Top-Up / Bottom-Down- & Glitch-Methods in Architectural Design

Sebastian Gatz b.a. arch.

The Royal Danish Academy of Fine Arts
CITAstudio 2016, 22nd December 2016
sebastiangatz@googlemail.com
30,027 characters

Abstract

This essay proposes new possibilities for site specific architectural design production for additive manufacturing by investigating the potential of combining bottom-up emergent systems – namely cellular automata and simulations of physarum polycephalum - and top-down interactions. The implementation of semi-controllable data-glitches - as a design driver for new forms of ornamentation - is tested.

BACKGROUND

New possibilities of architectural production methods, namely additive- and robotic-fabrication, ask for new modes of architectural design development. State of the art fabrication methods do not rely on human handwork anymore. Therefore, an integrated digital work flow - from design to manufacturing - enables new ways of designing: away from using the computer for manual drawing, towards architectural modeling with integrated semi-autonomous emergent systems like agent based modeling and new combinations of generative algorithms.

Those systems, more and more likely to be manufacture-able in bigger scales with additive manufacturing, have the potential to react on, one hand, site-specific (for example by using point cloud scans) - small and big-scale phenomena and, on the other hand, to develop new ornamental qualities.

While the use of parametric tools - widely favored in the industry of computational architecture - normally is a successive intentionally deformation of an input geometry, coding with its complex possibilities of internal data flow manipulations, allows for a more emergent bottom up approach of form development and - because of its non-linear information flow - creates a potentially broader space for formal / spatial discovery and is therefore used for the following investigation.

The bellow described methods and interactions of cellular automata (CA), simulations of physarum polycephalum (PP) and glitch art, are programmed on the pixel-level - all having the pixel, as the smallest building block we have in current architectural design development and representation, in common. The three-dimensional equivalent of the pixel - the voxel - is the smallest physical entity for 3D-printing and the smallest design scale we are used to work with and is therefore - algorithmically and scale-wise - prioritized in this work.

The project proposes a three-dimensional CA-

PP growing algorithm, which consists of multiple image layers - to create a three-dimensional volume of a 3d-printable architectural artifact. The outcome is a collection of sections through the artifact in a CT-scanner-like manner which could be translated directly into g-code for 3d-printing - colors of the pixels could inform a multi-material three-dimensional print.

During the emergence of the CA-PP patterns the designer is able to intervene in the recursive process by regulating the final output's ornamental qualities by steering pixel-sort-relations of the different CA-PP layers. The internal bottom-up / top-down loops of the PP and the CA, the top-down influence of a site-specific point cloud scan and the user interactions result in a here called top-up / bottom-down design method - showing the potential for new digital design-to-production work flows.

To test the overall work flow, a minimal bivouac is considered, based on a site specific point cloud - taken in the Swiss Alps. The proposal negotiates spatial modes of enclosure, structure, nocturnal / diurnal skin behavior and ornament. As a starting condition, the CC-PP algorithm is informed by special points of interest, specified in the point cloud (like desired ground- or tree-connections).

CELLULAR AUTOMATA

CA are computational models used in computer theory, biology and related scientific fields to study dynamic systems with local cell interactions - basic models for information transfer. CA differ in their dimensionality, in size and structure of their local cell neighborhood and are either discrete or continuous in their time steps and cell states.

One of the simplest – Conway's Game of Life (named after John Horton Conway) – is a two dimensional, two cell state (black, white) and time discrete CA with a so-called Moore neighborhood. A Moore neighborhood consist of eight neighbor cells surrounding the cell they interact with. Each cell in the two-dimensional lattice and its neighbors are iteratively checked for their cell state. Depending on the predefined rule set each cell will change its state in the next iteration of the code. In that classic example the rule set consists of four rules: under-population (a living cell with fewer than two living neighbors dies), over-population (a living cell with more than three living neighbors dies), reproduction (a dead cell with exactly three living neighbors becomes alive) and transition to next generation (any living cell with two or three neighbors lives on to the next generation). Those simple rules are capable to generate a set of basic shapes and behaviors which are constantly interact with each other to self-organize the whole lattice.

The mainly used rule for the here described studies could be specified as "majority-wins"-rule: either in two-dimensions or three dimensions a certain number of cells (pixels) and their 8 or 26 neighbors gets randomly checked. Depending on the CA's starting condition there is a certain amount of discrete colors (states). For each random check of a pixel and its neighborhood the number of pixels with the same color are counted. The majority color will be the color of the checked center-pixel in the next time-step. This rule creates discrete voids out of a noisy starting condition.

Figure 1 "majority-wins"-rule creating discrete voids

Weights for x-, y- and z-position of the pixels, their distance to the CA-center or other spatial points of interest and / or time-step related weights, additionally steer the overall behavior of the CA and their resulting discrete voids, their spatial formation, position and size.

If not stopped, the system tends to grow infinitely and is likely to favor one color over the other.

The overall growth behavior of the CA can be steered by implementing rules, related to pixel-positions or - potentially - more physics-related rules, like mathematical equations for material-specific R-values of wall-pixels or structural optimization formulas.

The use of a CA with combined physics- and design-informed rules has the potential for a more holistic bottom up emergent growth of the architectural structure rather than the classic top down imposing of it.

