 14: Scratch Interface
[Scratch Version 1.3 Reference Guide: http://info.scratch.mit.edu/Support (08.04.2009)]

6.3.5.2 Blocks Palette and Scripts Area

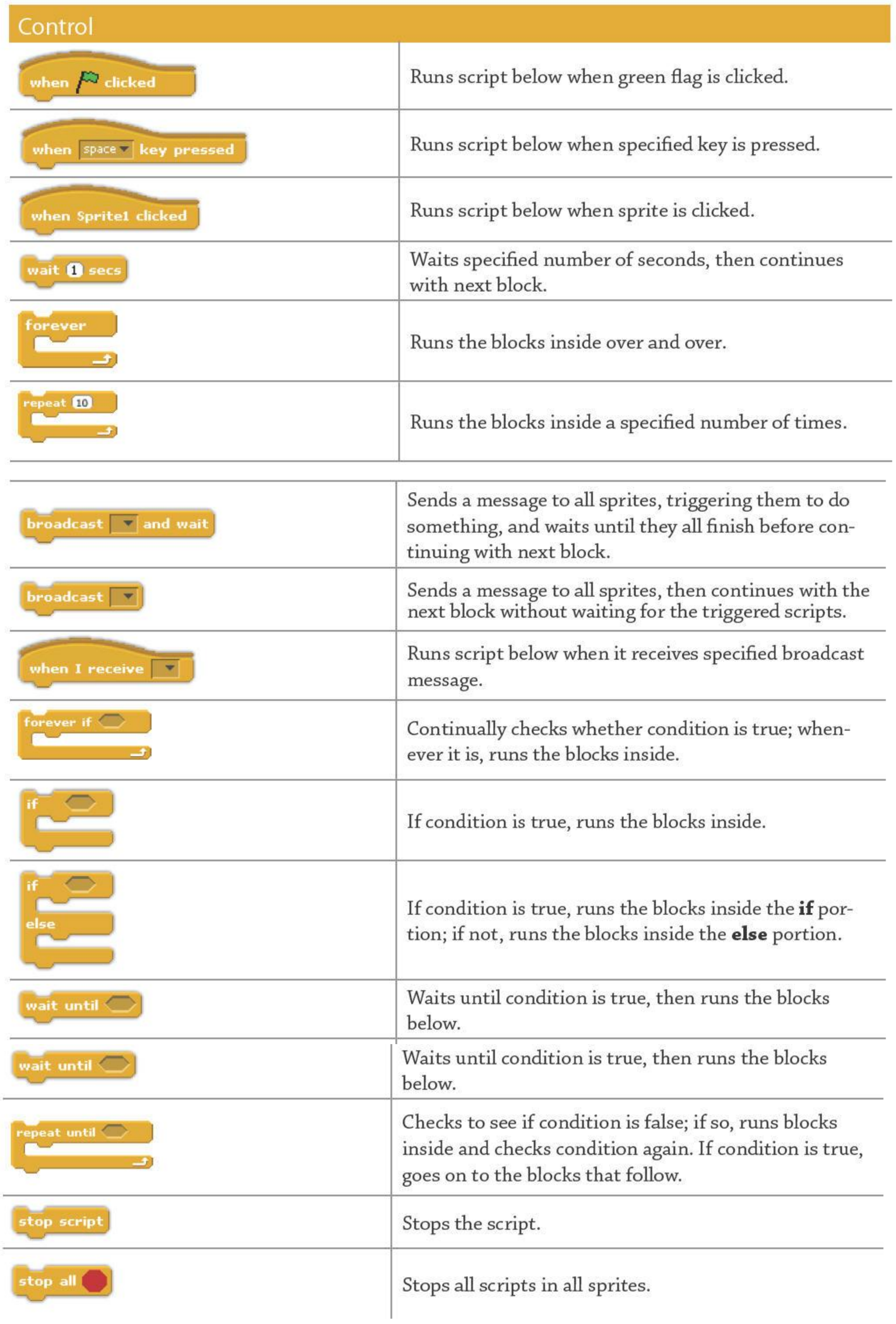
To are able to program a sprite in Scratch the blocks from the Blocks Palette should be dragged to the Scripts Area. Some blocks have white editable text fields inside, such as 

. To change this value the user just has to click inside the white area and to type a number. Some blocks have a pull down menu, such as 

. There the user just has to click on the arrow to see the menu, and then has to choose one of the options. 

.
[Scratch Version 1.3 Reference Guide: http://info.scratch.mit.edu/Support (08.04.2009)]

