Flaxus

toplap flash based aplication

In short

Flaxus is a software developed under the manifesto, that carries out visual performances in real time. The graphic piece is generated by code at the same moment of its execution, thus, the artistic experience comes closer to a performance event such as music or dancing. But Flaxus also brings forth a new paradigm, since it allows a user to execute visuals that hundreds of other participants can watch activated by their indicidual audio in different parts of the world. Therefore, the same visual composition becomes reactive to individual musical perception anywhere. Flaxus is also a collaborative tool that promotes network tasking by allowing the real time creation of a piece amongst various executors connected through the internet.

Motive

We believe that electronic audiovisual arts still demand great experimentation and further insight into their fundamental search. Flaxus is a field work, a statement that privileges visual over musical performance. It uses network capabilities to create elements that simultaneously react to different environments. It fosters collaborations between people working at a distance, programming codes and live interpretative processes. We seek to investigate the boundaries of live visual performance.

Path

For the last couple of years we've been researching the live performance potential of electronic audiovisual arts. Trying to explore the similarities between playing a musical instrument live and generating visual contents in real time.
At first we turned to Visual Jockey or VJ, a visual performance in which an operator displays images that accompany audio. But we found that Vjing is closer to what a DJ does, since it involves mixing rather than composing. Even when the mix incorporates pieces of the VJ's own authorship, it's not quite what we're looking for. Given that there is a huge difference between playing a tune live on a guitar composing a melody note by note, and mixing something, determining which previously recorded samples should be heard at a certain time. The same is true for VJing. This kind of performance is never 100% live, as playing a piano would be.
We then determined it was important to define the smallest visual unit, comparable to a note in music. The problem is music only has one axis of data, whereas a digital image contains a lot more information. A note is a wave moving in just one direction, with different values over time, while a flat visual display has at least two axis that run perpendicularly to make up a grid of pixels.
Additionally, the surface is related to a fraction of visual time, around 1/30 and 1/12 of a second. By attempting to make a metaphorical translation from music to image processin, we could only control minimal data in a timeline, we were only able to turn certain elements on and off, slightly shifting color or shape.
We then realized we could not metaphorically translate the wave creation process to an image. Otherwise the visual composition would involve different values of only one color occupying the whole surface, or in the best case scenario, offer a slightly more complex variable that could not fulfill our expectations.
At this point we noticed an indirect relation existed between playing an instrument and writing computer code. Researching further into this, we came across the manifesto. It originally began as a musical trend, but currently also has a visual branch. The manifesto deals with the same issues that we are interested in, mainly concerning the performance potential of electronic arts, and it arrives to our same conclusion.
sets a platform for programming visual content. Code is written and structured in way that is understandable to the machine, which in turn translates it into images. Hence, the performer is compelled to constantly act out in order to keep the image flowing in an esthetic way.
This solution is very similar to the act of playing live music, and may even be considered a return to the origins of electronic music and the ideals of pioneers such as Varesse. In the early stages of electronic music there was a keen interest in achieving a way to replicate a performance through electronic means. Since at the time the final outcome

