Playing the Piano: Understanding Algorithmic Music Through Interaction

Prof. Philippe Kocher
Institute for Computer Music and Sound Technology
Zurich University of the Arts
Zurich, Switzerland
e-mail: philippe.kocher@zhdk.ch
Dr. Daniel Bisig
Institute for Computer Music and Sound Technology
Zurich University of the Arts
Zurich, Switzerland
e-mail: daniel.bisig@zhdk.ch
David Inauen
Institute for Computer Music and Sound Technology
Zurich University of the Arts
Zurich, Switzerland
e-mail: david.inauen@zhdk.ch

Abstract

The project *Klavierspiel* (‘piano playing’) combines a piano automaton and interactive, touch-screen-based control software. The piano automaton is a piano-playing robot to be fixed on the keyboard of any conventional instrument. It can perform much faster movements and strike many more keys at the same time than a human pianist. Therefore, it lends itself to a variety of sonic experiments, especially for computer-generated music. The authors developed three different graphical user interfaces to control the automaton. Explicitly aimed at people without musical expertise, these interfaces provide the opportunity to gain hands-on experience with generative music. They illustrate how to create musical gestures, patterns and structures at different levels of abstraction, and convey specific algorithmic composition techniques in an easy-to-understand and practice-oriented way. This paper describes the three user interfaces regarding their technical implementation and their musical potential. Furthermore, it discusses the observations made during a three-day exhibition in Zurich.

1. Introduction

This article describes the installation *Klavierspiel* (‘piano

playing') that we presented at the *Design Biennale Zurich,* an exhibition that displayed national and international projects from various disciplines and took place in late August 2019 [1]. The title of this issue of the biennale was 'PLAY', which indicated that the public was not only meant to look at the artefacts but explicitly invited to explore them in playful interaction.

The realisation of the installation was motivated by our interest in generative music and educational intent to impart knowledge about this artistic practice. The main idea was to design an installation to guide a non-expert audience to obtain a perceptual experience of different generative approaches and an intellectual understanding of the underlying structural principles. We attempted to lead non-musicians to musical thinking by engaging them in playful interactions with generative processes.

The installation consists of three separate applications and thereby exemplifies three different generative approaches. It confronts the user with different structures, different levels of abstraction, and varying degrees of randomness. The applications employed different means to engage the user in a creative interaction: first, a physics engine that generates keystrokes as result of objects falling on a virtual keyboard, second, an interface that translates drawings into music, and third, a flowchart interface to construct simple rule-based compositions.

Apart from the educational aspect, we were also interested in observing how the public interacts with the installation, i. e. in which way, how long, driven by what motivation, and how musically 'meaningful' the visitors would engage with the three applications.

2. The Installation Setup

The musical instrument used for the installation *Klavierspiel* is a piano automaton. This device was conceived and built by the Austrian media artist and engineer Winfried Ritsch, and it serves to turn an acoustic piano into a computer-controlled instrument. It is, therefore, particularly well suited for algorithmic and computer-generated music [2]. The piano automaton is not a piano with a built-in playback technology but a kind of robot piano player, i. e. a self-contained device to be put on top of the keyboard of any grand or upright piano, and fixed in place with two large clamps. It consists of a metal frame that holds 88 solenoids to hit the keys of the keyboard. Three microcontrollers, one master and two slaves, actuate these solenoids. They receive commands over Ethernet from a control software.

One of the specific capabilities of the automaton is that it can depress an arbitrary number of keys simultaneously, all 88 at the same time if necessary, whereas existing, commercially available player pianos (the Yamaha Disklavier for instance) restrict this number to a value high enough for traditional classical piano music but by far too low for the kind of experimental music that Ritsch had in mind [3]. The aesthetic potential of the piano automaton lies in the fact that it can realise music that is beyond the abilities of a human pianist.

Figure 1: The piano automaton.

