Aesthetics of Programming - Interview with Mark Napier

by Andreas Brøgger

[This interview was completed in the fall 2000 for the online+print exhibition ON OFF at afsnitp.dk and in the printed magazine Hvedekorn, both based in Copenhagen, Denmark (http://www.afsnitp.dk/onoff/). Issues touched upon are: the role of programming in creation and appreciation of net art, the market value of net art, the (art) history of programming, the (non)physicality of the net, and the formal and political dimensions of net art.]

Andreas Brøgger: I would like to begin by asking your view about net art in relation to painting, first of all because I know you used to be a painter as well as a computer programmer, secondly because you often speak of a certain painterly dimension in your net art projects.

Mark Napier: Art works best when it is interesting both conceptually and visually. The Shredder is a good example of a project that has both a conceptual and visual dimension. I spent a lot of time tweaking the aesthetics of the layers of output text and graphics. I wanted the shredded page to be variable, somewhat random, but still have a structure that would create a visual balance and flow.

I enjoy the visual quality of graphics on the computer screen, the luminosity of colors, the hard edges and the atmospheric blurs that come out of distorted and pixelated computer graphics. There's a painterly side to the medium that I want to explore further.

The web interface has capabilities that go beyond painting though. In net art the artwork can move. It can change and evolve over time, which creates a whole other dimension to this art form that just doesn't exist in painting. In Pulse I spent a lot of time looking at how colors can evolve over time, how the mood and character of a series of colors can change and take on new qualities. That piece sets up certain color qualities, color areas that pulse at different rates. Once the visitor interacts with the piece they activate the color areas and the rest cannot be predicted exactly. I created a potential for color relationships, but the visitor actually triggers when and how the colors will unfold. I don't control the final outcome of the piece.

And that's the fun of software: it can create results that I didn't plan on.. Software can combine a simple set of rules to create a complex result, that can go in directions that I can't predict, even though I made it. To a software developer (my day job) unpredictable behavior usually means bugs, and gives me headaches, but as an artist I often prefer the unplanned results. In painting the medium can do things that I can't predict as well, but once the paint dries, that's it. The image is static.

AB: To push an analogy between painting and your current work could one consider <tags> to be equivalent to the painter's brushes, and the values in the tags as equivalent to the paint? In both cases the object produced with the two aspects - tags/values and brush/paint respectively - cannot be deduced from these two generalized aspects in themselves, only from their specific combination.

MN: In any medium the object produced can't be predicted by looking at the tools and materials of the medium. This is true of software languages, as well as paint and canvas, charcoal and paper -- that's what makes these tools so powerful and so attractive to creative people. This is also true of language: an alphabet and a grammar can produce endless combinations of words and sentences.

I don't agree that painting and software code compare so directly though. The two media are completely different. Paint is a physical fluid that is directly controlled by hand. Software code is a language (of sorts) that is controlled through a computer interface. A painting is a static physical object. Software is more like a machine with moving parts, or a piece of music.

AB: Do you find these comparisons between net art and "traditional art forms" meaningful and productive, or does it more often lead to a reductive view of what net art is?

MN: Comparisons are useful when they serve to create something. Most often comparisons between net art and traditional media end up saying that net art is "better" or "worse" than traditional forms. Neither judgment makes sense.

AB: To what extent do you think the creative dimension of net art resides (or should reside) in the coding itself?

MN: What I do requires a strong knowledge of code, because I like to create unusual effects that challenge assumptions about how the web works. Like "shredding" a page, or merging separate pages (i.e. RIOT). To do that I have to work around the restrictions the browser imposes, and that means working closely with code.

Knowing how the code works gives me access to more of what the medium has to offer. With Java and Javascript I can create visual effects that the folks at Macromedia never put into Flash and Dreamweaver. For the kind of art I make, this knowledge is essential. I'm always learning more about how the technology works.

But there is no one way to be creative, and not all net art necessarily relies on coding. Conceptual work can make a point just through html. What concerns me are the curators and critics that are ignorant of technology, and make decisions about net art without understanding how the artwork fits into a bigger picture of the technology environment.