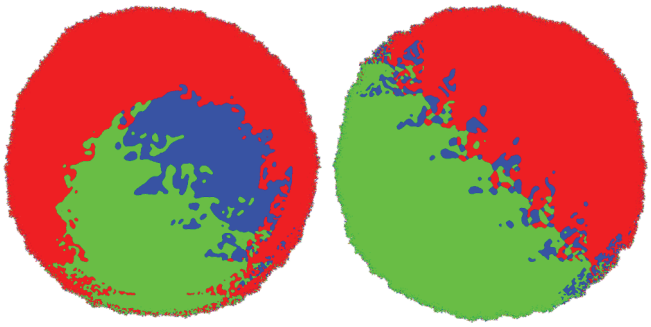

Figure 2: "majority-wins"-rule with different additional rules for x-, y- and center-distance-position of the pixels

Global "knowledge" of the cells are informed by agent-based vector math, namely stigmergic slime mold simulations (PP).

PHYSARUM POLYCEPHALUM

Another computational model partially related to the above described system of CA is the digital simulation of the behavior of physarum polycephalum, a single cell slime mold with some - not yet fully researched - adaptational abilities, like solving shortest path problems or even memorizing information about its environment.

Researchers like Jeff Jones and Andrew Adamatzky from the University of the West of England are exploring the computational potential of those systems in their International Center for Unconventional Computation.

The simplest version of a computational model of PP is simulated by having individual agents with their local world-interactions (similar to the CA) on two maps (two- or three-dimensional): a "pheromone"-map and an agent-map (both mainly for visualization). For the here described investigations a simplified version of modeling the behavior of PP, described in the paper *Characteristics of pattern formation and evolution in approximations of physarum transport networks* (Jones, 2010), is used.

The earlier mentioned pheromone-map serves as the means for indirect information transfer between the agents. There is no direct interaction between those entities.

The pheromone-map – here a metaphor taken from nature to describe a gray-scale value-map – consists of (in the beginning randomly assigned) pixels with color values between 0 (no pheromones) and 255 (many pheromones). Each pheromone-pixel transmits its pheromone-value (PV) partially to its neighbor pixels (similar to the majority rule of the CA) and generally reduces the PV for each time step.

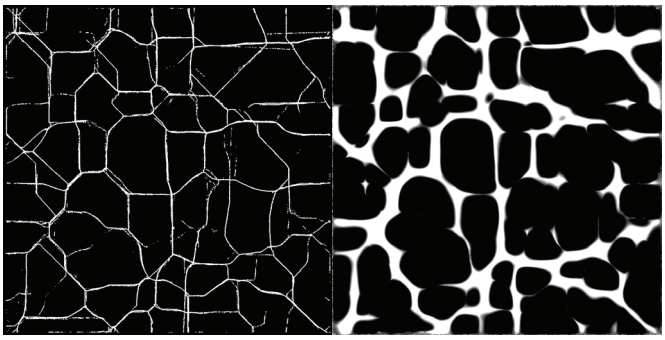

Figure 3: agent-map and pheromone-map

Each agent has a position on the agent-map with a specific orientation (in the beginning those are assigned by random). The agent is then randomly (for the first step) moving on the agent-map, while sensing the pheromone-map 10 pixel in front of it and 45° to the left and right of it. For every time step each agent makes a pixel-step in the direction of the highest PV of the three previously checked pixels (L, M, R). At the same time the agents release a high PV on

the previous position (on the pheromone-map). If a position on the agent-map is already occupied, the agent rotates 180°.

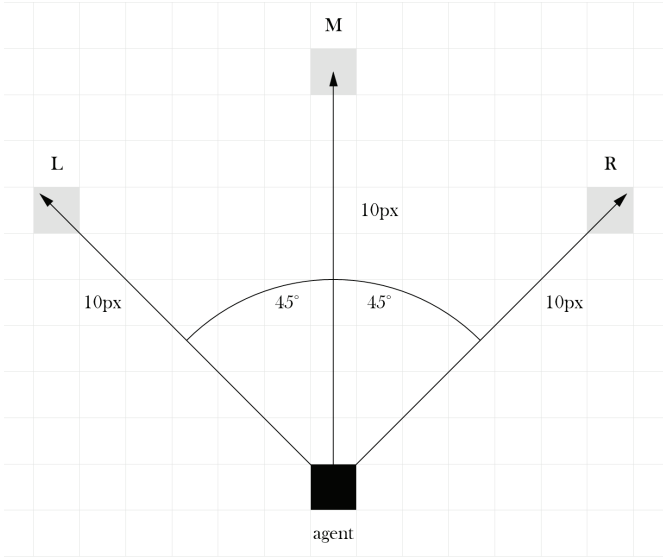

Figure 4: agent sensing

This indirect- and local-behavioral system finds shortest paths between high PVs, it tends to minimize its overall occupied area.

The PP simulation with its tendency to shrink and the majority-rule-CA which grows infinitively are able to find some kind of equilibrium when combined. The PP - in itself a top-down-bottom-up loop - informs the CA rules: the CA growth is restrained by the position of the PP-paths. Additionally, the CA writes on the pheromone map of the PP to reduce the shrinking behavior of the PP.

The pheromone map starts - as described earlier - with random distribution of PV.

However, during the simulation the three-dimensional CA-PP algorithm is also informed by a point cloud scan with additional predefined points of interest.

SITE-SPECIFICITY

The three-dimensional point cloud scan gets horizontally sliced - like CT-scans - according to the desired voxel resolution of the CA-PP algorithm (here one voxel represents roughly 3mm). The scan resolution here informs the final voxel-materials on a similar scale. The resolution could be much higher, with increasing computational power. The points of interest - like desired tree- or ground-connections - are constantly releasing high PVs on the pheromone-map, while the CA is informed by the scan itself - the xyz-coordinates and the colors of the scanned points. Brownish pixels (tree bark) are triggering the growth, while greenish pixels (ground) are ignored. Accordingly, the desired growth between different tree branches is encouraged.