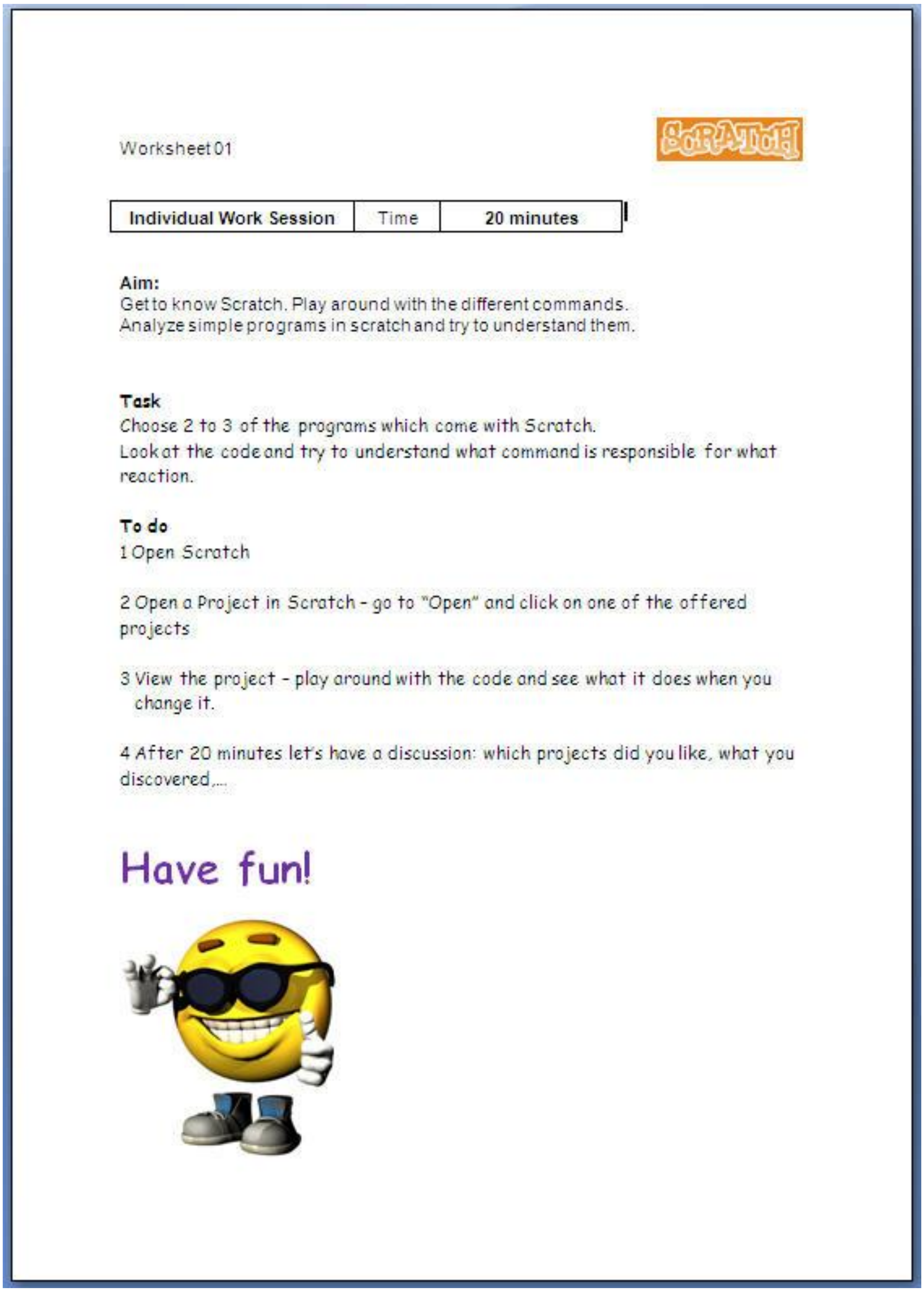
As an example – here the Control Block with the different explanations:



Control	
when clicked	Runs script below when green flag is clicked.
when space key pressed	Runs script below when specified key is pressed.
when Sprite1 clicked	Runs script below when sprite is clicked.
wait 1 secs	Waits specified number of seconds, then continues with next block.
forever	Runs the blocks inside over and over.
repeat 10	Runs the blocks inside a specified number of times.
broadcast and wait	Sends a message to all sprites, triggering them to do something, and waits until they all finish before continuing with next block.
broadcast	Sends a message to all sprites, then continues with the next block without waiting for the triggered scripts.
when I receive	Runs script below when it receives specified broadcast message.
forever if	Continually checks whether condition is true; whenever it is, runs the blocks inside.
if	If condition is true, runs the blocks inside.
if / else	If condition is true, runs the blocks inside the **if** portion; if not, runs the blocks inside the **else** portion.
wait until	Waits until condition is true, then runs the blocks below.
wait until	Waits until condition is true, then runs the blocks below.
repeat until	Checks to see if condition is false; if so, runs blocks inside and checks condition again. If condition is true, goes on to the blocks that follow.
stop script	Stops the script.
stop all	Stops all scripts in all sprites.

Figure 15: Overview control block
[Scratch Version 1.3 Reference Guide: http://info.scratch.mit.edu/Support (08.04.2009)]

There are different ways to introduce Scratch to the pupils – one way could be that they receive a worksheet and have to fulfil the task described on the sheet – see here an example of how such a worksheet can look like:



Worksheet 01

Individual Work Session	Time	20 minutes

Aim:
Get to know Scratch. Play around with the different commands.
Analyze simple programs in scratch and try to understand them.

Task
Choose 2 to 3 of the programs which come with Scratch.
Look at the code and try to understand what command is responsible for what reaction.

To do
1 Open Scratch

2 Open a Project in Scratch - go to "Open" and click on one of the offered projects

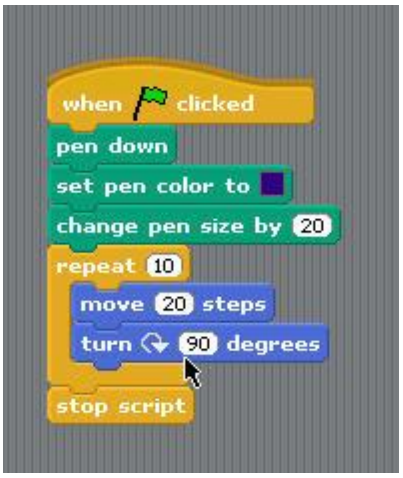
3 View the project - play around with the code and see what it does when you change it.

4 After 20 minutes let's have a discussion: which projects did you like, what you discovered,...

Have fun!