AB: You have previously spoken about a certain "secret" aspect in programming, a part of the code that you would usually like to keep for yourself,

because it holds the key to how you created the piece. Perhaps you would explain this further?

MN: There are two aspects to producing a software application. There is the 'code' which may look something like this:

```
if (ns4 == true) {

this.x = this.css.left

this.y = this.css.top

}
```

and there is the actual 'binary' or machine language which may look something like this (if you ever manage to see it on screen, which is rare):

ª_"__ï!2¬_J?HÙUæ"ºöË9[2ã\]I÷I?W-¥ZªJ±(åb

The code, which humans can read and work with, is translated into binary, which the machine can execute. This translation is called compiling, or interpreting the code. If you open a web page in your browser, then choose the View|Source option from the menu, you can see the source code for that web page (html and maybe Javascript). In other systems, such as Flash and Shockwave, you can't see the source code because the Flash software compiles the code into a binary form. The author keeps the code on his computer, and puts the binary file onto the web, where it can be retrieved and executed (with the help of a Flash plug-in).

In the software industry the code is very valuable since it contains the knowledge, recipe or blueprint of how the software product is made. The binary "executable" is distributed to the world, but the source code is carefully guarded. As an artist I'm happy to share most of my source code with other artists. Usually they can get to it just by looking at the html View|Source option on the browser.

When I work in Java, the source code is not immediately available to viewers of the artwork. In this case the source code becomes a unique blueprint for the artwork. Whoever owns the source code in effect 'owns' the artwork.

AB: Do you see the code itself becoming the "artwork" sometime in the future then, code being sold at art auctions, displayed in museums?

MN: I don't think so, not as a routine event anyway. I'm sure code will turn up in the collectibles market in some form, but probably more as a curiosity than as a work of art. Code is like paint and brushes. We don't revere Picasso's paint brushes, though I'm sure somebody owns a few, and could sell them for a nice price.

AB: But would you see the possible "marketability" of the code as a viable alternative to the current situation for net art, which is that art institutions commission work from net artists, meaning that artists have to make proposals (and do a lot of work) before possibly being paid for their art? Or would you still rather favor an "open source" principle?

MN: I prefer open source, but I have to eat, too. I'm still thinking about this and don't have any good answers. Since I'm using Java more now I will have to address this issue (as the source code in Java is separate from the final compiled product and can be withheld, thus making the source code rare in the art world sense).

AB: Would you describe the programming as a highly creative dimension in itself in your own work, or does the creative dimension of your work rather reside elsewhere: in coming up with an idea for the project, the "look" of the project, its interaction with users, etc.?

MN: These are all dimensions of the creative process. My projects begin with an idea, usually an idea about the web, how we use the web or how it impacts us. But then the coding process creates other possibilities and adds another dimension to the piece. The technology often limits what I can do: there are limits to memory, and processing speed, there are bugs and incompatibilities in the browser software. Other times the technology reveals possibilities. Accidents happen, mistakes in the code, that produce unexpected but wonderful qualities. Or I'll find some software that suggests new ways of working with a problem.

In general, it's not about looking at code as art, but understanding how the underlying technology affects the art. The invention of oil on canvas significantly altered the nature of art. That new technology made art portable. Paintings could be carried from place to place. They could be large and durable paintings, but still lightweight. Images no longer had to be painted directly onto walls or altarpieces or pieces of wood. They could be moved, and so could change owners relatively easily, opening up the possibility of a secondary market in which art could be sold and resold. The technology of oil on canvas played a role in transforming art from large public works to smaller, portable privately owned pieces, which is how we know art today.

In the case of the internet, there's an even greater need to understand the medium, since much of a networked artwork may be invisible to the eye. In projects like Shredder and RIOT the collage you see on screen is created by appropriating content from the web. I have talked to several curators who did not understand the relationship of the artwork to the medium it lives in. They didn't realize where the content of the piece was coming from, and completely missed the point of these projects. In Digital Landfill, and GraphicJam, the network is also a driving force behind the work, and the uninitiated may not realize how these pieces relate to the surrounding environment, to the space of the network, and how other users contribute to the work.