The installation setup consists of three computers connected to touchscreens with which the user can interact with the applications. The touchscreens are placed on black pedestals that are arranged in a semi-circle around a grand piano (see Fig. 2). Each computer runs one of the three different applications. All computers are connected to a local network over which they send the output of the application (see Fig. 3). This output, in OSC format but structured similar to MIDI data, consists of note events that carry two values: one to indicate the key-number from 21 to 108 (the range of a piano) and another one to specify the velocity as a value normalised to the range between 0 and 1 (where 0 marks a note-off event). A control software that runs on a fourth, headless computer receives all these note events and transforms them into a machine-specific data format to be read by the microcontrollers of the piano automaton. Due to their inter-connection in a network, all three applications can send note events concurrently which allows for three users to play with the piano automaton at the same time.

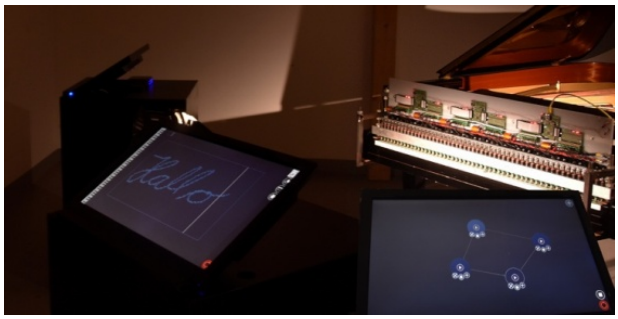

Figure 2: Situation at the exhibition.

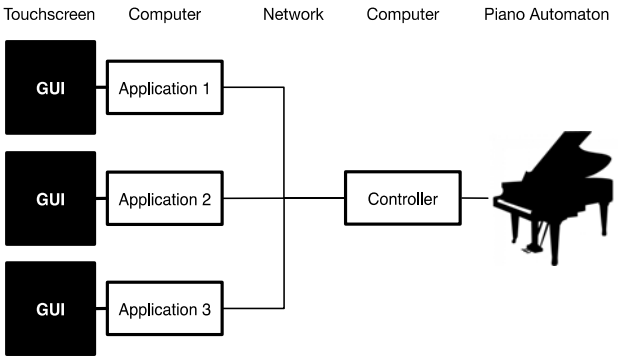

Figure 3: Flowchart of the installation setup.

3. Conceptual Approaches

Each of the authors conceived of an individual interaction scenario and implemented it in an application. This section describes the different conceptual approaches that informed the development of these three applications.

3.1 Cascading Cubes

The first application, named *Cascading Cubes,* provides an interface that emphasises playfulness and intuition of interaction. It does so by establishing a simulation-based environment that combines a realistic representation of a piano keyboard and of a marble run across with small cubes tumble before eventually falling on and triggering a piano key (see Fig. 4). In this simulation, the user can alter the shape of the marble run and modify some of the physical parameters. The situation thus created is intuitive and challenging at the same time. Intuition stems from the fact that the physical principles of a marble run and the correlation between piano key presses and sound events are familiar to anybody. The challenging aspects stem from the large amount of randomness as to where the falling cubes will hit the piano keyboard. Increasing the probability that the cubes are hitting only certain sections of the piano keyboard requires a redesign of the marble run. The more control the users would like to achieve on the musical result, the more carefully they will have to adjust the shape of the marble run. This combination between a readily understandable level of interaction and a difficult to achieve goal follows the principle of playing a game. Accordingly, it is through gamification that this software interface lets the users familiarise themselves with the principle of balancing control and chance in generative music.

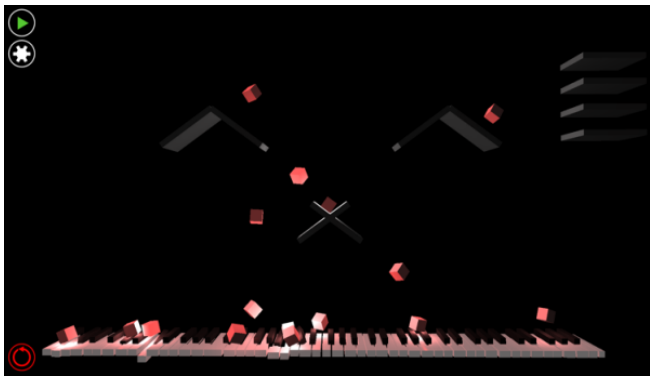

Figure 4: Screenshot of the application Cascading Cubes.