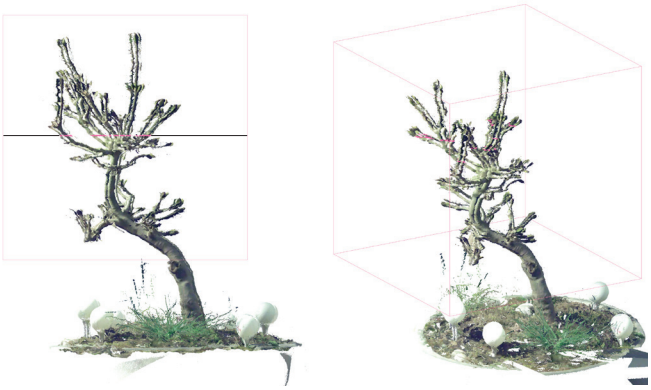
Figure 5: point cloud sectioning, box represents final volume

Because of the nature of the PP and the predefined points of interest, the PP creates some kind of main scaffolding, the CA is able to grow around it.

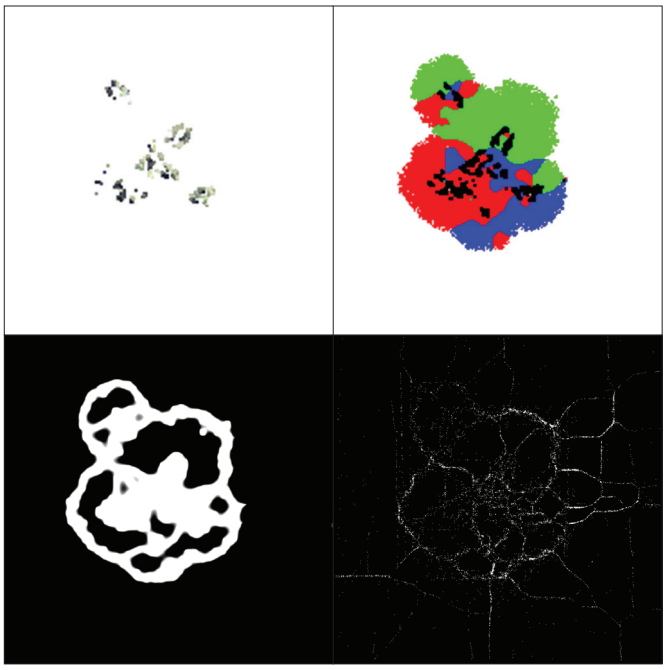
Figure 6: point cloud section, CA with three discrete voids, pheromone-map and agent-map

The last step of form generation - and the first step towards 3D-printing fabrication data - is to combine the site-specific point cloud scan, the pheromone-map, the agent-map of the PP and the CA-map, by allowing the user to steer a glitch pixel-sorting algorithm, which combines the maps in a semi-destructive / ornamental manner.

GLITCH

Depending on where you have your eyes and ears while roaming through the internet and today's pop cultural, there is a huge possibility that you once in a while come across one of those wired aestheticized colorful, noisy and corrupted images, videos or sounds: all coming together under the established term "glitch art". Either – as a more current example – Kayne West's 2008's music video to *Welcome to Heartbreak* or – to show since when creative folks played with that kind of notion – art like Nam June Paik's 1965's *Magnet TV.*

A glitch is mainly an (intentionally produced) malfunction of a (computer) system. Some people – conscious or not - feel attracted by those failures, either for aesthetic reasons or (more on a meta-level) because of the possibility to get a brief glimpse behind the scenes: the glitch reveals something about the shown which otherwise would be hidden. A corrupted sound file for example distracts us on one hand from the information (the music) but on the other hand tells us something about the structure behind the sound: how is the data stored, how is it processed, etc.

The glitch will not tell us exactly how things work but it reminds us that there is not just the polished surface – which industry and advertising wants us to see.

In that sense, a glitch gives us back some power and autonomy about our (nowadays very often digitized) environment. Some people like Nick Briz (*THOUGHTS ON GLITCH[ART]v2.0,* 2015) and Rosa Menkman (*The glitch momnet(um),* 2011) even translating and mixing the concepts of glitch with topics like life style and politics.

The "looking behind the scenes"-idea taken a step further is seeing the glitch as some kind of counter movement to today's fast, superficial, and polished (fake) social media world. Accordingly, it's maybe close to an old Japanese concept, called wabi-sabi – a concept of perceiving beauty in old, unfinished, or even broken elements. One can see parallels in that idea in the digital noise of the glitch, although its matter is made from bits and bytes.

The mold-free use of digital fabrication - here additive manufacturing - enables the designer to formally express what otherwise would be uneconomically. Time- and material-wise a 3D-printer makes almost no difference between a printed ornament or a cube. While handmade or conventionally produced materials are capable of telling stories about their development or treatment - like brick-forms and -textures informed by the materials position in the oven during the firing process - new generations of 3D-printers with increasingly high resolution are on the way of loosing this kind of meta-information about process.

As mentioned earlier the glitch is capable of revealing bits about the process of making or the underlying information. By implementing it in the design process or even possibly in the manufacturing process it is arguable that the intended form-corruption can be part of a new ornamental language, which lies on the other side of the shiny-white-endless-surface digital architecture spectrum. Additionally, the glitch here - partially because of abrupt changes between different layers - reveals how the architecture was designed and produced in horizontal steps - and therefore is able to transfer something of the process in the physical manifestation.

The glitch art community distinguishes between two types of glitches: the wild (the found "real" glitch) and the domesticated.

