X World Conference on Computers in Education
July 2-5, 2013; Toruń, Poland

Inspiration and Programming

Michael Weigend, *mw@creative-informatics.de*
Holzkamp-Gesamtschule Witten, Willy-Brandt-Str. 2, 58253 Witten, Germany
Institut für Didaktik der Mathematik und der Informatik, Fliednerstr. 21, Universität Münster, Germany

Abstract

This contribution presents a study on computer science education based on inspiration, adopting an educational pattern called "art inspired by art". The 41 participants (age 13 to 16) visited a gallery of Scratch artifacts, judged them, chose one of them as inspiration and tried to transform it to something new and unique. The study investigates the influence of previous programming experience, properties of inspiring digital artifacts and the social and cultural environment to self-directed programming projects.

Keywords

programming, inspiration, creativity, self-directed learning, Scratch.

INTRODUCTION

Why should schools teach programming? On the one hand, computational thinking (Wing 2007) and knowledge about fundamental ideas (Schwill 1993) of computer science (CS) are considered to be a facet of modern general education. Programming concepts are an important part of CS itself. Further, programming projects are a vehicle to learn (more abstract) CS concepts in a constructive way. On the other hand, programming is a good opportunity to be creative. By programming people can develop and implement technological ideas much quicker than by building physical artifacts. Programming focuses on explicating ideas. Time consuming routine activities (like sawing, sanding, drilling, welding, gluing) are unnecessary. Romeike (2007) suggests CS lesson plans focusing on creativity.

The Scratch programming environment is designed to cover both programming concepts and creativity. Its primary goal is "to nurture the development of a new generation of creative, systematic thinkers who are comfortable using programming to express their ideas." (Resnick et a. 2009, p.60). A possible source of inspiration for Scratch programming is the Scratch web site (which is integral part of the system). Millions of Scratch projects are collected in galleries. Visitors can download applications (open source code) and then change and extend them by adding new features. There exist Scratch projects that have been developed by many people. A version of the popular game Tetris has been remixed more than 450 times. "People have added color, instructions, look-ahead, and many other features." (Exploring the world of Scratch. http://info.scratch.mit.edu/node/119/1)

Mitchel Resnick sees Scratch as a modern version of a Fröbelan gift. That means, that Scratch itself – just the digital product – is assumed to initiate learning processes in (free) playful situations. In American Computer Club Houses, young people have started developing Scratch applications in their leisure time. They did not do that before Scratch was on the market. But according to Kafai et al. (2010), the technical quality of most of the programs is rather low (only in a small percentage of the Scratch projects there are variables and control structures).

These observations suggest that for many people the mere presence of Scratch and the Scratch web site is insufficient to initiate significant learning. Playful kindergarten-like situations are complex socio-cultural phenomena. Fröbel's metaphor of a kindergarten, a place where children can grow like plants in a garden, describes an environment, which supports intellectual development. The toys – the material – are just one element of this environment. There are other factors, including discussions with class mates, advice from the teacher and inspiration by other people's work. This study investigates the influence of inspiring digital artifacts to self-directed programming projects.

WHAT IS INSPIRATION?

Artists sometimes report that they have been inspired by the work of some other artist. In this context inspiration (derived from the Latin verb "inspirer": "to breathe into") means some kind of external power moving the intellect and causing activity. Being inspired (for example by a piece of art) does not imply the wish to copy or steal an idea. On the contrary the inspired person wants to create something new and unique that is still in some relation to the original – sometimes in a way that is obscure even to the artist. Inspiration by others may happen unconsciously and is sometimes completely forgotten or repressed later. This seems to be natural since a creator wants to be the true origin of his idea. Personal causation is a very strong motive for activity (Reich & Zautra, 1981).

The German Expressionist Ernst Ludwig Kirchner created a painting depicting Erich Heckel, walking along a park way. He adopted the composition of Edvard Munch's painting "Harry Graf Kessler". Kirchner had seen this picture earlier but, but he strictly denied being inspired by it (Weikop 2002, p. 415).

In other cases inspiration is explicit and even part of the work itself. Andy Warhol's painting "Converse Extra Special Value" (1985 or 1986), depicting a trainer and the number 12 explicitly refers to Leonardo da Vinci's famous masterpiece "The Last Supper", depicting Jesus and the twelve Disciples. The relation is subtle but explicit.