3.2 Little Loops

The application *Little Loops* provides an interface for converting graphical drawings into a musical result. By doing so, it illustrates compositional approaches in which musical material is ideated through visual sketching (see Fig. 5). The act of drawing can serve as a strategy to place the aesthetic focus on the visual domain and to appreciate the musical result as a coincidental or surprising outcome. Conversely, visual sketching can be guided by musical intentionality and thereby provide means for expressing musical thought. For example, scattered points create a pointillistic appearance both in image and music, slanted lines result in ascending or descending scales, multiple parallel lines produce musical gestures in parallel chords, etc.

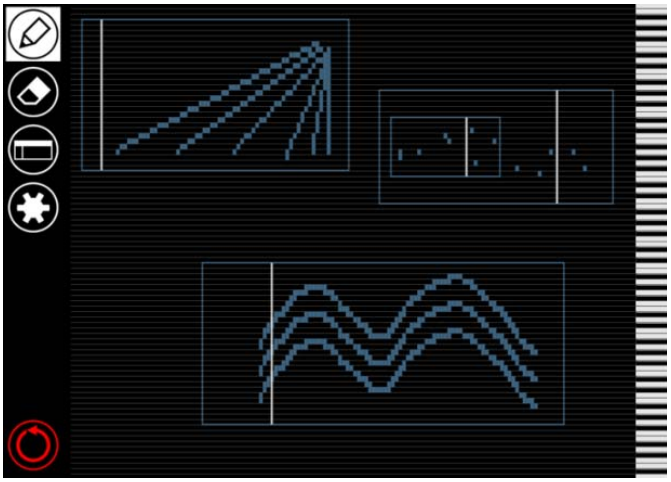

Figure 5: Screenshot of the application Little Loops.

Little Loops follows the convention of the piano roll notation. We chose this convention because it provides a straightforward connection between the two-dimensional layout of a drawing surface and the discretised dimensions of musical pitch and time. The sketching surface is displayed as a grid into which users can directly draw with their fingers. The active grid cells are then turned into actual notes through one or several 'players' that repeatedly read the content of a rectangular region of the image. The user can either choose small regions and thereby translate details of the image into short and frequently looping melodies (hence the name *Little Loops)* or instantiate players that cover the entire image. In any case, the resulting music is repetitive and pattern-based. By experimenting with the placement and size of the players, or by changing the playback speed, the user can explore the musical potential 'contained' within the drawing.

By offering the possibility to change the scale, *Little Loops* also allows the user to encounter more nuanced musical concepts. The following scales are available: the chromatic scale (all keys, default), a pentatonic scale (black keys only), a diatonic scale (white keys only), and whole-tone scale (every other key). By changing the scale, the user will obtain a result that retains the shape of the musical gestures but differs in its harmonic colouration. At the same time, the

user can also observe how a different scale requires a different discretisation among the vertical axis of the drawing surface and thereby alters the appearance of the image.

3.3 Flashing Flowchart

The application *Flashing Flowchart* provides an approach to musical composition on a higher level of abstraction than the other two applications. By means of a directed graph, it describes a musical structure as a rule-based succession of musical events (see Fig. 6). The formal grammar that is constituted by these rules is reflected in the topology of the graph. By manipulating the graph, the user can explore the grammar and the music that originates from it. The abstraction contained in this method reflects, at least in our opinion, the way of thinking that one can find among professional composers. Composing music deals with both the surface of the music and the underlying structure. Composing is not only about finding and selecting sound material but also about making decisions on how to arrange it.

Flashing Flowchart allows for network topologies with closed loops and arbitrary branching. The branching enables various paths through the flowchart. Whenever a node with more than one outgoing connection is reached while traversing the flowchart, the continuation of the path is randomly chosen. The resulting music is characterised by its variability, by containing repetitions that are not verbatim but only similar. This variability lends the music a particular, albeit modest, complexity.

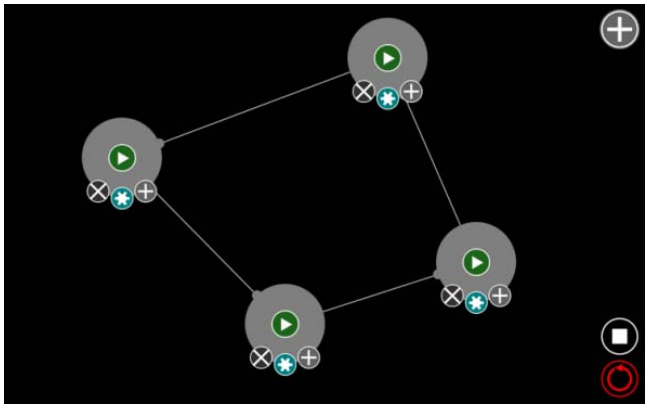

Figure 6: Screenshot of the application Flashing Flowchart.