These works are about distributed authorship and appropriation. They exist in the space created by the network, which has different rules and qualities than what we're used to seeing in the physical space we live in. But these qualities are not immediately obvious and have to be learned.

When I started working with computers I had to learn to visualize what was going on in the machine. Once I learned how to 'see' the environment of the operating system I had no problem navigating that space. Curators and critics that look at net art have to go through this process as well, but they may not

realize it. If they look at screen shots, or look briefly at a monitor, they may be looking at the work as a photograph, a static image conveniently framed on the monitor. If they don't know how that image was created, how it relates to the network, to other visitors, to interaction, then they can't possibly understand the message of the artwork.

AB: You've recently spoken about your interest in giving the code in your projects a double role, making it appear not only as the somewhat impenetrable mass of code, but also giving it some sort of conceptual meaning on another level. By way of illustration you mentioned the film The Matrix where the programmers can actually "see" the constructed worlds by looking at the seemingly impenetrable mass of code on computer screens. A kind of iconic level in the code, perhaps?

MN: I read Neal Stephenson's book Snow Crash recently and enjoyed the comparisons he makes between code (as a language that builds a virtual world) and the languages of early religions (another form of language that builds another sort of 'virtual' world). I'm thinking more about code as a metaphor for the creation of the world, as in Shiva which I will return to below.

Code is a language that can be used to create both structures and surfaces (appearances): machines that work behind the scenes, and interfaces that people see and interact with, and through which they can direct the machinery. We live every day in a physical world that is created by our language. When you cross the street you look out for "cars". You encapsulate an entire range of experiences and memories into a single word: "car". What you are "really" seeing is a structure of metal, rubber, plastic and glass that moves very rapidly. But even those words, "metal, rubber" etc. are terms that describe physical qualities like hard, smooth, cold and inflexible, or soft, pliable, black. And those physical qualities are really sensations that the words only point to. We all agree on what "smooth" and "cold" mean. The words are not the sensations, they are symbols that point to the sensations.

With a single word, "car" we can summarize an entire range of experiences, attributes, capabilities, and dangers. We can ignore the million hues of color reflecting off shiny surfaces, the light, glints of glass, the many shapes of curves of shining metal, the sounds, the movement, etc. If you see something for the first time you can get a sense of this. A completely foreign object looks exotic, exciting, intriguing. It holds our attention exactly because it cannot be categorized. Once we see a few of them and label them then the exotic beauty fades and we stop really seeing the object at all, it becomes another thing that we can instantly assess, name, and ignore.

The symbolic structure of language allows us to navigate in a map of the world rather than the world itself. We can't say what the world 'really' is because we would have to use words to say what it is and then we're back to the mapping and representing of things with symbols. To experience the world 'as it is' requires a direct experience without words. Nobody operates this way once they learn language, unless they engage in meditation, or take the right combination of drugs, or perhaps fall out of a building.

Language is a very powerful tool that allows us to create a map of our physical experiences and then navigate in that map, and by so doing we can communicate complex experiences and actions to others, even long after we die, if we write them down. Most of the time we can't separate the map from reality. It *is* reality as far as we can tell. But drop acid a few times, read Aldous Huxley, spend some time with Zen Buddhists, and you start to suspect that reality isn't reality. It's a map of some other experience that is outside of language, that we can't describe, exactly because it's outside of language.

The point of the above digression is that software is very similar to this. It also allows us to create symbolic structures that create an illusion of a 'reality', a constant, consistent environment in which we navigate. What's fun about it is that they're sort of mirror images of one another, reversed.. In physical reality, language maps a territory. In the software "reality", code *creates* a territory. We make it up.