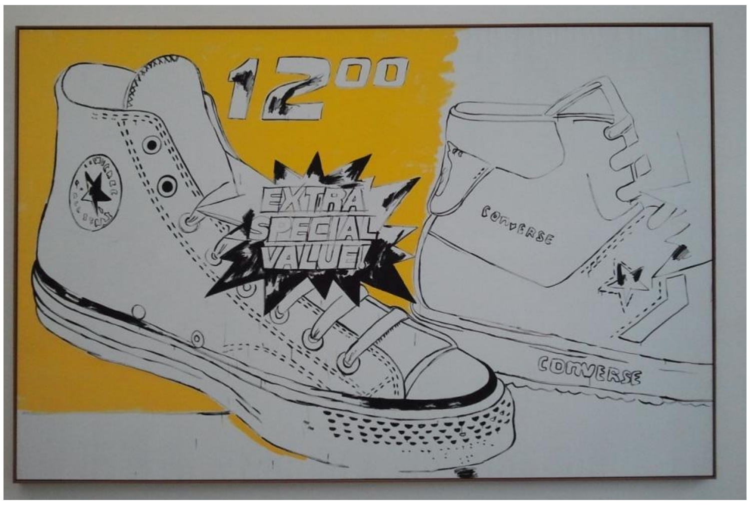


Figure 1: Andy Warhol: "Converse Extra Special Value" (1985 or 1986) Sammlung moderne Kunst, Pinakothek der Moderne, Munich, Inv. GV 124, photo: Michael Weigend 2012.

STRUCTURAL AND FUNCTIONAL INSPIRATION IN CS EDUCATION

There are media that were intentionally designed to inspire students to program: illustrations in textbook, posters depicting flow charts in a class room, movies or cards with concise explanations (like Scratch cards) just lying around on tables waiting to be looked at.

Regarding the content of such material I distinguish between (1) structure oriented and (2) function oriented inspiration. The terms “structure” and “function” refer to the view that technical artifacts have a “dual nature” (Kroes 1998; Schulte 2008). The function is the goal or purpose the artifact is designed for. For example, the function of a clock is to tell time. The structure describes the internal mechanics, the implementation. For example, mechanical clocks are made of gears and springs etc. moving hands over a dial.

(1) A media artifact focusing on structure explains some programming technique. The intention is that the recipient understands and appreciates the presented technique and uses it in a project for some purpose. But this design purpose (function) is not part of the inspiration. Basically, collections of design patterns (Gamma et. al. 1995) and programming “cook books” intend structure oriented inspiration. Other examples are LEGO blocks. Children tinkering around with LEGO – not following instructions – see the blocks, imagine their potential and search for “problems” they can solve with them. A very long block might inspire to build a bridge. Scratch blocks – representing commands – partly inspire in a similar way. Some blocks (like sound-playing blocks) have some immediate perceivable effect, when you click on them. Others (like if or repeat) do not.

(2) Media focusing on function are designed to raise interest in the application of software technology. They present some aspect of real (or fictional) life, point out its relevance and thus motivate to create artifacts that are related to that. For example Martin et al. (2000) suggest lesson plans for LEGO robotics that start with a story that the children listen to first.

Both types of inspiration finally lead to the same activity. The inspired person creates some design purpose (function) and creates an artifact (structure) implementing the desired functionality.

PROGRAMMING INSPIRED BY PROGRAMMING

In 2012 the Manchester Art Gallery exhibited pieces of art that were inspired by pictures from the gallery’s collection.