4. Programming and Implementation

The individuality of every single application is an essential aspect of the whole installation, as is the juxtaposition of the three applications. To realise the individuality, we developed the three applications and their user interfaces separately. We tried, however, to establish certain common design principles concerning the interface. As the installation *Klavierspiel* was meant to address musical laypeople, we decided to keep the user interface as simple as possible by presenting only a restricted set of parameters to the user. To achieve

a high user-friendliness was paramount for this project.

Each application can be reset to a default state by pressing the corresponding button. Also, when a time of five minutes has elapsed without any user input, the application automatically resets itself. A reset puts all parameters to their default values, stops any playback, and presents the user interface in a minimal and tidy state.

4.1 Cascading Cubes

The application *Cascading Cubes* was programmed in C++ in the openFrameworks environment. It employs the Bullet physics engine for realising a physically realistic simulation of a piano keyboard and a marble run. The cubes are simple shapes that accelerate their fall towards the keyboard through gravity. Whenever a cube exceeds a lower vertical position limit, it is removed from the simulation and a new cube is added in its stead in a random position at the top of the screen. The cubes can collide with other cubes, segments of the marble run, or piano keys. Each of these elements possesses its own physical characteristics. Depending on these characteristics, the cubes will slide and bounce off from obstacles very differently. By changing these characteristics, the user can obtain different behaviours, ranging from a realistic simulation to an unrealistic caricature of physics.

The simulation controls the piano automaton in the following manner. Whenever the rotation of a simulated piano key exceeds a lower limit, a note-on event is triggered. The pitch of the note depends on the giving piano key. The note's velocity is proportional to the mass of the cube that was involved in the collision. As the simulated key returns to its rest position, it traverses an upper limit, which in turn causes a note-off event.

The visual rendering of the simulation is overlaid with a GUI that contains the following elements: a button for starting, stopping, and resetting the simulation, numbered buttons for choosing among a set of predefined configurations, and sliders for manually changing the values for the simulation parameters. The following simulation parameters are exposed through the GUI: the number of cubes, the mass of cubes, the restitution of cubes, the lower rotational limit of the piano keys, the velocity of the keys returning to their rest position, and the time step of the simulation. In addition, the user can directly interact with the segments of the marble run. Once selected, a segment can be moved around by single-finger gestures or rotated by two-finger gestures.

4.2 Little Loops

The application *Little Loops* was programmed in Java in the programming environment Processing. Its main interface takes the form of a grid. The grid's vertical extension represents pitch. This extension is subdivided into 88 rows, each of which corresponds to a piano key. The horizontal extension of the grid represents time (running from left to right). This extension is subdivided into 150 columns. Aligned with the grid and displayed on its right side is a schematic representation of a piano keyboard.

By touching the screen, the user can draw into the grid and thereby activate individual grid cells. The user interface provides different drawing tools: a simple pen to activate single cells, a triple pen to activate three cells evenly spaced along the vertical axis, and a line tool to activate all cells along a straight line between a starting and ending point. In addition, there is an eraser tool to deactivate the cells within a four-cell square region.

A player region appears as an outlined rectangle that superimposes a portion of the grid. Each player region contains a playhead (indicated as a vertical line) that continuously moves from left to right and wraps around when it exceeds the right edge of its region. Active cells within a region are translated into notes whenever the playhead passes over them. Each player contains a set of graphical interface elements that allow users to change the size or location of the player's region, to start or stop the playback, to alter the speed of the playback, or to delete the player. The user is free to add any number of additional players whose regions can either be located next to each other or overlap.

When the user switches to another scale, the available vertical positions for the grid cells change according to a pattern associated with the scale. All active cells are shifted vertically to the nearest available position. This shift is reversible; returning to the scale that was used during drawing restores a cell's original location. No matter which drawing tool the user has selected, new cells can only be added at one of the available positions.

