# Franz Hemmingcraft



## Introduction

Generative art has since its beginning been facing problems in the world of art. Questions to the authenticity, authorship and intent have been much debated since its emergence in the mid-20th century. A common debate in the study of generative art has been whether the importance of generative art lies with the process or the product made thereof.

In the making of our project we tried to answer the question: “*Which role does the viewer play in relation between code and product in generative art?”****,*** concluding that the viewers’ role is of great importance for generative art to be generative.

Our project is a movie plot generator that displays its generated plots onto a twitter account called Franz Hemmingcraft. The name and its description is used to obfuscate its non-human nature and blend in with the rest of twitter’s users.

To discuss our project in terms of theory, Frieder Nake, with his terms surface and subface will be used along with Philip Galanter, who has addressed several of the problems that generative art faces in the perspective or more general art.

## Making a digital plot writer

The program runs in node.js, which enables us to run JavaScripts in a command prompt, rather than running it through a browser window or a website. Instead, we can upload this program to a server, which means it can keep running for as long as the server is up. This is ideal, since we want the program to run independent of the viewer.

This also allows us to use the Twitter API, which requires OAuth and is only runnable through a server side JavaScript.

With node.js comes the capability to use packages, which is ‘premade’ code, that we can use rather than writing everything ourselves. In this project, we make use of a package called Twit, which is uploaded and maintained by GitHub-user ttezel.

This package comes with functions for setting up a connection between the script and Twitters API. Furthermore, it gives us functions for writing or reading through the Twitter API, which normally is done via URL requests, but with the package we are able to do it through functions that take arguments for creating the URL requests. Most functions also have a callback, which we can use in the program to interpret any errors the URL request returned.

The part of the program we wrote ourselves, is the generation, or rather assembling, of the movie plots.

For the generation we have a database, with various tables, containing the words we choose. This is a static JSON-file, which is hosted on the same server as the program.

The JSON-file is structured as following:

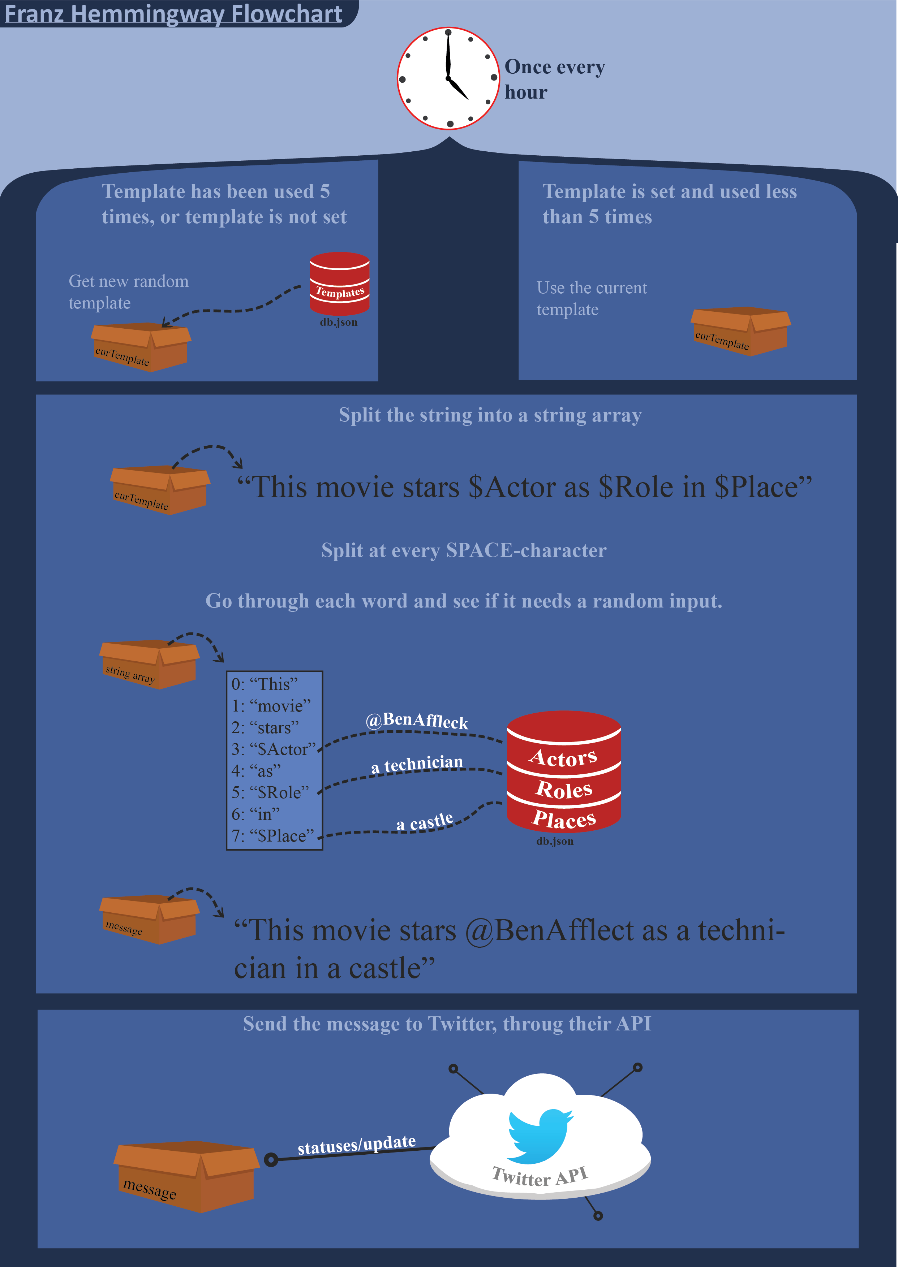
|  |
| --- |
| Templates: [ string ] |
| Roles: [ object { DefSingluar: string, IndefSingular: string, Plural: string} ] |
| Actions: [ object { Verb: string, ThirdPersonSingular: string } ] |
| Adjectives: [ string ] |
| Places: [ string ] |
| Settings: [ string ] |
| Objects: [ object{ DefSingluar: string, IndefSingular: string, Plural: string} ] |
| Actors: [ string ] |
| Genres: [ string ] |
| Events: [ string ] |
| Concepts: [ string ] |
| Names: [ string ] |
| Surnames: [ string ] |