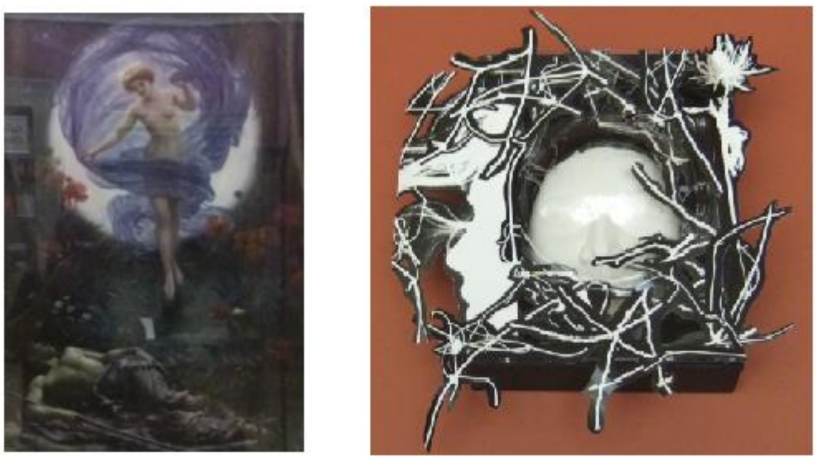


Figure 2: Poynter’s painting (left hand side) inspired an artist to create a sculpture (right hand side), Manchester Art Gallery 2012, photos: Michael Weigend 2012.

This (educational) concept is called “art inspired by art”. I will now discuss a lesson pattern, which I call “programming inspired by programming”. The Scratch web site is regarded as a museum. The students visit a gallery, which was designed especially for this study, look at Scratch applications (“exhibits”), judge them and take them as sources of inspiration. In contrast to “free play” in the Computer Club Houses or Kindergartens, the students have to follow this dramaturgy of three stages, which I am going to explain in more detail in the following section.

DESIGN OF THE STUDY

41 pupils (28 boys and 13 girls, average age 13.7) participated in this study, which I performed in 2012 at a high school in Witten, Germany. Three groups (grade 8 and grade 10) and two teachers were involved. The aim was to find out the students' perspectives on inspiration by Scratch artifacts. The sequence of six lessons consisted of three stages

Stage 1: Introduction to Scratch

The students started with a general introduction to Scratch, following instructions in a manual (90 min). They took the role of apprentices learning given content. Each class had been divided into two groups using two different manuals, each consisting of two parts. Part one was identical in both versions. Following the examples, the students learned basic concepts: blocks, sprite, script, editing a costume, running, storing and loading an application. Part two was different. Manual A focused on controlling movements through the keyboard (when …key pressed, move … steps). Manual B focused on dialogues between characters (say...for ... seconds, wait ... seconds). A few students (around 20%), who were very quick, did both projects of part two.

Stage 2: Visiting a Gallery – Inspiration

In Stage 2 the students visited a gallery on the Scratch web site ("Magical action"), which had been designed and created for this study. It contained eight different applications.

Three things were considered in the design of the applications:

- They are very small, almost minimal, and contain just one or two sprites and a few short scripts, just enough code to implement a functional idea.
- The programs are completely comprehensible for the students. Most of the commands were practiced in Stage 1.
- All applications have an obvious purpose (like simulation or learning game).

Four applications focus on dialogue:

(D1) Two dogs appear, talk to each other and tell a joke.
(D2) An animation visualizing the abstract idea of a "true encounter".
(D3) An interactive language learning game. On the screen there is a command written in a foreign language ("Pick the coconut!"). When the user does the right thing (thus proving that she or he has understood the text), something happens.
(D4) Simulation of a chemical reaction.

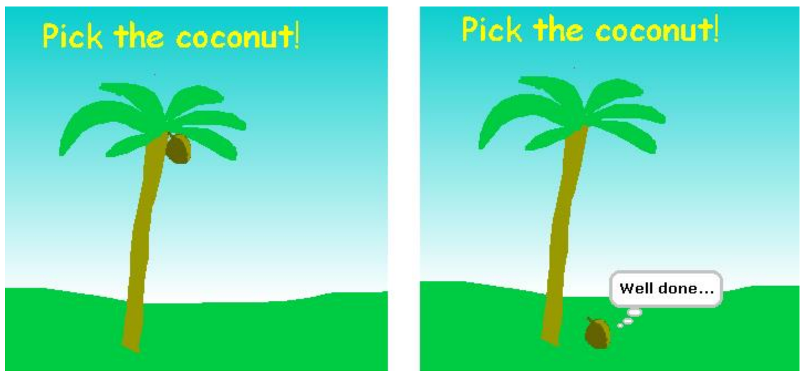


Figure 3: Screenshots from the learning game "Coconut"

The other four applications focus on controlling motion:

(M1) The player navigates a plane to a city on a map.
(M2) A rocket has to be directed to hit a moving meteor.