4.3 Flashing Flowchart

The application *Flashing Flowchart* was programmed in Java using the programming environment Processing. The nodes of a graph with directional edges represent instances that can play a musical event. They hold a set of parameters that describe the type of musical event: whether it is a single note or one of four predefined chords, and whether

the duration of this event is short, long or very long. Every node can be connected to other nodes by an arbitrary number of outgoing and incoming connections.

When a node fires, the application sends out the appropriate note-on message(s). Then, one of the outgoing connections is randomly chosen, and a cursor is added to this connection. The cursor, visualised as a little spark, travels along the connection until it reaches and triggers the next node. As the cursor travels at a fixed speed, the elapsed time until the next node plays its musical event is proportional to the length of the connection. The layout of the graph and the relative distances between interconnected nodes determine the inter-onset-intervals of the musical events, which in turn define a rhythm. There can be several cursors at the same time, which leads to a canonic structure.

The GUI contains the following elements: two buttons for stopping and resetting the simulation, one button to add a new node to the graph. The nodes themselves are visualised as circles with another four buttons on it: a start button to trigger the node, a button to delete it, a button to open a menu to specify the node's parameters, and a button to sprout a new, connected node. The nodes can be dragged around on the touchscreen. When a new node is created, it appears as 'ghost' to be first moved to its definite place before it is instantiated. In this provisional state, a node can be dragged over another node with which it then merges. This behaviour serves to create loops in the flowchart. The parameters of a newly created node are set to 'single note', 'short duration', and a random pitch.

5. Results

In order to gain an understanding of how users interact with the three different applications, we gathered information with regard to the users' behaviours, interface usage, and achieved results. The users' behaviours were evaluated through observation and lead to anecdotical evidence about different forms of engagement. We made these observations during our on-site supervision of the installation.

Further information about the user interaction was acquired through a mechanism integrated into each application that stored the state of the interface as snapshots at a regular interval of one minute. Based on these snapshots, a small statistical evaluation of the frequency of usage of interface elements was conducted. From this evaluation, insights could be gained concerning the usability of the interface and the users' willingness to delve deeper into some of the more nuanced

possibilities that the applications provide.

A manual comparison between the application states that were reconstructed from the saved snapshots led to a grouping of these states into different categories. The identification of these categories alongside with the frequency of their appearance provided cues about the focus of the users' attention and the exhaustiveness of their attempts to reach interesting results.

The evaluation revealed patterns in the users' engagement that are similar among the three applications. Overall, the installation enjoyed great attention among younger people, in particular among children. The attractivity for children was particularly prominent for *Cascading Cubes* and *Little Loops*. In case of *Flashing Flowchart,* predominantly adult people were engaging with the interface. Concerning the duration of user engagement, three types could be distinguished. Some users would leave the installation without trying to interact with one of the applications. Others spent only a brief amount of time with each of the applications. These users were satisfied once they acquired an initial understanding of the functionality of each interface but did not feel compelled to explore its possibilities further. Finally, several users were sufficiently fascinated by at least one interface that they would dedicate an extended period of time to their attempt to achieve an satisfying musical outcome.

The following three subsections describe the results obtained from the evaluation that are distinct among the three applications.

5.1 Cascading Cubes

Through observation, we found that some users were initially clueless concerning the means of interaction. Several users tried to touch the keys of the simulated piano keyboard directly. This type of failed interaction usually happened when the simulation showed no falling cubes. Eventually, the users found and pressed the play button, after which the observation of the falling cubes and their effect on the sound production was sufficiently self-explanatory to understand the functioning of the interface.

The saved states of the interface were statistically analysed to distinguish between the following types of interaction: Selection of predefined settings (4%), change of simulation parameters (56%), change of position and/or rotation of marble run segments (40%).

The following table shows a categorisation of the different marble run designs and the frequency of their occurrence (for

representative examples of these categories see Fig. 7).