Figure 16: Worksheet example

By using all this knowledge – it is already easy for pupils to write a small example program in Scratch. Such as:

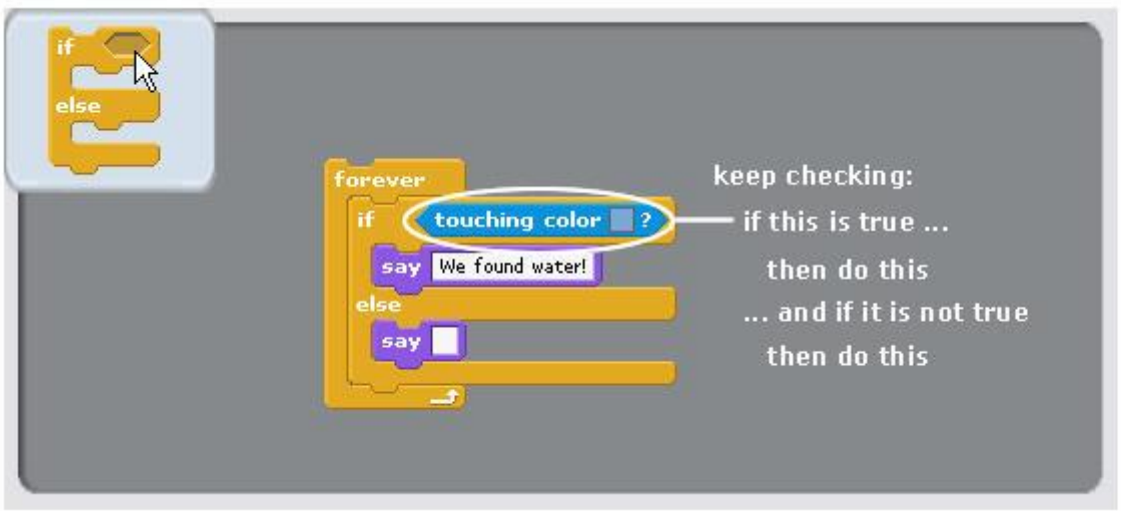


```
when [flag] clicked
pen down
set pen color to [■]
change pen size by (20)
repeat (10)
    move (20) steps
    turn ↻ (90) degrees
stop script
```

By reading the code from top to bottom it shows exactly what it will do.

Figure 17: Example for simple code sequence

When pupils are already able to follow these easy sequences – they can be introduced to using the more sophisticated control blocks. Here some examples:



```
if <>
else

forever
    if <touching color [■] ?>
        say [We found water!]
    else
        say []
```

Figure 18: Example for if statement

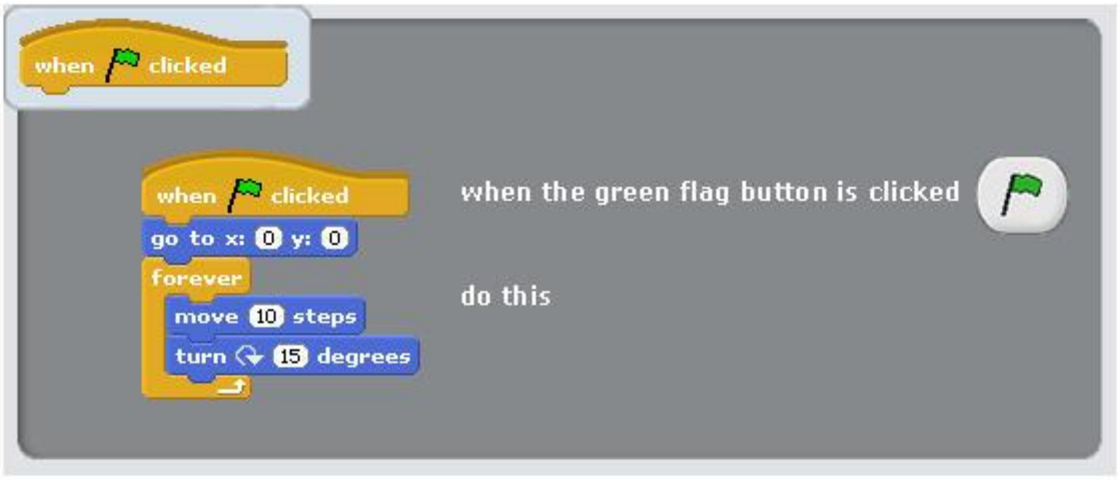


Figure 19: Example for loop

Example for creating variables:

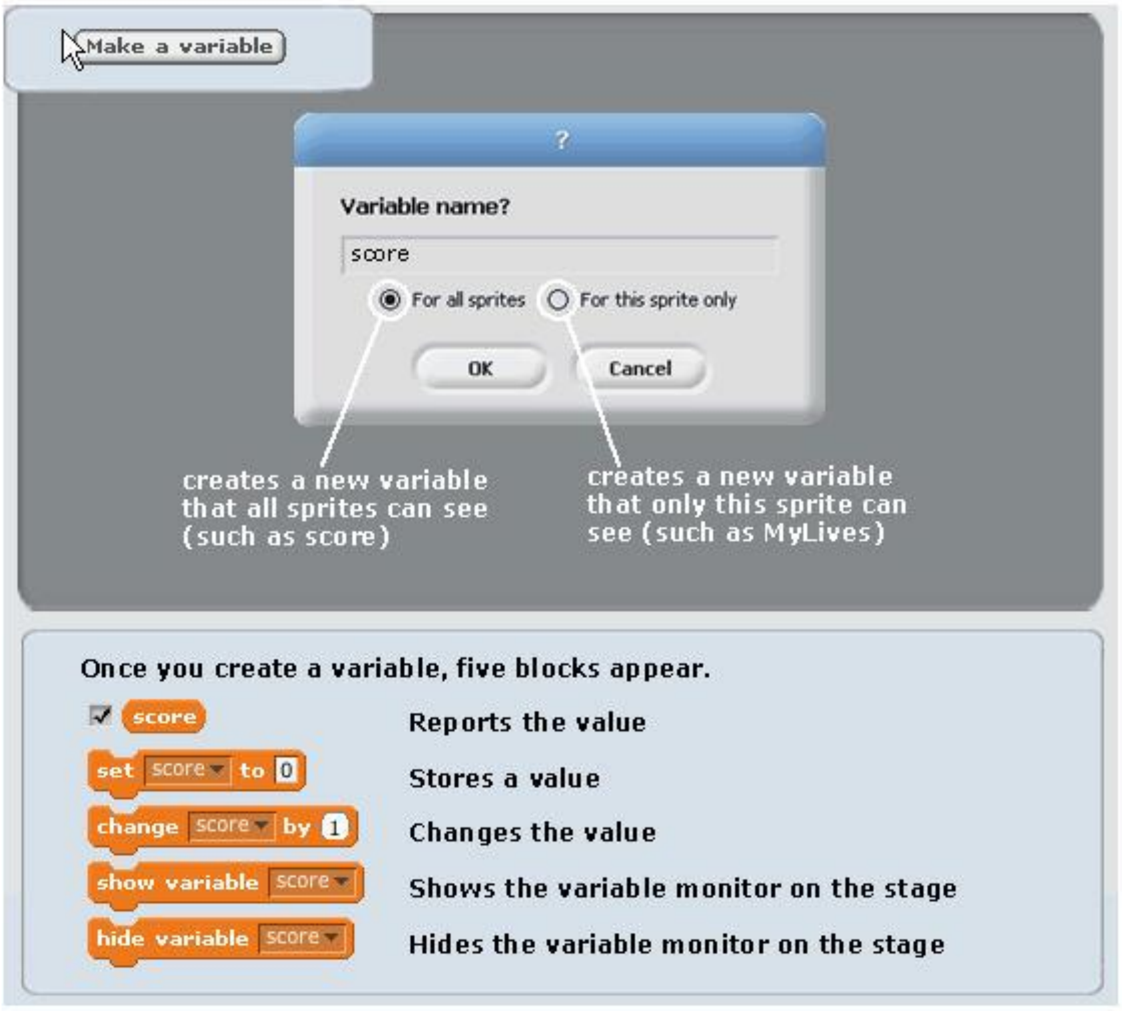
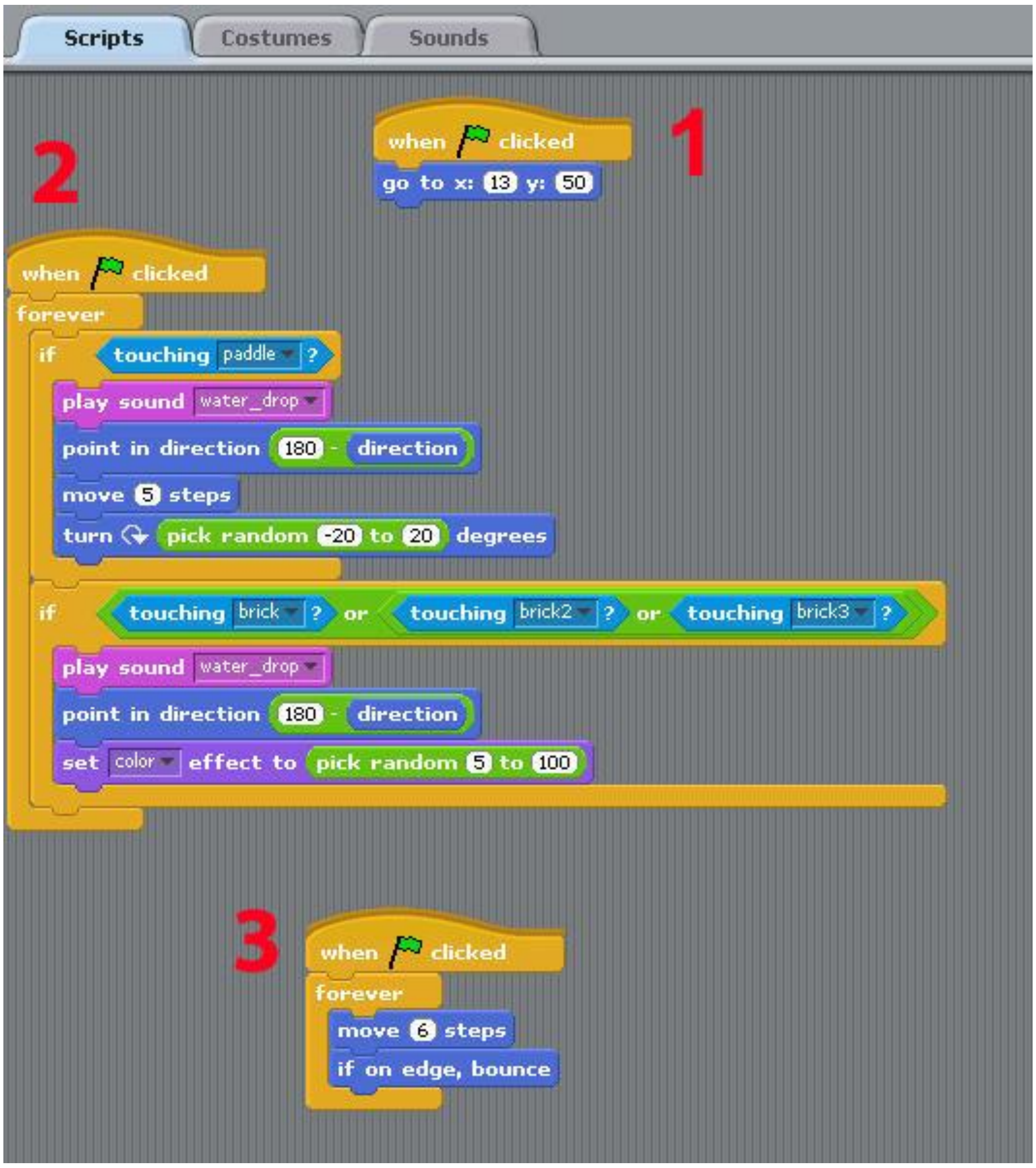