A generation of a message, starts with grabbing a template from the JSON-file and splitting the string up into an array of strings. This is done with the JavaScript function str.split(‘ ’);. The criteria of the functions, chooses where to split. In this case, we split the template up at every space-character in the string.

An example of a template, from the JSON-file:

*“$Actor stars as $Role:IndefSingular that fall in love with $Role:IndefSingular . Everything seems perfect until $Event”*

The message is created in a single string, in which the words are inserted one by one. If a word doesn't call for a random input, it is just parsed to the message string. The only check we make for these, are commas and punctuations, since we need spacing in between those in the template.

We created our own syntax for inserting random words. If there is a need for a random input, we write a dollar sign and the table we want to grab from. Some tables require additional information, for grammatical purposes, which is inserted after the table-name, separated by a colon. So $Actor, looks for a random entry in the table Actors, and $Role:IndefSingular looks for a random entry in Roles, with the criteria of it being an Indefinite singular.

After assembling the message, we check the length of it. If it is above the 140 characters that twitter allows, we generate a new message, if not we post it to twitter.

The handling of timings is done with the JavaScript function setInterval, which can run a function in intervals set in milliseconds. Here we run the generateMessage() function, which is our main ‘loop’, in which the assembling of messages is happening and where we call other appropriate functions.

## Expressing the Aesthetic of Code

Philip Galanter is a respected professor in generative art, and he lists different options, which characterize generative art theory in the book “A Companion to Digital Art”[[1]](#footnote-1) in chapter five “Generative Art Theory”:

“*Generative art is art that uses randomization.*

*Generative art is art that uses generic systems to evolve form.*

*Generative art is art that is constantly changing over time.*

*Generative art is art created by running code on a computer.”[[2]](#footnote-2)*

Our project is correlative to several of these options. Firstly, our movie plots from the Twitter Bot is built around randomization. The different variables are picked out from a JSON-file and randomized into a prewritten template.

Secondly our product has evolved its materiality through textuality by generic systems. In addition to this, our product changes its form over time. New movie plots are tweeted every hour, where the variables are changed, and the templates are changed every 5th hour. These generic systems, which lies beneath the twitter bot, are created from established lines of code on our server.

Frieder Nake takes on the problem of code without product and vice versa in his speech “*Art that makes itself*”[[3]](#footnote-3). He introduces two terms: *surface*, the perceived part of a program by a viewer; *subface*, the code behind the program unperceivable for a viewer. These two terms in generative art are opposing to each other, but one cannot be without the other according to Nake.

Subface without surface is simply unexecuted code. Though able to attain certain aesthetic qualities, it does not count as generative art. Surface without subface is not generative art, as generative art emerges from the process i.e. subface.

The understanding of this statement from Nake is that the generativity, i.e. the fundamental process and subface, must be perceivable for the viewer. If the program is not using the computation, the viewer will never perceive it as generative art.

In his article, “The Aesthetics of code”[[4]](#footnote-4), Geoff Cox visits the relationship between code and its product, in which he states that these two are inseparable. In our project, we tried to bring this relationship to the viewer of the product. The use of the same template makes it seem more dependent on an algorithm making it more computational and bot-like, while still making different outcomes from the same template.