Default 14.3%	Only minimal differences from a predefined setting.
Random/ Few 14.9%	A few segments distributed in a seemingly random manner.
Random/ Many 16.7%	Many segments in a seemingly random manner.
Filter 9.0%	Segments placed closely above the keyboard with small gaps between them. The gaps act as filters allowing the cubes to hit only a few keys.
Slide 16.7%	Rotated and closely spaced segments form a slide along which cubes move.
Barrier 9.0%	Horizontal and closely spaced prevent cubes from reaching the keys.
Steps 2.1%	Horizontal segments evenly spaced along a declining line resembling steps of a stairs.
Basket 5.1%	Horizontal and rotated segments in the shape of a baskets in which the cubes tend to get caught.
Funnel 2.0%	Segments arranged as two mirrored slides with a small gap at the bottom..
Keyboard Abuse 10.2%	Segments intersect with the key-board causing the keys to tremble due to instabilities in the simulation.

Through observation, we found that all users would immediately begin to draw, but some of them were confused by that fact that their drawings were not immediately audible. These users spent some time struggling with the interface until they figured out that they had to place a player above their drawing.\

Based on a statistical evaluation of the snapshots, it became evident that most users did not bother to work with more than one or two players (see Fig. 8). It must even be assumed that this number includes the player that the application provides when reset to its default state. Most of the users ignored this player and left it running idle without any content. The fact that the users mostly utilised only very few players possibly indicates that the relationship between a drawing and the musical result is more readily understandable in case of a single player covering the entire drawing. Splitting up the drawing into several regions to be read by individual players results in a multi-layered musical result which seemed to be too complex for most users.

5.2 Little Loops

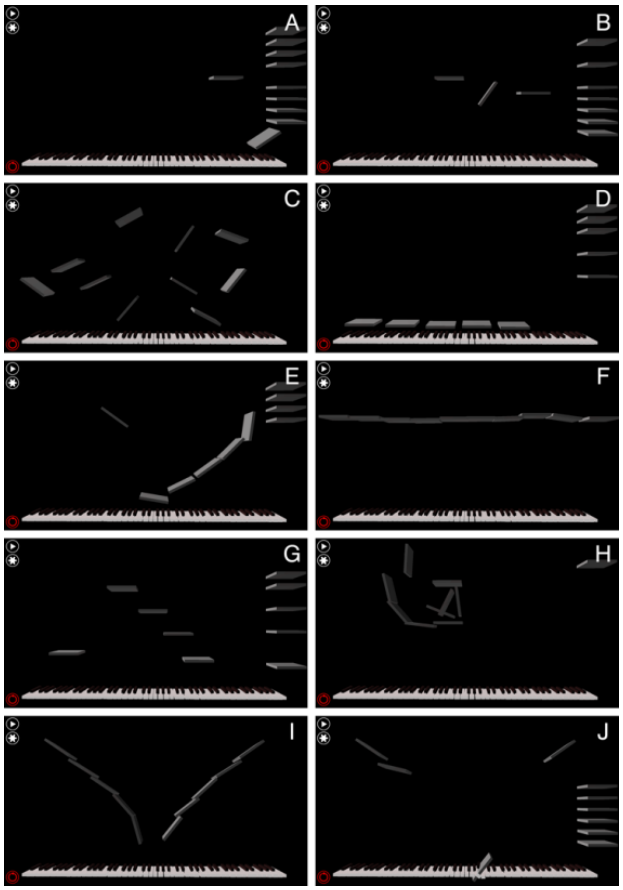

Figure 7: Representative examples of categories of marble run designs. A: Default, B: Random/Few, C: Random/Many, D: Filter, E: Slide, F: Barrier, G: Steps, H: Basket, I: Funnel, J: Keyboard Abuse.

Users rarely explored some of the more advanced possibilities provided by the application. In 91.6% of all snapshots, users used the chromatic scale, which corresponds to the application's default setting. Other scales appeared with the following frequencies: diatonic scale 1.4%, pentatonic scale 3.8%, whole-tone scale 3.2%. Very similarly, in 90.7% of all snapshots, the user used the default 'simple pen' as a drawing tool.