Make a variable
?
Variable name?
score
For all sprites
For this sprite only
OK
Cancel
creates a new variable that all sprites can see (such as score)
creates a new variable that only this sprite can see (such as MyLives)
Once you create a variable, five blocks appear.
score
Reports the value
set score to 0
Stores a value
change score by 1
Changes the value
show variable score
Shows the variable monitor on the stage
hide variable score
Hides the variable monitor on the stage

Figure 20: Example for creating variables
[http://info.scratch.mit.edu/Support/Help_Screens#variables (08.04.2009)]

6.3.6 An example for Scratch in use

For example I programmed a very popular game (bricks), which contains many different commands from scratch and shows that nice output can be achieved with a view commands.

6.3.6.1 The code



```
when ⚑ clicked
go to x: 13 y: 50

when ⚑ clicked
forever
    if touching paddle ?
        play sound water_drop
        point in direction 180 - direction
        move 5 steps
        turn ↻ pick random -20 to 20 degrees
    if touching brick ? or touching brick2 ? or touching brick3 ?
        play sound water_drop
        point in direction 180 - direction
        set color effect to pick random 5 to 100

when ⚑ clicked
forever
    move 6 steps
    if on edge, bounce
```

Figure 21: Example program code

Ad 1: This block of code is responsible that the “ball” is set to the coordinates x: 13 y: 50 and will “start” from there.

Ad 2: This block consists of two queries – the first one deals with touching the paddle and the second one with touching one of the three bricks. The first block of code indicates that if the ball touches the paddle, Scratch will play the sound “water drop” and the ball bounces on the paddle. Additionally it is programmed that the ball points in a defined direction, moves 5 steps and turns into a random degree angle.
The second block of code indicates that if the ball touches one of the three bricks, Scratch plays the “water drop sound”, the ball bounces back from the brick and randomly changes its colour.

Ad 3: This block of code is an infinite loop which is responsible for the movement of the ball. The ball moves 6 steps into the direction specified into the sprite info field and if it touches the edge of the stage the ball bounces back into the stage.

6.3.6.2 The result

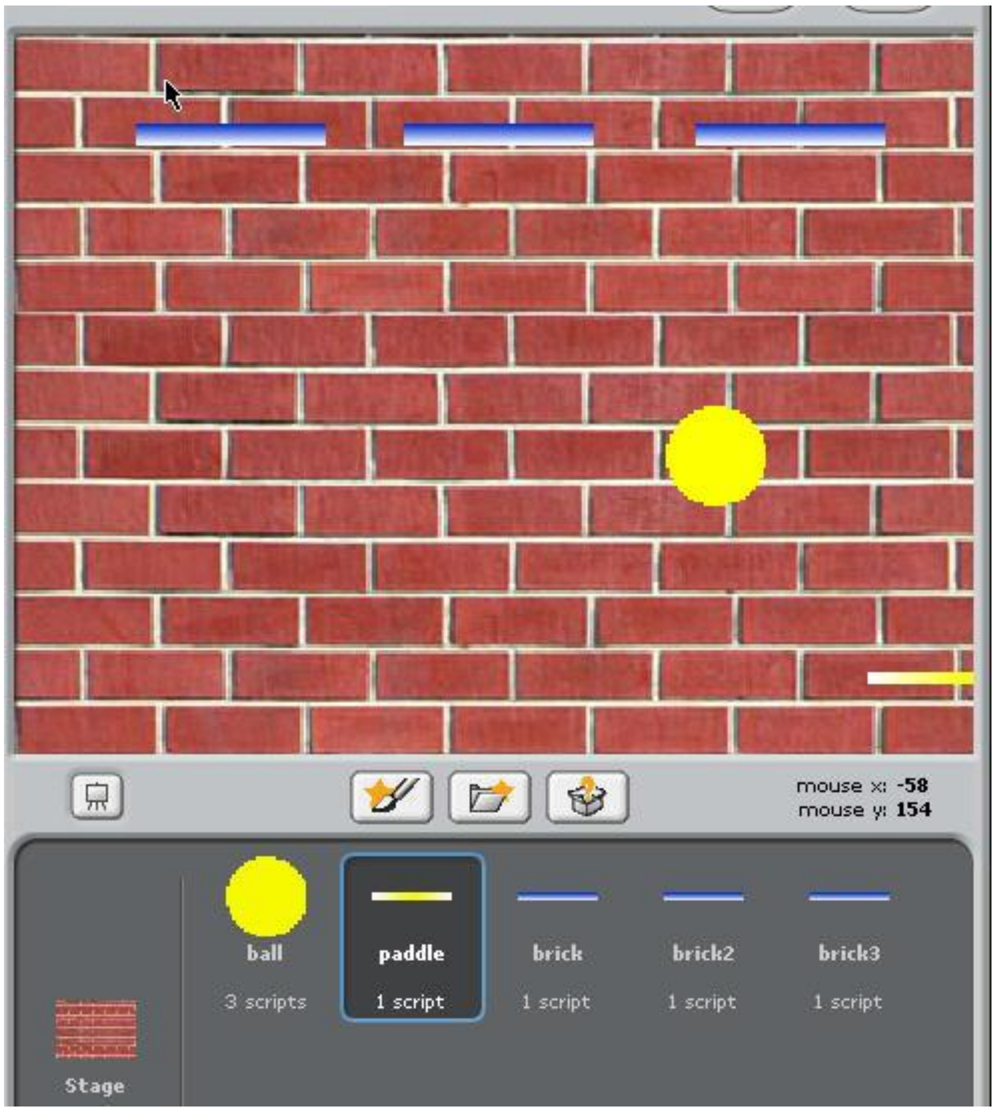


Figure 22: Example for stage window

This is the final Stage where the movie is being shown after clicking the green flag on the top. The paddle is to move with the mouse and the aim is to let the ball bounce and terminate the bricks on the top of the stage.