When I start writing code that will take actions, create aesthetic qualities, respond to user actions, and run 'machines', that will all add up to an artwork, I am compelled to ask myself "What do I call these bits of code?". A program is divided into pieces called functions, and functions can be combined with data into "objects". So an object may be called "Box" and it may have a function called "drawSelf" that draws a box on the screen, and it may have a piece of data in it called "opened" which can hold the value "true" or "false". The function "drawSelf" will look at the data "opened" and if "opened" is true, then draw a picture of a box that is opened at the top, otherwise draw a box that looks closed. So the data and the functions work together to create something that behaves consistently, like an object in the "real" world, though this object exists only in the virtual metaphorical environment of the software system that the program is running in.

Obviously I called this thing "Box" in the example above, since that's what it looks like and behaves like. But what do I call an object that is part of an artwork? I see art as a metaphor for life. Life is a creative act. It exists to re-create itself in infinitely varied forms. It's like a dance. Life dances because that's what life does. If it stops the dance, it's no longer life, it's dead. There's no reason for it. And that's like art, when it's really good, there's no reason to do it. I make art because it's alive and I'm alive when I'm making it. Art is a creative act for no reason. That's why I thought of Shiva when I started looking at naming the objects that would operate an artwork. Shiva is a Hindu god that embodies both the destructive and creative forces of the universe. Shiva is portrayed dancing; in a particular sculpture Shiva dances within a circle of fire. At the core of the code that I'm writing is an object that communicates with all the other objects, that animates them, triggers their behaviors, activates and deactivates them. This object reminds me of Shiva in that it animates the universe of objects that I create within the code. Each of the objects creates a different kind of color effect, and this central object is like Shiva, dancing in a circle of fire.

This is all pretty metaphorical and nebulous considering that code is also very "nuts and bolts", like riveting together a thousand pieces of metal to build a bridge. Coding can be completely tedious and frustrating when you're dealing with the nitty gritty details. But I suppose that both of these qualities are part of our reality too: the mundane details and the spiritual. If I didn't see the potential for

these beautiful structures in code I would never have the patience to be a programmer.

AB: Speaking of nuts and bolts and beautiful structures in programming, I am wondering if these dimensions might not be interesting to look at from a historical point of view? Do you think there have already been episodes in the history of programming which parallel the much discussed developments in the history of art in the previous century, such as e.g. the introduction of the readymade, the death(s) and revival(s) of painting?

MN: Yes, though I think the words "death" and "revival" may be too strong to describe trends in the short history of programming. Structured programming, and Object oriented programming are two trends that I've seen in my career. "Client-server" is another form of programming that evolved considerably over the history of software development. Still these trends are more about practical needs than the passionate philosophical debates of the arts.

AB: Let's talk about your project for the printed magazine Hvedekorn. For instance, I am curious to know where the blurred backgrounds in the images in Hvedekorn come from? And to know more about your interesting ideas about virtual and real packaging...

MN: I took a screenshot of some windows, then used the Photoshop "sharpen" filter several times on that image. Sharpening many times produces a ripple effect -- the filter sharpens the already sharpened edges. I expanded those ripples about 10 times to get that blurry rippled quality.

The cover of the magazine is a screenshot from RIOT, blending Yahoo, The Weather Channel, and The White House homepage. The 14 pages in the magazine, the 7 full size spreads, feature a lot of fragments such as images and instructions lifted from product packaging. I've been collecting bits of the generic instructions and labeling that appear on boxes and bottles in the supermarket: "Open here", "Tuck flap into slot", "Store in cool dry place". Once I started working in the virtual world I began to notice these instructions that appear on literally every piece of packaging (at least in the US). I find them to be peculiar to the physical world -- they make no sense in a virtual environment.

The 'window' is the ubiquitous packaging of the computer world: windows deliver information in manageable chunks. Like the pages of a magazine, another form of packaging and delivery. So I wanted to put together pieces of windows with scanned fragments of packaging, and place these on the page in such a way that the page becomes a kind of "package" referring to the packages of window and physical products.