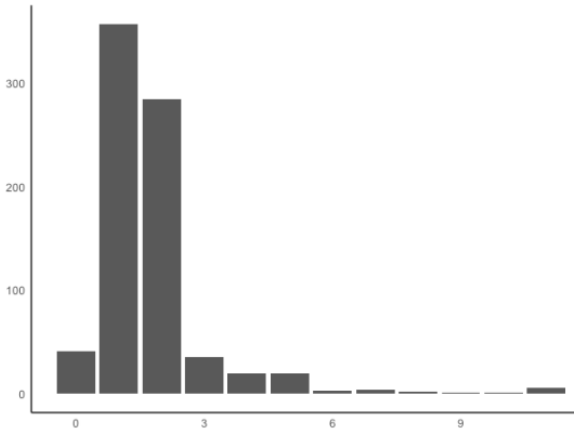

Figure 8: Distribution of the number of simultaneously opened players on the piano roll.

An inspection of the reconstructed application states led to the identification of the following four different types of drawings (for representtative examples of these categories see Fig. 9).

Figurative 10%	The drawing is figurative and resembles an iconic object (e. g. a house, a face, a heart shape).
Writing 10%	The drawing resembles a written text that shows either single letters, phrases, or names.
Lines 47%	The drawing consists of multiple distinct and simple shapes that usually take on the form of continuous or dotted lines that are either straight or curvy. These drawings resemble a score and therefore represent musical expressions.
Scribble 33%	The drawing is neither identifiable nor is the user's intention in creating the drawing apparent.

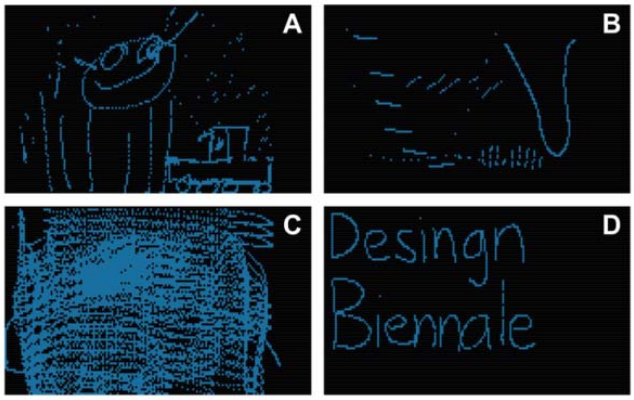

Figure 9: Representative examples of the four drawing categories: A: Figurative, B: Line, C: Scribble, D: Writing.

Among all the user drawings, those that correspond to the category ‘Lines’ were the most prominent. This indicates that users readily understood the drawing functionality as a method to test out musical ideas. It can be assumed that the appearance of the interface as a piano roll grid and the presence of a real piano helped to nudge the users’ intention into a musical direction. For those users that created drawings of the categories ‘Figurative’ and ‘Writing’, the application appeared first and foremost as a graphical tool that allowed to explore the musical rendition of the drawing as an additional feature. These users were, for instance, interested in hearing how their name sounds on a piano.

5.3 Flashing Flowchart

We observed that the users engaged either only for a short moment with this interaction or for a long time. The fact, that some of the users spent only little time in front of the screen can indicate that the interaction is too complicated, the musical result does not sound catching right from the beginning, and the application provides no game-like challenge. Concerning interaction, most of the users failed to figure out how to close a loop in the flowchart (or did not even try to find out). Without loops, the musical playback ends after only a few notes, which is not particularly attractive. Those users, in turn, that spent a lot of time with this application were driven by a musical interest and tried to create their own, small compositions. Primarily, this interaction seemed to attract people with prior knowledge in music-making.

We had to acknowledge that the application *Flashing Flowchart* is the most demanding. The users had to spend some time to explore the interaction to find out how to build a musically interesting flowchart. Moreover, it is an application that addresses mostly people who are able or willing to engage with musical abstraction. In the light of this project's aims, to present different approaches to algorithmic composition, this limitation seemed acceptable because it is presented in

contrasting juxtaposition with the other interactions.

Most users did not further explore the parameter settings as can be seen in the fact that for the majority of the nodes recorded in the snapshots the default settings were left unchanged (70.1% ‘single note’ and 78.5% ‘short duration’). The number of nodes that appeared simultaneously in a flowchart ranges up to 19 (a maximum given by the size of the screen). The distribution of nodes can be seen in Fig. 10.