of the execution of a musical piece was influenced by many variables, such as the musicians, the location, the conductor, the audience, etc..
We are interested in rescuing the factor of error implied in live performance, as well as its ephemeral quality. Conveniently for our purposes, under the manifesto the code produced in the act is lost, pertaining the fleeting aspect of live performance.
Another principle stated in the manifesto, is that the code should remain visible to the public at all times. Pretty much in the same way as the guitarist's movements are a visible part of the music he plays, even if the straddling fingers make no sense to the audience.
Consequently we decided to create a soft that responds to all the principles in this manifesto. We are not the first to do this, the choice of name acknowledges this fact. There is a previous software, named after the homonymous artistic movement. We chose the name Flaxus, to honor the same genesis but replaced the initial syllable for "Fla" a common suffix in software when the program in question has some kind of relationship with Flash. And of course this is the software we developed our application in. But instead of stopping there we went one step further with our software, by incorporating network capabilities, particularly the concept and atomic dispersal Internet allows.
Our software was built with the aim of reformulating live visuals. It basically allows someone to generate visual sketches by composing them live, while somewhere else on the planet another person can watch them live. Though, the contents aren't passively transmitted from one place to the other. A passive transmission would imply taking the whole data from one point to the other without alterations or major distortions, except for the unavoidable loss of quality that transmission provokes. This is the case, for instance, of radio and television transmissions.
Rather our software has as in its core a series of elements that react to audio stimuli. Thus, the visual composition is altered by the interaction with the music applied to it. We propose a shift in audiovisual practice by establishing the image as an engine of the whole experience. The different users can apply to it any music they want. Consequently, as a user writes live code, it is reinterpreted everywhere else in the world and react to the sound or music heard in that particular place and time.
As a result, visual compositions cease to be rigid and become adaptable to various circumstances. But the flexibility attained is not limited to the sort of linear transmission consisting of a transmitter and a receiver. Digital channels allow a circular performance. Flaxus makes performance a collaborative affair. Instead of there being a single performer, a spectator can act out and modify the piece for the rest of the audience (who at any time can become performers) to see. Thus, the performance becomes a group dialogue between many parties. The opposition between the performer and the audience is dissolved and can even be inverted.
The esthetic error is incorporated as a basis of the system. In each place the final perception of the execution can be completely different without altering the piece's artistic composition at all.

Grammar

The logical structure of this programming language is based on a system of modular hierarchies. They are inserted one inside of another and are solved from the inside out, clearing the unknown quantity the same way as in maths. Each structure generally has a verb as its first element and as a second one a noun that operates on the verb. The noun is sometimes followed by different adjectives or secondary verbs that operate on it. These structures are always placed between parenthesis.
Here's an example of a verb in use:
(Build_Cube)
Creates a cube
(deleteModel ACTIVE_MODEL)
This phrase is composed of a verb and a noun and it tells the system to delete the last active model, that is, the last model created or accessed.
(Build_Cube 30)
This creates a cube at a scale of 30. In this case the noun is implied.
(setVar miCubo ACTIVE_MODEL)
Here "miCubo" is the denomination for a new noun with the solved value of an

adjective.

There is also another type of structure that refers to a group, and usually its first value is an element of that group.
(math random)
In this case the system takes the random value of the math element.

The structures can also replace values within other structures.
(setPosition (math random 100) 0 0)
The verb "set position" operates automatically on the last model created, changing its position according to the three adjectives or parameters submitted, which respectively represent X Y Z.
In the example the value of X is given by a random value between 0 and 100.

Consequently, the structures may serve not only as an action, but also as a definition for a value involved in an action of greater hierarchy.

Capabilities
Flaxus is an experimental conceptual tool. It's a not high performance software. It has great educational value, in the same way as John Maeda's MIT did or as Ben Fry (Broad Institute) and Casey Reas's (UCLA Designs | Media Arts) does.
The tool allows the user to create simple 3D polygons in real time in a space that reacts to sound.
It support the use of 2D and 3D particles;
variables management and mathematical calculations;
bitmap creation in real time;
different layer blending operations;
the use of textures, fonts and gradients generated in eral time;
as well as direct operations on videos and bitmaps such as Kerneling, Convolution Filters, Bitmap Copy, Video Feedback, etc.
All of these processes are executed in real time.

Furthermore, Flaxus is a collaborative tool that allows different people to operate simultaneously on the same piece as they instantly share the code and its execution.

Technology
The entire software was built in , a tool initially conceived to develop animations which currently has a powerful language oriented to esthetic programming. Which also lets us perform real time operations on bitmaps. Moreover it can run on any platform that supports the , thus as well as being multiplatform the software is able to run embedded in a web site.

What's Next
We will begin by adding documentation and examples. Seeing as we believe that example analysis is one of the strongest and most effective ways of understanding a tool.
The software is still at a beta stage, so from here on in we'll be correcting errors pointed out by the very community that uses it.
Hence the next step is implementing a series of forums so that the community may post questions, codes or scripts.

Community
The software is offered to the community under a
Its use is free to any individual.
The code is shared for anyone to correct or modify.
We only ask that the sources and authors of the tool are clearly stated.

Non for commercial use