The intention of the project is partly to challenge people's perception of Franz Hemmingcraft. Even though the bot has different human features such as humor and grammar mistakes, viewers should still be able to discover that Franz Hemmingcraft is a bot.

## The Project as critique

When working with generative art, another aspect which occurs is the question of authorship. Philip Galanter refers to McCormack et al. who questions the relationship between *“authorship and agency, creativity and intent.”[[5]](#footnote-5)* As Galanter states, these features are usually not questioned and inherent in non-generative art, but comes to question when the humans behind the product is not the producers.

In correlation to our project, it is possible to discuss where the element of authorship lies. Who have the rights as author of the program: Twitter; as it is the website that is the surface of our products, the program; as it generates the products, or us as programmers; as we made the bot that makes the products? Furthermore, the process of the bot have required tutorials and help from external sources such as Daniel Shiffman, who then could be considered a co-creator of the products generated. Our bot contributes to this discussion as it itself is given a role of a creator.

Galanter questions the note of creativity and thereby authenticity[[6]](#footnote-6). The bot can be questioned on the same. Opposing the unanswered question of authorship, the bots authentic creativity is a simpler matter to discuss. As it blindly picks a template and fills in the spots with random words from a database, it resembles the way Dadaism practitioners would cut newspapers into their individual words, draw them from a hat, and call the arrangement a poem[[7]](#footnote-7). The idea in itself is creative, but the execution is less so. A simple bot could do it.

A critical element which must be taken into consideration is the description and name of the bot. The name and info text of the bot, does not reveal the fact that it is a twitter bot though some suspicion might arise. Viewers should be challenged and interested through the tweets from the bots, as their characteristics are homogenous across many of the tweets on the profile e.g. the structure for the plots looks the same but the words are differentiating between the different plots. Afterwards it should be possible for other people on twitter to perceive and understand the twitter account as a bot. Thereby it would encourage people to be more interested in the code behind the twitter bot. As viewers are introduced to the bot, they will see the generativity of the products from the bot. As stated earlier, the surface is nothing without the subface in generative art, thus when the viewer perceives the generativity of the program and product, the product becomes generative to the viewer.

## Conclusion

To conclude: we introduced this project by asking the question: “*Which role does the viewer play in relation between code and product in generative art?”.*

The viewer plays an important role in this relationship, as without his acknowledgement of the art as generative, the generative aspect becomes obsolete, thus the art ceases being generative. For generative art to be perceived as generative, it must have a degree of transparency to show the viewer the generative process. In this project, we focused on trying to let the viewer discover this transparency themselves, in hopes that it would peak their interest, as to develop an interest in the code itself.

## Literature list

Geoff Cox, Alex McLean and Adrian Ward, The Aesthetics of Generative Code

Link: <http://generative.net/papers/aesthetics/> or <https://goo.gl/wJOsiQ>

Last accessed: 16-05-17

Frieder Nake, Art That Makes Itself Symposium 2015 | Frieder Nake

Link: <https://www.youtube.com/watch?v=ICOs_8pjOIc> or <https://goo.gl/evNWcV>

Last accessed: 16-05-17

Christiane Paul, A Companion to Digital Art, chapter 5, Philip Galanter, “A Generative Art Theory”, Wiley, 2016

Link: <http://cmuems.com/2016/60212/resources/galanter_generative.pdf> or <https://goo.gl/fC6FPw>  
Last accessed: 16-05-17

## Related projects

Magic Realism Bot

Link:<https://twitter.com/MagicRealismBot?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor> or <https://goo.gl/RWGSIH>

Last accessed: 16-05-17

Note: This was one of the sources of inspiration for our project.

1. Christiane Paul, A Companion to Digital Art, Wiley, 2016 [↑](#footnote-ref-1)
2. Christiane Paul, A Companion to Digital Art, chapter 5, Philip Galanter, “A Generative Art Theory”, Wiley, 2016, p. 150 [↑](#footnote-ref-2)
3. Frieder Nake, Art That Makes Itself Symposium 2015 | Frieder Nake, link: <https://goo.gl/evNWcV> [↑](#footnote-ref-3)
4. Geoff Cox, Alex McLean and Adrian Ward, The Aesthetics of Generative Code, link: <https://goo.gl/wJOsiQ> [↑](#footnote-ref-4)
5. Christiane Paul, A Companion to Digital Art, chapter 5, Philip Galanter, “A Generative Art Theory”, Wiley, 2016, p. 166 [↑](#footnote-ref-5)
6. Christiane Paul, A Companion to Digital Art, chapter 5, Philip Galanter, “A Generative Art Theory”, Wiley, 2016, p. 169 [↑](#footnote-ref-6)
7. Geoff Cox, Alex McLean and Adrian Ward, The Aesthetics of Generative Code Link: <https://goo.gl/wJOsiQ> [↑](#footnote-ref-7)