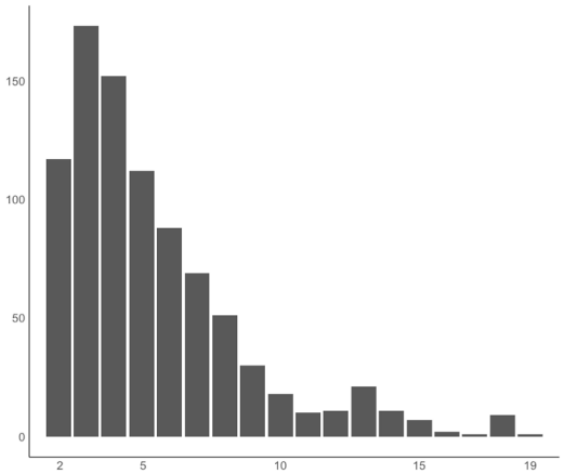

Figure 10: Distribution of the number of nodes per the flowchart.

We categorised the snapshots with respect to the topology of the flowcharts as listed in the following table (for representative examples of these categories see Fig. 11). The numbers affirm the observation that many users had difficulties in finding out how to create a closed loop on the screen.

Continuous Iteration 15.4%	Non-terminating flowchart, one single path.
Continuous Variation 23.1%	Non-terminating flowchart, several possible paths.
Terminating Variation 13.8%	Flowchart with loops and one or more branching-offs leading to a dead end.
One-Shot 47.7%	Flowchart without any loops.

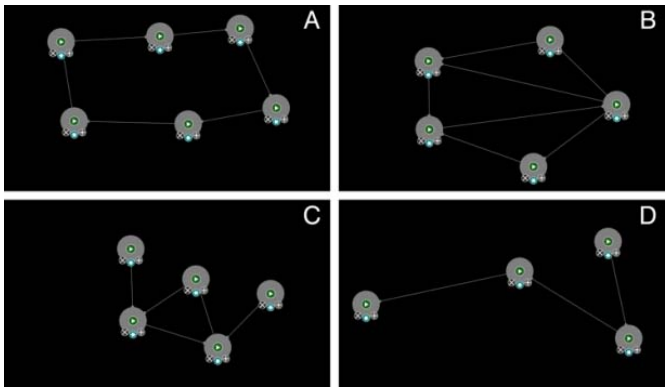

Figure 11: Representative examples of the four flowchart categories: A: Continuous Iteration, B: Continuous Variation, C: Terminating Variation, D: One-Shot.

6. Conclusion

The presentation of the installation *Klavierspiel* at the *Design Biennale Zurich* provided an excellent setting to communicate to a non-expert audience some of what we consider to be core principles of employing generative systems in musical composition. To gain as

many insights as possible from the visitors' response to this setting, an evaluation was conducted that combined observation of user behaviour, statistical analysis of interface usage, and category formation of achieved results. We believe that such a combination provides insights that can be of value for any artist working in the field of generative and interactive art and music. Among others, such a combined evaluation allows a discrimination between a visitor's level of understanding concerning the possibilities of interaction versus the visitor's level of comprehension of the generative principles. Furthermore, this combined evaluation also permits to assess the diversity of results that a generative system can generate in response to a user's first-time interaction and these results can convey information about a user's motivation and intention to interact in the first place.

Our motivation for realising this generative installation was predominantly a didactical one. For this reason, not all of the design decisions that were made are readily transferrable to other, more artistic approaches. If nothing more, this article highlights that for any interactive and generative artwork that is meant to be used and experienced by non-expert users, an informed decision has to be made by the artist concerning the balancing of understandability and complexity. The three applications exemplify different attempts of finding such a balance. They do so by choosing distinct levels of abstraction, order, and randomisation while being similar in their sacrificing of autonomy in favour of a high level of control, and their limiting of the diversity of possible outcomes in favour of more readily accessible results. We believe that by documenting the rationale for these decisions, they can also help to inform artistic strategies that deal with the development of sophisticated generative systems which are meant to be interacted with by a professional audience.

7. References

[1] https://www.designbiennalezurich.ch (accessed: November 2019).

[2] Philippe Kocher: "The Piano Automaton as an Instrument for Algorithmic Music", Proc. of the Generative Art Conference, Verona, 2018.

[3] Winfried Ritsch: "Robotic Piano Player Making Pianos Talk", Proc. of the Sound and Music Computing Conference, Padova, 2011.