Jeff Donaldson and Antonio Roberts founded glitch safari, an online community for wild-glitch

archiving, in 2012. The homepage shows endless collections of broken TVs, LED screens and other primarily public media / information devices.

Wild glitches - at least in the beginning of the "movement" - were basis for the reproduction of domesticated glitches. The main characteristics found in both, are unnatural colors, general noise, linearity and compression artifacts. Each of those aspects are attributable to the underlying technology: the way digital colors are defined, media is compressed and decompressed or pixels are stored.

Domesticated glitches are often either created by reproducing the error which generated the wild equivalent in the first place (for example wrong compression or file treatment) or by generally and more freely collaging images with pixel-algorithms to archive similar visual qualities and aesthetics.

After some initial tests with corrupting data-types and file formats the use of algorithmic approaches - which are mainly based on defined rules for pixel-sorting images, rules similar to those programmed in cellular automata - were identified as the most productive for the here described exploration.

Both methods - glitching and CA - have a lot in common, but they are coming from different backgrounds: one more scientific while the other one is more pop-cultural.

INITIAL EXPERIMENTS

First experimentations were based on "classical" methods like data-moshing, bit-shifting and tests with wild glitches. Data-moshing for example is the changing of file extensions and the discovery of those files' behaviors when opened or treated in programs not made for them. Bit-shifting (although just loosely connected to the glitch-scene) is a computational method were bits directly are manipulated to change – for example – the order of pixels in an image. Other techniques which were tested are related to scanning-methods, hexadecimal number manipulations, pixel-list shifting and finally CA-related pixel-sorting experiments. Depending on the glitching technique visual results are either subtle or drastically.

The first test which was investigated, was scanning images with an overhead scanner. By rapidly moving the images while scanning them, one is able to get distorted "wrongly" colored versions of the initial picture, a first glimpse into the aesthetic qualities found in glitch art in general. Next to the change of the overall form, the originally black and white images obtain additional colorful schlieren around the dark areas of the picture. Assumingly, the different color channels of the scanner are working slightly offset in time, by that creating this kind of overlay. Even though produced intentionally, this kind of glitches are more likely to fall in the category of wild glitches. They show interesting and partially steerable deformations but are hard to automate algorithmically and therefore are not further considered for the overall process.

Figure 7: "wild" glitch made with overhead scanner

Following tests were made to discover the potential of hexadecimal systems for glitching and low-lev-

el data manipulation - even though not used very often in the glitching community. The here shown example translates strings (text) in hexadecimal code and finally into color.

Each letter of the text - here randomly copy-pasted - is converted into a two-digit combination of letters and numbers. Three of those two-digit combinations are then used to create a color value (which in hexadecimal code is made out of 6-digits). Each color is then represented in a colored pixel, which are subsequently ordered in a row and finally in columns. For example:

- string: “halloo”
- hex: 48 61 6c 6c 6f 6f
- color pixel 1: #48616c
- color pixel 2: #6c6f6f

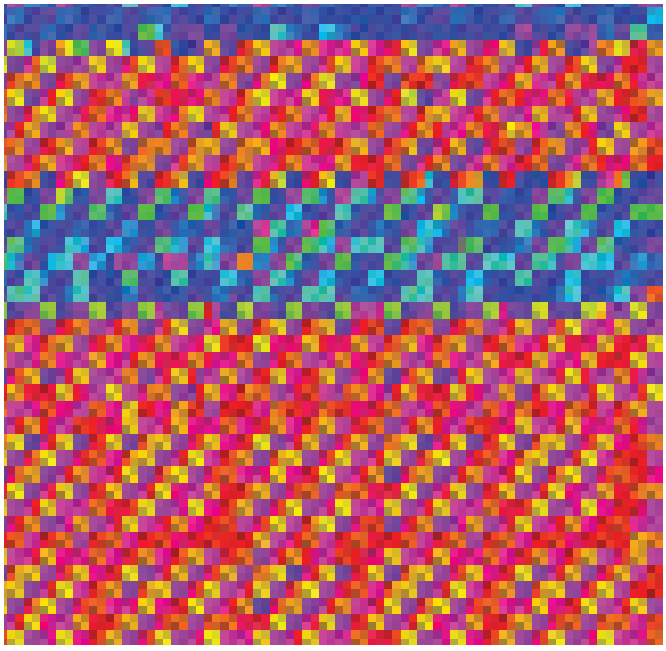

Figure 8: string to hexadecimal to color

The resulting image is basically some kind of data-visualization and quickly asks critically for a useful implementation in the proposed work flow.

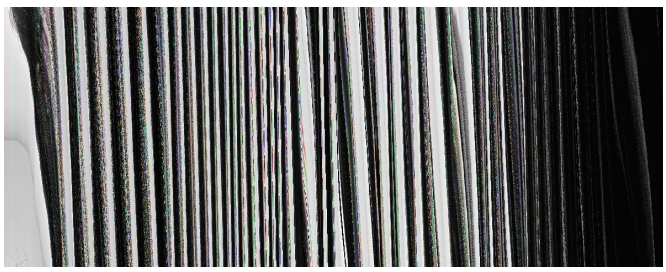

Figure 9: data-moshing an image

Additionally made tests - mainly for the purpose of further discovery of the overall aesthetic-language of glitches - were related to data-moshing (heavily used in the community): opening files - for manipulation - in programs not intended to work with them. In this case an image (jpeg) opened in a text editor and saved again as a picture. This process could be automated but shows drastic change in the image’s appearance, which could end up in unusable changed images, instead of using the glitch as a more subtle ornamental layer for the overall design process.

