Award Image

For the second time, transmediale.02 will award the prize for Software Art. In this category the artistic process is largely dependent on the execution of computer code. Software not as a functional tool on which the "real" artwork is based, but software code as the material of artistic creation. Software Art can be the result of an autonomous creative practice, but can also refer critically to the general technological and social meaning of software.

Jury Statement:

The definition of software art seems to start from a negative point. Software art is said to be neither media art nor interactive media art nor net.art, but something in between.

Going more into depth, one can say that interactive art is a step towards a deeper human experience of electronic and digital media, where artists try to touch the untouchable by creating concepts that offer possibilities to get closer to the structures and processes that reside inside electronics, computers, and networks – or by creating concepts that reflect problems to reach this goal. Media art and net.art, on the other side, reflect more of the social, communicative and political aspects of electronic, digital and networked media: more of what happens in front and because of the media.

Art that people have started to call software art is more general and more specific at the same time: It is more general in the respect that it comprises complex interaction just as well as critical reflections of computer and media use. It is more specific in the respect that it focuses explicitly on the creation, on the direct interaction or on the practical critique of software: how it is made, how it is used and how it spreads and integrates into human life. But where lies the real difference to older art forms like the mentioned media art, interactive art and net.art? All these involve software in one or the other way. However, they don't focus on software as their crucial point, but on media criticism, man-machine interaction or aspects of an online life. In software art, the software does not serve a merely instrumental function, but the code is its material, or the art of coding is the artistic craftsmanship of software art. Our criteria as jury for software art have been to only consider artistic propositions which focus on the confrontation with software as a conceptual system or a process. Running code had to be in the center of an artwork to qualify it for the award. However, this does not mean a total reduction on an art of coding, but it includes pieces where the system is perceived through a visible, audible, interactive or in any other way graspable product, as long as this sensual output is considered simply a part of the system rather than its raison d'être.

Among the 47 projects that were submitted as software art we found a large number that did not qualify for the category, meaning that they did not fit our above stated definition of software art (while among those off-track works were several artistically interesting ones). That shows on one side that such categories are always in a sense artificial, and especially a category as new as this one (this is only the second year that an award for software art is being given) is in the danger of being misunderstood – either by the artists reacting to the announcement of the award or possibly even by those who try to establish a term in order to make new artistic developments »speakable«. On the other side, announcing a new category or kind of art award states that there are already artistic activities going on which can't be grasped by the traditional categories any more. So there is a need to draw the outlines of a new array of artistic activities, but what exactly should be included in this sketch is not clear at this point of time.

The projects that we nominated for the software art award of the transmediale.02 reflect outstanding tendencies which can be traced among this year's submissions.
There is a type of software art that approaches the art of coding as a craft, which transforms into an art in the emphatic sense. Close to the tradition of the objet trouvé – the forkbomb.pl program is a found object, being a variation of a famous computer science exercise – the author radically stresses the poetic dimension of programming by confronting this very short 5 lines program with its dramatic effect: by duplicating itself as fast as possible, the program grinds the underlying operating system to a halt, all the while blowing open its internal mechanisms. Roman retrograde is an artwork that reflects on our relationship to software automata by playing with the awkward situation of having a piece of software that makes its own choices among our very personal data. The data of our hard drive is part of our personal memory, and we have our personal way of remembering and using this data. Roman retrograde is like an alien externalisation of our internal mental processes: as if the thoughts / data of our memory were thought by somebody else, resulting in surprising connections and insights. This artwork thus demands a very personal relationship and can hardly be exhibited publicly.

In the realm of the critique of public instances of software, Tracenoizer reveals and uses marketing software techniques to create, endlessly, actual lures against these same digital marketing strategies. By giving your name to this Frankensteinian system, you unleash an invasive protector, faithfully reporting to you all the effort it goes into for your supposed sake The lures it creates are clone web sites that look like yours could look, so that maybe you are less easily traced down by marketing software. A protector (or guardian), like the lures of submarines. Funnily, the contents of web sites produced by this system are in many cases not far from the majority of information present on the net. So Tracenoizer can also be understood as an ironic, automated version of the kind of easy-to-use HTML editor software that does everything for you – except generating content. Content simulation could be an appropriate term for this, possibly to be introduced as an influential technique for the IT sector soon.

Somewhere in between the mentioned concepts flows the last piece, retroYou r/c, which directly points to the essence of software by braking the internal system of a classical videogame, thus revealing the very nature of

software systems: they are deep simulations (just like reality). You won't forget the sensation of this aggressive physical encounter with a broken world, in which the rules of physics have changed – maybe by a step in evolution? However, retroYou r/c takes a clear stand on what to consider functioning or broken: With the slogan FCK TH GRAVITY CODE it supports the evolutionary concept that there is not one truth and that so-called errors easily lead to constructive results and new aesthetic outcome.

Golo Föllmer, Robert O'Kane, Antoine Schmitt