(M3) Simulation of a car race. The player controls a car and tries to keep it on the track.
(M4) The player has to land a spaceship on a planet using the arrow keys.

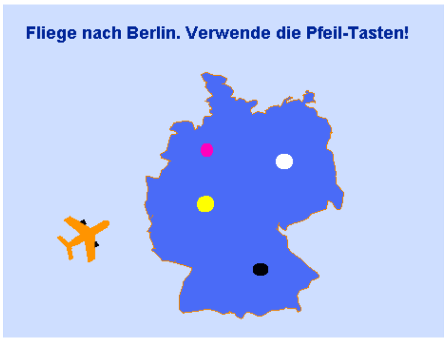


Figure 4: Screenshot from an interactive application simulating a flight.

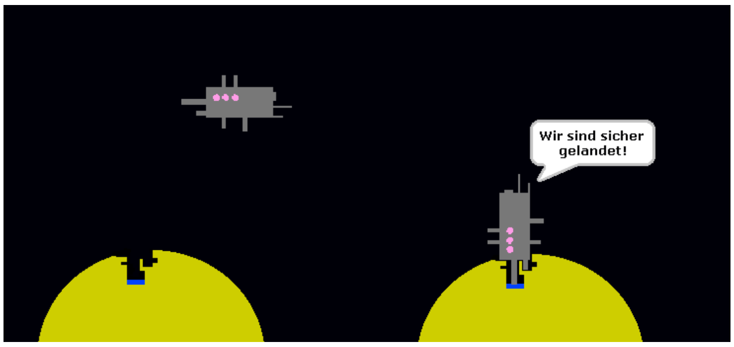


Figure 5: Screenshots from an interactive application simulating the landing of a space ship

The students were asked to write comments to a few Scratch applications on the Scratch web site. They took the role of an art critic. What are good features of this application? What could be improved?

Stage 3: Creation

At the beginning of this stage the students imported one application from the gallery as a starting point for their own project. They had complete freedom in the design. But they were encouraged to develop their own ideas and change the original as much as possible. They got approximately 90 min (two lessons) for this. Some of them worked alone, others in teams of two. There was a “clubhouse atmosphere” in the classroom. The students could show their work to class mates and discuss it with them. Teams, who had finished their project before the time was up, were encouraged to think about additional features and to continue. Thus the development process consisted of short iterations within a limited time interval, similar to Extreme Programming (Beck 2000).

At the end the students reflected on their development processes by answering a questionnaire.

RESULTS

Comments in Stage 2

Although the students were encouraged to write a comment to each program, only 172 comments were posted. Usually they were just a few words. All statements in the comments were classified according to certain categories. Most comments were general judgments like “cool” or “boring” (115 out of 172 comments), most of them positive (70). About 10% of the comments included specific functional criticisms or suggestions for improvement like “the meteors should explode, when hit”. Another

10% expressed incomprehensibility (e.g. “I do not understand this “, “this does not make sense”). Remarks on the visual design tended to be rather specific (referring to a certain detail like “the dogs are cute”) rather than being general judgments (“looks nice”).

Choice of the Artifact

Most students (74%) chose the car race simulation as a starting point for their own project. This decision seems to be completely independent from the programming experience in Stage 1. In an introductory project of manual A, 26 students had learned how to use keyboard-events for controlling the movement. This is a crucial technique in the car race simulation. 15 did not have this experience. Nevertheless the percentage of those who took the car race was the same.

However, the reasons the students gave for their decision show that some knowledge about the function and structure of a digital artifact was relevant.

Some items (R4, R7, R8) are structure-oriented. For example, you need to consider the internal structure of an application to qualify it as an “easy project” (R8, 29%). Someone, who wants to learn how to create a certain application, is interested in its structure (R7, 27%).

Other statements (R1, R2, R3, R5, and R6) are more related to the (visible) function of the artifact. For example, when a student claims to like the “idea“ of a digital artifact (R3, 51%), she or he is referring to its function and not to the (invisible) structure. 61% stated that they immediately had an idea what to improve. This was the most often stated reason. Again, in this context it can be assumed that the “improvement” is related to the functionality and not the internal structure.

Only 15% stated as a reason for choosing the project that they have developed a similar program in Stage 1 (this was correct in all six cases).

Table 1: Stated reasons for selecting a specific Scratch application as starting point.