The actual game can be extended as wished – for example by including a counter, different levels, additional layouts and changing degrees of difficulties.

7 Annex

7.1 Abstract English

The on hand thesis deals with possible ways for pedagogues to give pupils an understanding of programming and algorithms in a simple and understandable way.

Discussions with students often show that computing lesions where they are taught programming languages are dreaded and unpopular, which might evolve from the subject matter, as well as from the didactical method teachers are using. In this thesis it was tried to show different approaches and learning theories so that it is obvious for the reader that many different methods can lead to the destination and that there is not existing a single unique way.

To understand how algorithms can be taught to pupils it is of prime importance that teachers are aware where pupils are confronted with them and how they handle these situations. Often it is just a question of awareness that all our daily acts are following an algorithmic order.

This thesis covers much ground – starting from elementary definitions of pedagogies, didactic, computing and the topic of algorithms – building on pedagogical and didactical principles of the learner and the teacher, continuing talking about the actual situation of computing in Austria's schools – the usage of computers in school up to the main topic algorithm (logic, usage, methods of teaching) and ends with the principal of hands on programming with some examples.

With the introductions of a simple but effective programming environment (Scratch), the thesis of this work is proofed.
The thesis is: *By using the right didactical methods considering different pedagogical approaches and offering pupils the right tools, they can get into computer programming more easily.*

7.2 Abstract German

Die vorliegende Arbeit befasst sich mit möglichen Wegen für Pädagogen, welche Kindern Algorithmen und Programmierung auf einfache und verständliche Art näher bringen wollen.
Aus Gesprächen mit Schülern geht immer wieder hervor, dass die Programmierfächer eher sehr gefürchtet und unbeliebt sind, was sicher an der Thematik an sich, aber auch an der didaktischen Vermittlung der Lehrinhalte durch die Lehrer liegen mag. In dieser Arbeit wurde versucht mehrere Ansätze und Lerntheorien in Betracht zu ziehen, so dass es für den Leser/die Leserin offensichtlich wird, dass verschiedenste Methoden zum Ziel führen können und es nicht einen – den einzig richtigen Weg - geben kann. Dies würden schon alleine die verschiedenen Lerntypen, welche auch in der Arbeit beschrieben sind, unmöglich machen.

Um zu verstehen wie Algorithmen den Schülern beigebracht werden können, ist es von großer Wichtigkeit sich bewusst zu machen, wo Kinder mit diesen konfrontiert werden und wie sie mit diesen umgehen. Oft ist es nur ein Bewusst machen, dass alle unsere täglichen Handlungen einem algorithmischen Ablauf folgen.

Diese Arbeit spannt einen Bogen – beginnend mit grundlegenden Definitionen der Pädagogik, Didaktik, des Faches Informatik und der des Algorithmus – aufbauend auf pädagogische und didaktische Grundlagen des Lernenden/Lehrenden über den IST Zustand der Informatik in österreichischen Schulen, den Einsatz des Computers in der Schule bis zu dem Hauptthema des Algorithmus (Logik, Einsatz, Methoden der Vermittlung) und schließt mit dem Prinzip des spielerischen Programmierens plus Beispielen dafür ab.

Mit dem Vorstellen von einer einfachen, aber effektiven Programmierumgebung (Scratch) ist die These dieser Arbeit bewiesen.

Die These lautet: *Unter Anwendung der richtigen didaktischen Methoden mit Berücksichtigung verschiedenster pädagogischer Ansätze und der Verwendung der richtigen Programmierumgebungen, können die Schüler das Programmieren leichter erlernen*.

7.3 Curriculum Vitae

EUROPEAN CURRICULUM VITAE FORMAT	
PERSONAL INFORMATION	
Name	**DANIELA BAECK**
Nationality	Austrian
Date of birth	13.05.1975
WORK EXPERIENCE	
• Dates (from – to)	Since 2006
• Name and address of employer	Marketing assistant , Mondi Packaging Paper Sales GmbH, Kelsenstrasse 7, 1032 Vienna
• Dates (from – to)	2004 - 2006
• Name and address of employer	Marketing assistant, Austrian Federal Economic Chamber, Wiedner Hauptstraße 63, 1040 Vienna
• Dates (from – to)	2002 - 2003
• Name and address of employer	Trainer, BTC Weiterbildung, Rahlgasse 3, 1060 Wien
• Dates (from – to)	1997-1999
• Name and address of employer	Governess, Boarding School , Leonhardstraße 42, 8010 Graz
• Dates (from – to)	1996-1997
• Name and address of employer	Governess, Boarding School , Langbathstraße 44, *4802 Ebensee*
EDUCATION AND TRAINING	
• Dates (from – to)	1999-2002
• Name and type of organisation providing education and training	University of Glamorgan, Pontypridd, Wales, United Kingdom