An experiment made with binary-shifting - equally loose related to glitch as the hexadecimal operations, more related to general computation - is building up on the idea of manipulating data at a low level: numbers stored as binary numbers - for example 7 = 0111 (8,4,2,1) - allow to be manipulated with so called bitwise operations. The binary number 0111 shifted to the left (<<) becomes 1110, to the right (>>) it becomes 0011. The, in a for-loop (iteratively) shifted values, were used for color, size and position of points in a tow-dimensional image. The operations did not show any unexpected glitches, the result is mainly an exponential number addition or subtraction of the starting number.

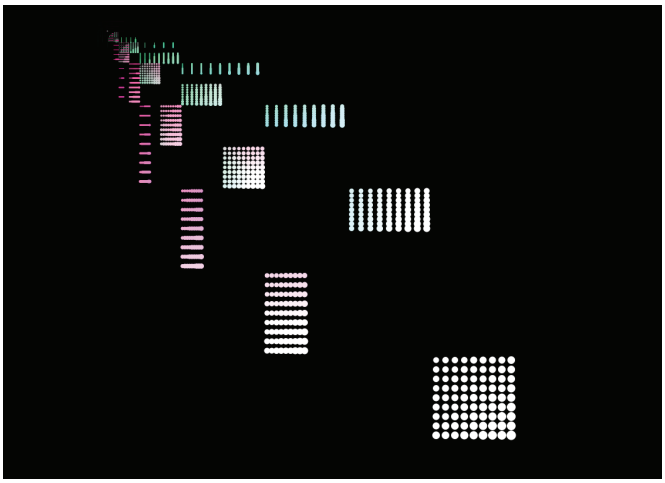

Figure 10: binary bit-shifting experiments

A method, here called list-shifting, slowly enters the territory of the pixel. Pixels are generally stored in one-dimensional lists - instead of a two-dimensional, what one would might assume. If represent-

ing an image with 120,000 pixels, with an initial width of 300 pixel and a height of 400 pixel, in an image with 400 x 300 pixel, one will get results like the image bellow.

Figure 11: pixel-list-shifting

The following described methods are algorithmically similar to the ones used for CA and are slowly conceptually bleeding into this field. Forms either emerge from local pixel-interactions or are defined through top-down pixel-movement rules.

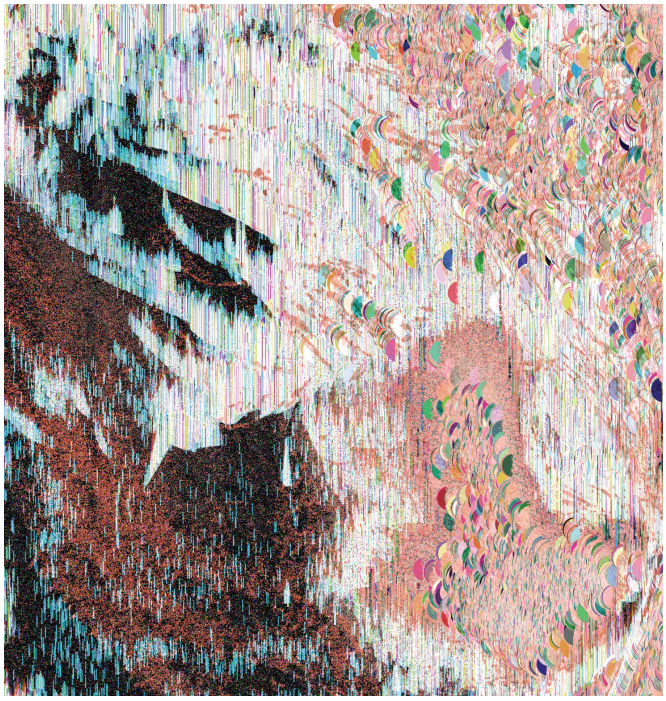

Figure 12: pixel-sorting

The following explained model is a recursive loop, with a bottom-up analysis - in a CA style manner - and a top-down pixel-manipulation. Randomly checked pixels get compared to their eight neighbor pixels. The eight pixels are added together and divided by eight. If this averaged color-value is far away from the center's color-value, an edge in the image is recognized. This edge recognition triggers top-down pixel-offsets, color changes or line drawings. The recursive algorithm generates useful - for the here investigated work flow - and partially controllable pattern.

The last method, which finally is being used in the overall work flow, is an adaptation of the previously described pixel-sorting algorithm. Instead of sensing local neighborhoods in a two-dimensional image, the rule set is based on differences in three-dimensionality. Pixels are randomly checked and compared with the pixels bellow and above them - in an image stack. In this case the pixels getting sort by color in the z-direction: colorful pixels are "floating" to the top of the image stack, dull pixels are "sinking" to the bottom. This method is used in the final model, except its comparing the pixels of different stacks with each other - from "parallel" boxes (pheromone maps, agent-maps and CA-maps are compared for each layer in z-direction).

Figure 13: z-sorting

Project related, the right ratio of destruction and emergence of interesting patterns are found it this method.

The notion of pixel-sorting algorithms corresponds to the manipulation of the image's smallest building block (the pixel) and the printers smallest printing resolution (the voxel). By inventing rules informed by pixel color, -location and -neighborhood, it is possible to change the structure of the overall layer, which finally reveals process in the overall artifact.

The intervention of a site-specific layer-based combination of CA, PP and pixel-sorting glitch algorithms, gives possibilities for new ways of design production for additive manufacturing, in a top-down / bottom-up manner.