	Reason	Girls (n=13)	Boys (n=28)	Total (n=41)
R1	It is useful and not just a gimmick	1	1	2
R2	I like the visual design (figures, background)	6	6	12
R3	I think the idea is funny and smart.	8	13	21
R4	I had immediately an idea how this program works	4	6	10
R5	I had immediately an idea what to improve	8	17	25
R6	I have seen similar programs in my spare time	2	0	2
R7	I wanted to know how to program such application	4	7	11
R8	To me this seemed to be the easiest project	7	5	12
R9	This program is similar to one of the programs that I have developed in Stage 1.	2	4	6
R10	Other reasons	0	0	0

Interaction and Inspiration during the Development Process

How did the students perceive their situation during the development process in Stage 3? Personal advice from class mates was obviously of much more value than the help system (Table 2)

The students were encouraged to change the chosen artifact as much as possible. It had to be transformed into something new. What did the students actively do to get knowledge and ideas for this? The single activity that was mentioned most, is trying out Scratch commands (21/41).

Table 2: Sources of help during development (n = 41)

Source of Help	Frequency
I got good advice from class mates.	13
I got good advice from the teacher.	9
I found good advice in the Scratch help system or online.	1

Table 3: Knowledge winning activities during development (n = 41)

Activity	Frequency
I tried out new Scratch commands.	21
I looked at projects from other teams, when they were still not finished.	17
I asked class mates for help.	15
I asked the teacher for help.	11
I showed my project to class mates (from a different team), while I still was developing.	10

As Table 4 suggests, Scratch artifacts seemed to be in fact a very important source of inspiration – more important than conversations with the teacher. Although the majority tried out new commands, only 4 students claimed that changing the program code lead to a better understanding. This can be interpreted in two ways: The other students comprehended the program from the beginning or they still did not understand it after having modified it.

Table 4: Sources of inspiration and knowledge during development (n = 41)

Experience	Frequency
I got new ideas during the development	17
I got ideas from other Scratch programs	14
While talking to the teacher I got new ideas for my project	6
While talking to other teams I got new ideas for my project	6
While I changed the program I began to understand it	4
There were other situations, where I got new ideas for my project	3

Inspired Creativity

The students participating at the study were beginners. Beside the introduction in Stage 1, most of them did not have any programming experience. Nevertheless, they were creative. They changed the images (68% claimed to have done so), changed or added commands (54%) or added scripts (41%) or sprites (37%), thus creating a more or less new artifact.

One could observe different “degrees of creativity” represented by different amounts of new intellectual content that was produced. Let me give some examples from projects that were inspired by the car race (M 3).

Some student changed the image but did not change the story. The left hand screenshot in Figure 6 shows an example. There is more decoration, different colors and forms but it is still a car race.

Other students did not change the algorithmic structure very much, but created a completely different story just by redesigning the images. Among the projects of this kind there were a football game (Figure 6 right hand side) and a game, where the player must navigate some kind of balloon through holes in several horizontal floors to a goal (Figure 7 left hand side).

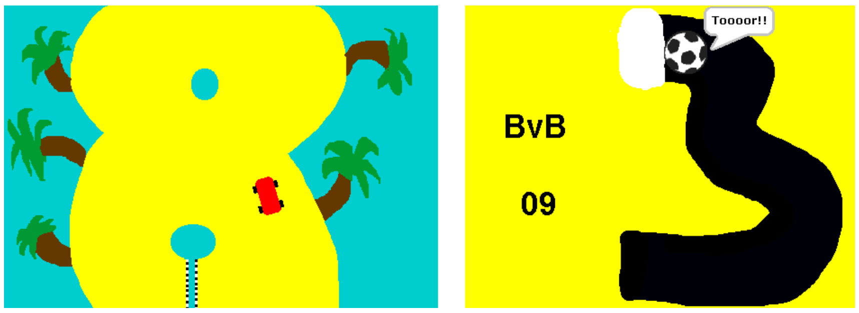


Figure 6: Screenshots from two games developed by two girls-teams (13 years). The right hand image refers to "Borussia Dortmund" (BVB 09), the German soccer champion 2012

In some cases the functionality of the chosen project was extended. Two students (age 13) had the idea to add a second car, so that it was possible to simulate a race with two drivers. They changed the scripts, which are responsible for movement control.