• Principal subjects/occupational skills covered	Joint Honours Degree in Mutimedia/Computing
• Title of qualification awarded	Bachelor of Science
PERSONAL SKILLS AND COMPETENCES	
MOTHER TONGUE	**GERMAN**
OTHER LANGUAGES	
	ENGLISH
• Reading skills	excellent
• Writing skills	excellent
• Verbal skills	excellent
SOCIAL SKILLS AND COMPETENCES	Thanks to my studies abroad I learned to live and work in multicultural environments. It showed/revealed/demonstrated/illustrated that I am an agreeable person; conscientious; effective in communication; efficient in problem solving; a good team player; possessing good time management skills; and the ability to apply critical thinking techniques.
TECHNICAL SKILLS AND COMPETENCES	E-Learning; Web Development & Design; Programming; Multimedia; Databases; Adobe Photoshop; Adobe Premiere; Macromedia Authorware; Macromedia Flash; Macromedia Dreamweaver; Macromedia Director; SoundForge; OS, Windows 3x, 9x, ME, NT4, 2000, XP; Microsoft Office (Word, Powerpoint, Excel, Access); Visual Basic, Java, Java Script, html, Assembler;…
OTHER SKILLS **AND COMPETENCES**	Conducting classes, training courses, co-training in the fields of: ethics, PC training, management training, presentation techniques, communication, job application training, rhetoric training. Attending several courses at the WIFI to expand on my personal education.
DRIVING LICENCE(S)	B

8 Bibliography

Books:

Badegruber, Bernd [7]1997: Offenes Lernen. 28 Schritte vom gelenkten zum offenen Lernen. Linz: Veritas

Bäumler, Claus E. 1991: Lernen mit dem Computer. Weinheim: Beltz

Bittner, Günther 1996: Kinder in die Welt, die Welt in die Kinder setzen. Eine Einführung in die pädagogische Aufgabe. Stuttgart: Kohlhammer GmbH

Bubolz, Georg 2002: Identität und Erziehung. Berlin: Cornelsen Verlag

Dammasch, Frank / Katzenbach Dieter (Hrsg.) 2004: Lernen und Lernstörungen bei Kindern und Jugendlichen. Zum besseren Verstehen von Schülern, Lehrern, Eltern und Schule. Schriften zur Psychotherapie und Psychoanalyse von Kindern und Jugendlichen. Frankfurt am Main: Brandes&Apsel

Downey, Allan B. / Gay Guido 2003, How to think like a Computer Scientist. Italy: Green Tea Press

Dreikurs, Rudolf / Soltz, Vivki [17]1986: Kinder fordern und heraus. Wie erziehen wir zeitgemäß. Stuttgart: Klett-Cotta

Felten, Michael (Hrsg.) 1999: Neuen Mythen in der Pädagogik. Warum eine gute Schule nicht nur Spaß machen kann. Ein bildungspolitisches Lesebuch. Donauwörth: Auer Verlag GmbH

Geiling, Ute (Hrsg.) 2000: Pädagogik, die Kinder stark macht. Ansätze zur Arbeit mit Kindern in Not. Opladen: Leske + Budrich

Giesecke, Hermann 1996: Das Ende der Erziehung. Neue Chancen für Familie und Schule. Stuttgart: Klett-Cotta

Gudjons, Herbert, ^{6}1999: Pädagogisches Grundwissen. Bad Heilbrunn: Klinkhardt

Harel, David, ^{2}1998: Algorithmics, The Spirit of Computing, England: Addison-Wesley Publishing Company

Hubwieser, Peter ^{2}2003: Didaktik der Informatik. Grundlagen, Konzepte, Beispiele. München: Springer

Humbert, Ludger ^{2}2006: Didaktik der Informatik. Wiesbaden: B.G. Teubner

Jarvis Peter 2001: The age of learning. Education and the knowledge society. London: Kogan Page Limited

Kern, H.P. 1987: Leichter Lernen. St. Pölten: Verlegergemeinschaft Neues Schulbuch

Kline, Morris 1977: Why The Professor Can´t Teach, Mathematics and the Dilemma of Education, New York: St. Martin´s Press

Kron, Friedrich ^{5}2008: Grundwissen der Didaktik. München: Ernst Reinhardt GmbH & Co KG

Liessmann, Konrad / Zenaty, Gerhard 1990: Vom Denken. Einführung in die Philosophie. Wien: Wilhelm Braunmüller

Meyer, Hilbert ^{6}1994: UnterrichtsMethoden. II: Praxisband. Frankfurt am Main: Cornelsen

Niederle, Charlotte ^{4}1992: Methoden des Kindergartens 2. Sonderdruck der Fachzeitschrift Unser Kinder. Linz: Landesverlag

Niemeyer, Christian 1998: Klassiker der Sozialpädagogik. Einführung in die Theoriegeschichte einer Wissenschaft. Grundlagentexte Pädagogik. Weinheim: Juventa

Pinson, Lewis J. / Wiener, Richard S. 1988 An introduction to Object-Oriented Programming and Smalltalk USA: Addison-Wesley Publishing Company, Inc.

Popp, Manfred 1974: Analyse elterlichen Erziehungsverhaltens. München: Ernst Reinhardt Verlag

Reiter, Anton / Scheidl, Gerhard / Strohmer, Heinz / Tittler, Lydia / Weissenböck, Martin 2003, Schulinformatik in Österreich. Erfahrungen und Beispiel aus dem Unterricht. Wien: Carl Ueberreuter

Reiter, Anton 2005: 20 years of Informatics Instruction in Austrian Schools and the Use of ICT in Class, Vienna: CDA Verlags- und HandelsgesmbH

Resnick , Mitchel 2002 Report, Chapter 3: Rethinking Learning in the Digital Age, Media Laboratory Massachusetts Institute of Technology, (http://web.media.mit.edu/~mres/)

Schenk – Danzinger, Lotte ^{2}1990: Entwicklung Sozialisation Erziehung. Von der Geburt bis zur Schulfähigkeit. Stuttgart: Klett-Cotta

Schubert, Sigrid / Schwill, Andreas 2004: Didaktik der Informatik. Heidelberg: Spektrum Akademischer Verlag