FINAL MODEL

The top-down approach in architectural practice is a common method used for design development. Overall design ideas are imposed in a top-down manner guided by the will of the architect in charge. Projects are iteratively formulated through manually alternating between hand-sketching and computer-aided-design drawings. On one hand the benefits of those work flows is the (supposedly) full control over the project. On the other hand, it is a very time consuming process with no possibilities for the emergence or discovery of unintended spatial or formal qualities.

The other approach, less seen in architectural practice but surely rising in architectural education, is the development of architectural form using bottom-up systems. Those emergent systems – or the way they are used – are often criticized for their inability to cope with context and design intention. Very often those systems are used in pre-architectural designs in academia - mainly as uncontextual experiments.

Both systems have their strengths and weaknesses, but have a real potential when combined and used in a single recursive process.

The proposed recursive process combines a list-based and pixel-represented three-dimensional CA-like emergent system with a user interface for intervening, namely pixel-sorting - enhancing the emergence of the overall system, as a design layer for revealing information about the design- and the production-process.

Instead of using a completely visualized tree dimensional voxel representation for the process, a list-based system is introduced, to minimize calculation and iteration time. Each layer in z-direction (pixel-height) gets a list containing all pixel color values – for x and y-direction - as numbers. Only the current operation layer is rendered on the screen - different layers of the CA-PP algorithm are chooseable. The z-layers are stepwise iteratively loaded, manipulated and updated again.

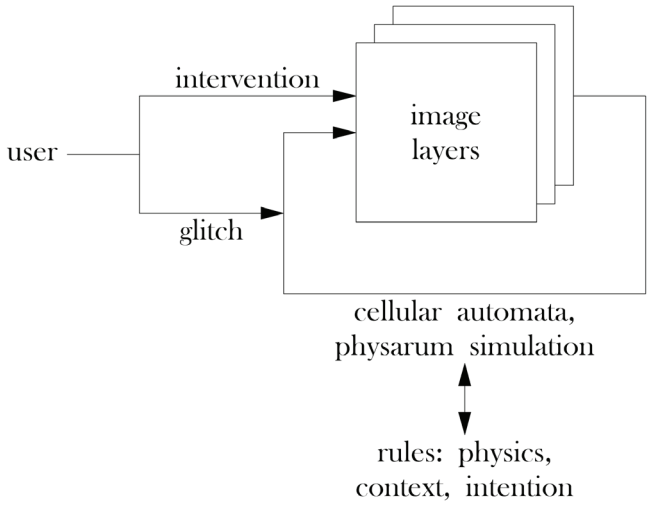

Figure 14: general data-flow

The glitches are introduced before finally saving the frames. Different combinations of pixel-sorting between the different layers (pheromone-map, agent-map, and cellular automata map) can be steered through user interaction: pressing different keys while the algorithm is running, allows for the change of the layer-weights.

The final output of the combination of the different layers will be the basis for the 3d-printing g-code development. The glitching allows for additional information for a multi-material print. The different layers influence each other on a formal-basis - by pixel-movement - and on a color-basis - by creating RGB-values from the layers.

The formal glitching is related to the desire to show the production- and design-process, while the color-glitching is used to inform color and material-properties - like rigidity and translucency - of the final artifact. The translucency changes - as mentioned earlier - the diurnal and nocturnal appearance of the final bivouac.

The images for the pixel-sorting process are taken from the final four stacks of images: site-map, PP-map, agent-map and CA-map.

The PP-pixel define the more rigid areas of the shelter, the CA the softer. The glitching informs surface-finish and color / translucency.