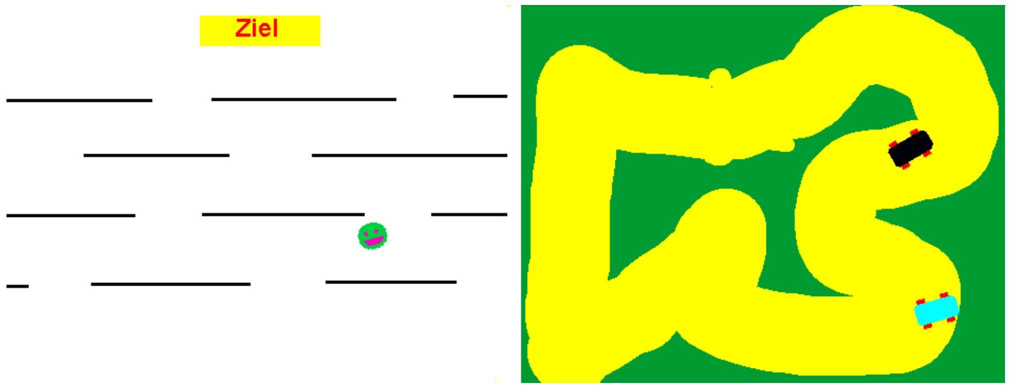
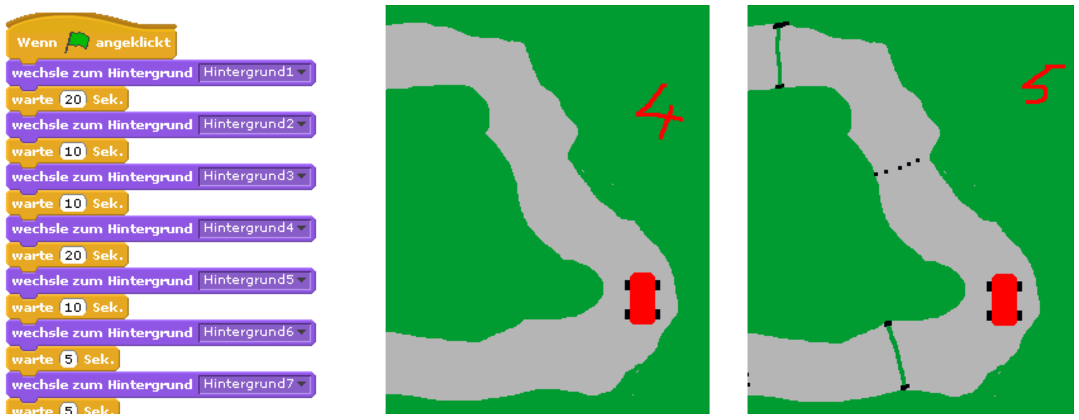


Figure 7: Screenshots from a "balloon game" and a car race simulation with two cars instead of one.

In the last example the structural development was driven by a functional idea. But inspiration also works in the other direction. Tom (13) found out how to change the background image, depicting the racing track. Originally he wanted to create several levels of difficulty. Since he did not know any other mechanism, he wrote a script that changed the background in certain time intervals using the wait-command. First he designed different tracks for the different levels but then he realized a severe problem: when the background changed, in many cases the car stood on the green off the track and stopped immediately. This was boring to play. So he designed different tracks with the same shape and created new features (like obstacles on the track) to make it more interesting to play. In this case a structural feature (time controlled events) inspired to create new functionality.

Most students enjoyed creating a Scratch application. 41% of the participants stated that they did not really want to do the project at the beginning but started to like it during the process. This positive attitude - emerging during work - fits to the results of motivation research. The "art inspired by art" metaphor seems to generate a classroom situation with a high degree of personal causation. This happens in a "natural" way, and not through teacher's interventions. Richard deCharms points out that “...to help a person be an Origin, you must help the person, (a) to determine realistic goals for himself; (b) to know his own strengths and weaknesses; (c) to determine concrete action that he can take now that will help him to reach his goals; and (d) to consider how he can tell whether he is approaching his goal, that is, whether his action is having the desired effect.” (1972, p. 97) “Getting inspired” instead of solving a task implies a personal decision about what to do. And the iterative development process (similar to Extreme Programming; Beck 2000) improving a given minimal application step by step, corresponds to the factors fostering personal causation given by deCharms.



```
Wenn angeklickt
wechsle zum Hintergrund Hintergrund1
warte 20 Sek.
wechsle zum Hintergrund Hintergrund2
warte 10 Sek.
wechsle zum Hintergrund Hintergrund3
warte 10 Sek.
wechsle zum Hintergrund Hintergrund4
warte 20 Sek.
wechsle zum Hintergrund Hintergrund5
warte 10 Sek.
wechsle zum Hintergrund Hintergrund6
warte 5 Sek.
wechsle zum Hintergrund Hintergrund7
warte 5 Sek.
```