What intrigues me is the contrast between virtual and real. Virtual "space" makes the old space obsolete in certain respects. I don't mean that we're all going to abandon our bodies or live in a science-fiction world tomorrow.. But we do have new options for moving goods, services, and experiences (like art) that used to be confined solely to the physical world. It's easier to do a lot of these things online now. For instance I can meet and socialize with people online, and not just talk to them (i.e. telephone), I can see their web site, see things they've created, share

experiences with them online. I almost never meet people in my own apartment any more -- most of my social life happens online. And that puts my apartment in a new context. I don't need it the way I used to. Same goes for my office, and my studio. Both are wrapped up in my computer and potatoland.org.

These thoughts led me to scan my apartment in "Negative Space", a project I started in 1996. And that's why I notice these little instructions all over the physical world ("Tear here", "Fold tab into slot", "Open other end", "Shake well"). They are completely tied to that environment, and make no sense in a virtual world. In fact these actions are impossible in a virtual world, a space without solid objects, where we find nothing we can shake, fold, or tear.

I put a "Shake well" label on the opening page of Potatoland a while back. That non-sequitur appeals to me because it points to the differences between the virtual and physical. We are shifting a very fundamental aspect of our world, whether we realize it or not. We can "interact" with the virtual world, but we can't really *touch* anything there.

AB: I see a connection between the idea of *touching* physical material and your Shredder project. Shredder seems to treat the virtual as *material*, just like another of your projects does, Digital Landfill, a virtual, public garbage dump for web material.

MN: Yes. Both the Shredder and Digital Landfill contrast the physical and virtual worlds. The Shredder appears to rip a web page to pieces, and Digital Landfill uses the metaphor of a garbage dump to 'dispose' of unwanted digital material. I want to satisfy a physical urge to get hold of this virtual 'stuff', the material of the web, whatever that is, to act upon it, to alter it, even if it is the content of another web site that is supposed to be off limits to me. In the process it becomes clear that you can't really shred the web the way we can shred the 'real' world, but that's what makes it fun to try.

AB: I also like the fact that you break with the usual packaging of virtual products, which in a way seems to me very limited. For instance, browsers and operating systems reduce the packaging and handling of an endless variety of virtual products to a limited range of procedures, such as clicking and dragging (despite the fact that these techniques have taken a long time to perfect). Why do computer files, such as texts and images, have to be "square"? Of course, this resembles our usual containers for information like the text page or the television screen. But do you think we will abandon the square shape of screens and virtual windows, and in the future have screens and files which are, e.g., round or triangular, and maybe even have a tactile dimension built into them, like real packaging and real objects?

MN: I don't think rectangles will go away any time soon. They've been a dominant interface element for thousands of years (rows of text are basically 'rectangular'). Some software interfaces break with the tradition of square buttons and windows. And the web, with its proliferation of unique graphics for interface elements, has loosened up the standards in windowing operating systems. Still standards arise fairly quickly, i.e. the 'rollover' effect, where a button changes color when the mouse passes over it, which started as an unusual addition to web pages

and now is nearly ubiquitous. Interfaces benefit from being simple and standardized. The mouse is a simple device that can perform a wide variety of tasks through a simple operation like drag and drop. The simplicity is what makes these interface elements powerful.

There have been attempts to make the virtual world look and act like the real world, through VRML for example, but these seem to have failed, at least for now. A two dimensional web page is usually easier to navigate than an illusion of three dimensional space. The computer interface is still two dimensional, and translates awkwardly into the spatial dimension of a rendered 3D space. When it comes to navigating information, the written page is a more familiar metaphor than 3D space. We read newspapers, for instance, which are mostly text. The layout of text and images on a page is a familiar and practical way to present a wide variety of information.

We don't need to replicate 3D space in the virtual. VRML worlds don't need to have gravity. They don't even need a flat plane on which people walk. The first time I visited Active Worlds I spent the whole time flying. It just made sense as a way to get around in that space. I didn't want to have to navigate on the ground, the way I have to in the real world. These worlds don't need a 'ground' at all. The VRML world can exist as a spherical space without gravity, which could give access to more experiences more conveniently than a traditional world with gravity and a horizontal plane.