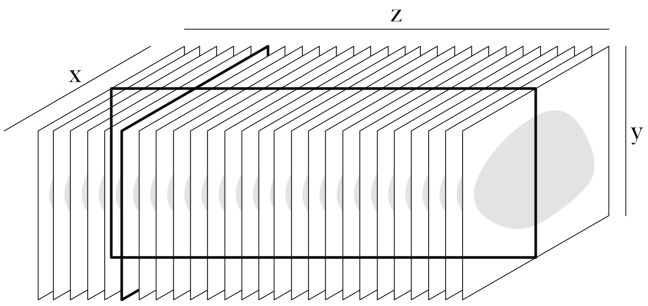

Figure 16: image-stack

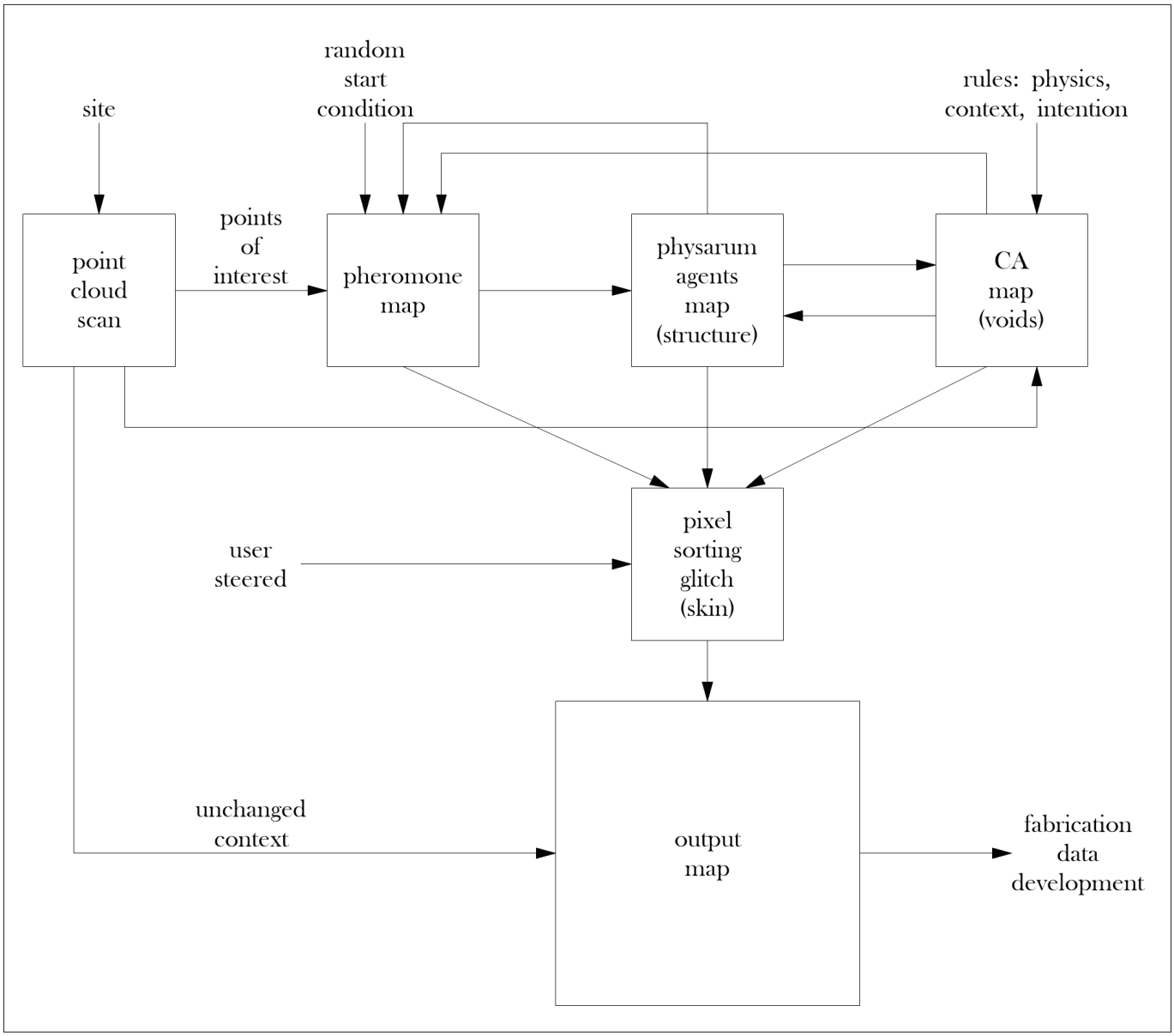

Figure 15: overall information-flow

ORNAMENTAL GLITCHING

The final layers - later used for g-code generation - are divided in two different outputs: one informs the material properties (rigidity) and the other the color and translucency of the final 3D-print.

The outside edges of the CA-voids are used, in the first step, to inform the soft-material ornaments of the bivouac (gray color, Fig. 17). The pheromone map pixel-values are affecting the offset, the user the direction, of the hair-like skin. On the outside, those soft-material arms, functioning in a roof-like manner. The overlap is providing minimal rain protection. Inside, those arms provide nest-like material for sleeping-areas. The harsh spikes seen in the material-map are repositioning themselves, due to gravity, when produced.

The hard-material (black) areas of the material-map are informed by the agent-map of the PP: the fine dots from the agent-map are used in an additional growing algorithm - to grow bigger in size. The time of the growth is informed by the pheromone-map. The dark areas in the material-map are "floating" in 2D but are connected in the 3D-stucture.

White represents unprinted areas.

The different skin-properties (overlap of soft-material) potentially generate different micro-climates in the overall structure.

The color-layers are produced by additional color glitching, which could be more subtle - i.e. for camouflage for the proposed bivouac - but are here shown in an intensive manner. The gray -and black-material-pixels of the material-map get colored by combining the information of all four maps (pheromone, CA, agents and site) and an additional color-map. The RGB-values of the pixels are later used for defining different translucency-values for each voxel. Accordingly, the bivouac has different intensities of internal brightness. Additionally, the voxel-translucencies create different appearances of the overall structure in day- and nighttime. At night the structure "glows" when illuminated from the inside, at day the inside gets illuminated by natural light from the outside.