Figure 8: Screenshots from a car racing game, where function design was inspired by a structural feature

CONCLUSION

In a self-directed programming project students are origins of activity. They get the experience that knowledge is necessary for being able to produce new ideas and implement them and that there are several different sources of knowledge: artifacts that you can study and take ideas from and persons who know things you do not know and who you can ask. To see the input from the environment as inspiration implies to keep control. A creative person is an origin. An inspiring artifact for beginners must be simple (minimal) and its structure must be easy to comprehend. According to students’ perceptions, substantial learning by deconstructing a selected Scratch program did not happen that much. They primarily chose a certain Scratch project as a starting point because they liked its purpose, its visual design and because they had ideas about how to change functionality. They focused on function – not on structure.

But for implementing functional ideas, beginners often need additional structural knowledge, which is not easy to take from other programs. For this they need sensitivity, curiosity and an “open minded” attitude. “What can I learn from this piece of work?” Taking inspiration is a skill.

REFERENCES

Beck, K. (2000): Extreme Programming Explained. Addison-Wesley.

deCharms, R.(1972): Personal Causation Training in the Schools, Journal of Applied Social Psychology, 2, 2, 95-113.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. (1995): Design Patterns -Elements of Resuable Object-Oriented Software. Reading, MA (Addison Wesley).

Kafai, Y. B.; Peppler, K. A.; Chapman, R. N. (2010): Computer Clubhouse: Constructionism and Creativity in Youth Communities.

Kroes, P.(1998): Technological explanations: The relation between structure and function of technological object. In Techné: Journal of the Society for Philosophy and Technology, Vol 3, No. 3.

Martin, F., Butler, D., Gleason, W. (2000): Design, story-telling, and robots in Irish primary education. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Piscataway, NJ., 730-735.

Reich, J.W.; Zautra (1981), A.: Life events and personal causation: Some relationships with satisfaction and distress. Journal of Personality and Social Psychology, Vol 41(5), Nov 1981, 1002-1012.

Resnick, M.; Maloney, J.; Hernández, A. M.; Rusk, N.; Eastmond, E.; Brennan, K.; Millner,A.; Rosenbaum, E.; Silver, J.; Silverman, B.; Kafai Y. (2009): Scratch: Programming for Everyone. In: Communications of the ACM, Vol. 52 No. 11, 60-67.

Romeike, R.: Animationen und Spiele gestalten - Ein kreativer Einstieg in die Informatik [Designing animations and games – a creative introduction to computer science], LOGIN 146/147 (2007) 36-44.

Schulte, C. (2008): Duality reconstruction – teaching digital artifacts from a socio-technical perspective. In Proceedings of the 3rd International Conference ISSEP: Informatics in Secondary Schools (1-4 July 2008, Toruń, Poland).

Schwill, Andreas (1993): Fundamentale Ideen der Informatik [Fundamental Ideas of Computer Science]. In Zentralblatt der Mathematik 20, 20–31.

Weigend, M. (2007) Intuitive Modelle der Informatik [Intuitive Models of Computer Science], Universitätsverlag Potsdam.

Weikop, Ch. (2002) : The Dresden pioneers of the German avant garde? In: Art History, Volume 25, Issue 3, 412–417.

Biography

Michael Weigend studied Computer Science, Chemistry and Education at the University of Bochum and the University of Hagen and received a PhD in Computer Science from the University of Potsdam. He is a teacher at a secondary school in Witten, Germany and he has taught Didactics of CS at the University of Hagen for almost 20 years. He has published several books on computer programming, web development and visual modelling.