Schwendenwein, Werner, ^{6}1998: Theorie des Unterrichtens und Prüfens. WUV Studienbücher Sozialwissenschaften 4. Wien: WUV Universitätsverlag

Shell-Gellasch, Amy / Jardine, Dick 2005: From Calculus to Computers, Using the last 200 years of mathematics history in the classroom. USA: The Mathematical Association of America

Skiena, Steven S. 1998: The Algorithm Design Manual. USA: Springer-Verlag

Steffe P., Leslie / von Glasersfeld, Ernst / Richards, John / Cobb, Paul 1983: Children´s Couting Types, Philisophie, Theory and Application. USA: Praeger Publishers

Stiller, Klaus Dieter 2007. Computerised Multimedia Learning. Modes of Text Presenation and Access to Text. Hamburg: Verlag Dr. Kovač

Tergan, Sigmar-Olaf Dr. / Sparkes, John J. Prof / Hitchcock, Cheryl / et al.: Open Learning and Distance Education with Computer Support, Nürnberg: BW Bildung und Wissen Verlag und Software GmbH

Thyer, Dennis / Maggs, John ^{2}1981: Teaching mathematics to young children. Great Britain: Holt

Vester, Frederic 2002: Die Kunst vernetzt zu denken. Ideen und Werkzeuge für einen neuen Umgang mit Komplexität. München: Deutscher Taschenbuch Verlag GmbH & Co KG

Weinhart, Karl (Hrsg.) 1979: Informatik im Unterricht. Eine Handhabung. München, Wien: Oldenburg

Links:

http://mediaresearch.orf.at/index2.htm?international/international_nutzer.htm (30.01.2009)
http://web.media.mit.edu/~mres/ (30.01.2009)
http://www.emeraldinsight.com/fig/0370270801002.png (08.04.2009)
http://www.mftrou.com/honey-mumford.html (08.04.2009)
http://www.ic.polyu.edu.hk/oess/POSH/Student/Learn/Learning_to_learn.htm (08.04.2009)
http://www.mhhe.com/socscience/intro/ibank/ibank/0075.jpg (08.04.2009)
http://www.learningandteaching.info/learning/piaget.htm (15.03.2009)
http://www.learningandteaching.info/learning/piaget.htm (15.03.2009)
http://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_informatik/smt/dil/ib/laendervergleich/Oesterreich.pdf (12.02.2009)
http://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_informatik/smt/dil/ib/laendervergleich/oesterreich/oestereich (12.02.2009)
http://learnlink.aed.org/Publications/Sourcebook/chapter4/Computers_in_Schools_modelofuse.pdf (13.02.2009)
http://web.media.mit.edu/~mres/ (08.04.2009)
http://cai.au.edu/concept/benefit.html (10.02.2009)
http://ezinearticles.com/?The-Advantages-of-eLearning&id=603386 (10.02.2009)
http://www.newman.ac.uk/Students_Websites/~m.m.friel/dis.htm (10.02.2009)
http://www.1stopbiztro.com/_mgxroot/page_10752.html (10.02.2009)
http://www.about-elearning.com/e-learning-advantages-and-disadvantages.html (10.02.2009)
http://www.audiblox2000.com/logical-thinking.htm (17.03.2009)
http://www.learninginfo.org/logical-thinking-2.htm (17.03.2009)
http://www.oppisworld.de/extras/illusionen/ (17.03.2009)
http://technologyhistory.blogspot.com/2007/05/logo-and-seymour-papert.html (15.03.2009)
http://info.scratch.mit.edu/Educators (09.04.2009)
http://www.engin.umd.umich.edu/CIS/course.des/cis400/logo/logo.html (08.03.2009)
http://technologyhistory.blogspot.com/2007/05/logo-and-seymour-papert.html (15.03.2009)
http://www.bfoit.org/itp/IntroCmds.html (15.03.2009)
http://www.bfoit.org/itp/images/mindstorms_turtle.jpg (15.03.2009)
http://www.bfoit.org/itp/IntroCmds.html (15.03.2009)
http://www.python.org/ (15.03.2009)
http://www.python.org/doc/essays/blurb.html (15.03.2009)
http://oreilly.com/pub/a/oreilly/frank/elkner_0300.html (15.03.2009)
http://www.squeak.org/ (22.03.2009)
http://info.scratch.mit.edu/Support/Videos (08.04.2009)
http://scratch.mit.edu (09.04.2009)
http://scratch.mit.edu/files/ScratchReferenceGuide.pdf (08.04.2009)
http://info.scratch.mit.edu/Support/Help_Screens#variables (08.04.2009)

http://www.learningandteaching.info/learning/behaviour.htm (08.04.2009)
http://en.wikipedia.org/wiki/Ivan_Pavlov (08.04.2009)
http://encarta.msn.com/encyclopedia_761578034/ivan_pavlov.html (08.04.2009)

List of figures:

Figure 1: Internet User in Europe 9
Figure 2: Example Flow-Chart Diagram 13
Figure 3: Learning types circle 17
Figure 4: Pavlov's experiment 21
Figure 5: Description of the stages of cognitive developent 23
Figure 6: The parallel 43
Figure 7: Up and Down 44
Figure 8: Columns 44
Figure 9: The original turtle from "The Children's Machine" 54
Figure 10: Basic commands in logo 55
Figure 11: Layout of the house 56
Figure 12: Adding a door to the layout of the house 57
Figure 13: Complete the house with adding a window 58
Figure 14: Scratch Interface 68
Figure 15: Overview control block 69
Figure 16: Worksheet example 70
Figure 17: Example for simple code sequence 71
Figure 18: Example for if statement 71
Figure 19: Example for loop 72
Figure 20: Example for creating variables 72
Figure 21: Example program code 73
Figure 22: Example for stage window 75