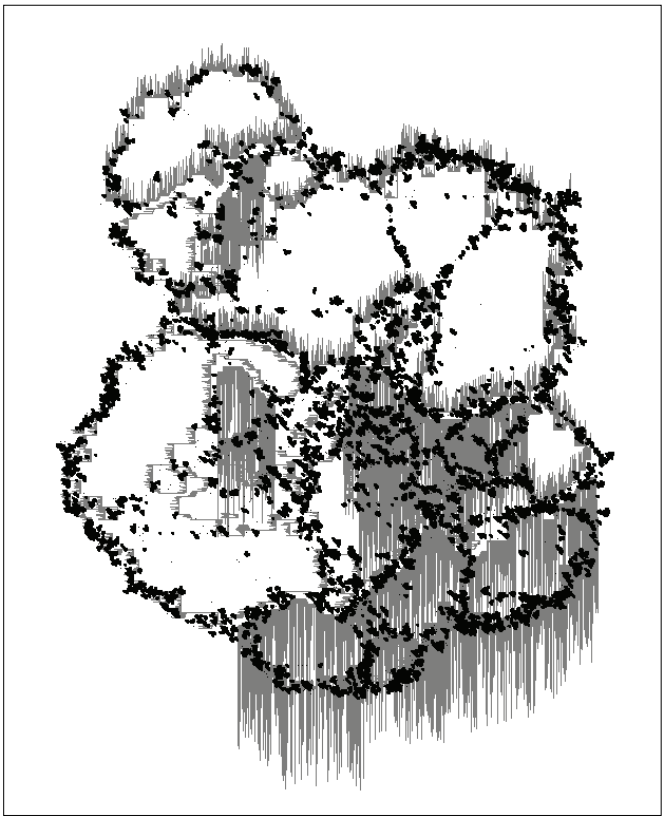

Figure 17: material-map

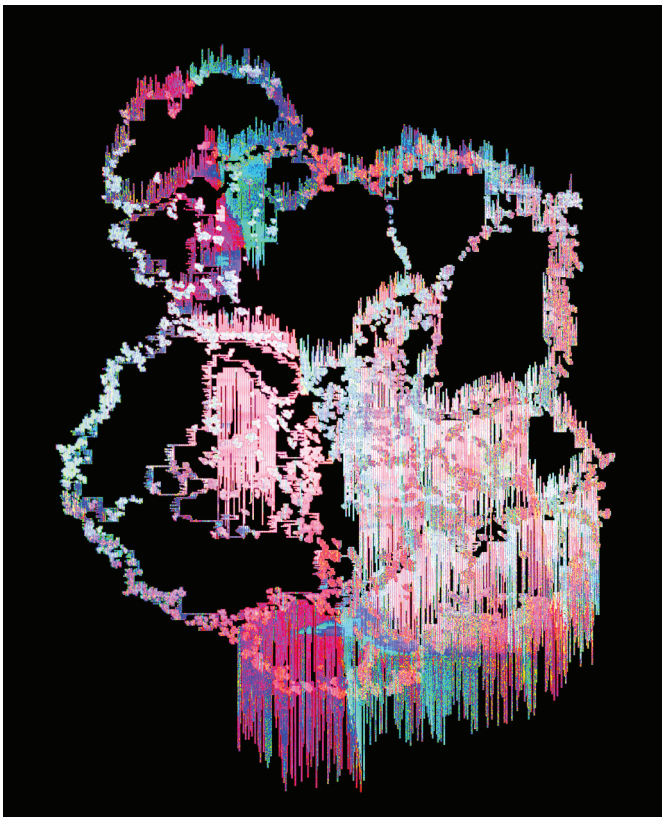

Figure 18: cleaned color-map

The general assembly-logic follows the glitching-theme of the production process, therefore the here proposed tectonic system must be seen as a first step of an experimental setup, instead of a finished construction idea.

The overall structure is vertically divided (in the same direction as the printing layers) in multiply parts. The part sizes are dependent on the minimal amount of layers needed for each element to be stable for itself.

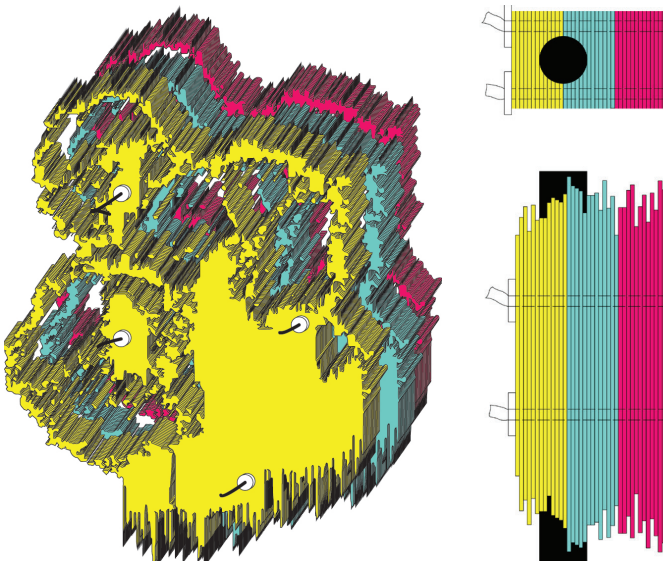

Figure 19: glitch assembly-logic around tree. left: isometric-, right: top- and side-view.

The parts are threaded on steel wires, embracing the trees who initialized the whole design process in the first place, and then brought into position (for example with some kind of wooden scaffold). When the elements are in position the wires get tensioned and locked at their ends. It's expected and desired that, after removing the scaffolding, the parts will slip partially away from their perfect fit. This will create an additional, for the inside, spatial- and, for the outside, more formal-glitch.

CONCLUSION

The here proposed design-to-production work flow shows first steps into a new world of design-methods, useful for future production processes like additive-manufacturing. The formal outcome is quite experimental and includes the personal desire to implement glitching, as an speculative ornamental driver.

The here used site-specific CA and PP show, that they, when used in combination, are capable of informing spatial and structural aspects of architecture in general.

The idea of a voxel-based design space allow for much higher degree of material-difference in the overall structure or artifact and the list-based approach of those voxel-interaction calculations make it computationally possible - even on personal computers. Nevertheless, the desired user-interaction starts to make more sense with higher computational power.

The overall data-flow shows ways of combining top-down and bottom-up design methods.

The proposed design work flow could be used - more specifically in terms of spatial and behavioral qualities - for the briefly mentioned forest-bivouac, if modified to match personal available computational resources.

REFERENCES

This essay was influenced by ideas developed by Stephen Wolfram (cellular automata), Jeff Jones and Andrew Adamatzky (physarum polycephalum simulations) and Nick Briz and Rosa Menkman (glitch art).

- Wolfram, S. (2002) *A New Kind of Science*
- Jones, J. (2010) *Characteristics of pattern formation and evolution in approximations of physarum transport networks*
- Menkman, R. (2011) *The glitch moment(um)*
- Briz, N. (2015) *THOUGHTS ON GLITCH[ART]v2.0*