So to answer your question, I think that real packaging and real objects have a lot of drawbacks when compared to virtual objects. It is easier to navigate in a virtual world that is non-tactile. While there is a vicarious thrill to simulated spaces, that's a different kind of experience than the day-to-day process of navigating the web.

AB: Is there a real life parallel of RIOT as is the case with Shredder?

MN: The real life parallel to RIOT is a real riot, I suppose. When you have two cultures and classes facing off in a confined territory, like Tompkins Square here in New York in the late eighties, you get a forced overlapping of ideologies. Beatnik squatters vs. yuppies and real-estate developers. RIOT pursued that physical clash, but in a virtual arena, by overlapping very different web sites into one browser window. For many people the result is an insult, because they see their domains as unassailable territories, or at least that's what they want them to be.

AB: This leads me to a last set of questions. Shredder, RIOT, and Digital Landfill have a kind of "anarchic" feel to them, and you have previously found yourself in a legal battle with Mattel over your Distorted Barbie images. To what extent is there a political dimension in your work? And how do you see this dimension in relation to the more formal issues you talked about earlier?

MN: These works have an element of frustration, a desire to influence, to affect an environment that has been structured to be outside of my control. I enjoy finding ways, through code, to get inside somebody else's web page. That's essentially what the Shredder and Riot do. I used code to get around the rules of division and territory that the browser imposes. These rules are artificial, they are

created by software, and as a programmer I have access to software too, so I can undo some of those rules and create a web browser where two or three different web domains can be overlapped in one window. The word "domain" has a such a territorial sound to it, yet that territory exists only so long as browser software follow the rules. It begs to be broken.

I see this as a question of personal identity, not so much a political issue. I'm interested in how we react to the web, as humans, as physical bodies, as individuals with our own personal concerns: a desire to be seen, to be known, but also to be private. That can be political, but I don't take it that way.

The formal issues of technology create new legal challenges. The web gave me a means to publish the Distorted Barbie, the same means that Mattel employed when they made Barbie.com. Through this medium, Mark Napier and Mattel meet on new terms, and are suddenly producing very similar products. Technically there's not much difference between my web site and Barbie.com. Now the corporation has to compete in a new way with an individual. The 'soft' world created by code has different rules than the 'hard' world that we're familiar with and that will lead to changes in law and politics, however slowly.

AB: Do you think net art invites a new constellation of the political and the formal in art?

MN: The internet creates a new level of portability for art, and that changes the nature of art.

About five hundred years ago the invention of oil on canvas revolutionized art by creating a portable, standardized art form. Anybody with a wall could hang a painting, and that opened up the possibility that art could become a private activity, outside of the public and commercial projects sponsored by state and religious authorities. The convenience and practicality of the new medium made it easier to distribute. Coupled with new knowledge of the natural world that allowed for richer rendering of subjects, oil paintings became compelling and desirable objects with enough potential to survive for hundreds of years as the medium of choice for visual art.

The net has a similar potential to alter the nature of art, by creating a very portable art form. Net art can be viewed by anyone with a browser. They view the art in the same way, in the same space in which they view any other site on the web. Give or take a few plug-ins, the equipment for viewing net art is no different from the equipment needed for viewing the web itself. This creates a much broader means for distributing art, not just the artwork itself, but also spreading the conversation about art.

I don't think that art itself has much impact on the political. But the structure of the internet creates many challenges to the existing legal and political system. The internet brings new definitions of ownership. Objects and territory are redefined. New conceptions of value arise, and new ways of creating value. For example, in open source projects many people work together to create software. The value of this process is not reduced to a profit motive, to a dollar amount. The software has value for those that use it, so programmers are willing to contribute

time and skill to creating what they need. The resulting open source product is not a commodity to be owned..

The internet is spawning changes in business, in software development, in economics and law. The changes we see in the art world are part of the larger process.

http://www.afsnitp.dk/onoff/texts.html
http://www.afsnitp.dk/onoff/
http://potatoland.org