Journal of the Japanese Society for Sonic Arts, Vol.1 No.1 pp.17–21

Research Report

ZIRKONIUM MK III - A TOOLKIT FOR SPATIAL COMPOSITION

Chikashi Miyama *Götz Dipper* *Ludger Brümmer*
ZKM | Institute for Music and Acoustics
Lorenzstr. 19 D-76135 Karlsruhe

ABSTRACT

Zirkonium is a set of Mac OSX software tools to aid the composition of spatial music; the software allows composers to design multiple spatial trajectories with an intuitive GUI and facilitates arranging them in time. According to the provided trajectory information, the actual audio signals can then be rendered in real-time for 2D or 3D loudspeaker systems. For developing the latest version of Zirkonium, we focused on improving the aspects of usability, visualization, and compatibility. Consequently, the software structure and the GUI were drastically redesigned, and a number of functionalities, such as parametric trajectory creation, automatic interpolation, event-filtering, are newly implemented. Furthermore, the integrated SpatDIF library enables the reuse of spatial trajectories, created by Zirkonium, in other software.

1. BACKGROUND

1.1. ZKM | Institute for Music and Acoustics

Figure 1. Klangdom

The mission of IMA (Institute for Music and Acoustics) at ZKM Karlsruhe is producing and presenting concerts of electro-acoustic music. The *Kubus*, the main concert hall of the institute, is equipped with a 3D surround loudspeaker system, called *Klangdom*[1]. The system comprises 43 loudspeakers, arranged in the form of a hemisphere (Fig. 1). The development of the Klangdom was initiated in 2004. A crucial part of the project is the *Zirkonium*, a software for realizing spatial compositions primarily for the Klangdom[2].

1.2. The first version of Zirkonium

Zirkonium was first released in 2005 as a stand-alone Mac OSX application. It provides users with control over max. 32 independent mono sound sources. The user is able to place these sound sources onto the virtual surface of the Klangdom and move them along the surface. The position and movement may be controlled by a score, created and edited within Zirkonium, or from another software via OSC[3]. The Zirkonium score comprises an arbitrary number of spatial events and each event instructs the movement of a single or a group of sources. For editing scores, Zirkonium offers a simple text-based interface in which the user enters numerical values for moving sound sources. This first version of Zirkonium had been refining continuously until 2012, when it was decided to completely rewrite it.

1.3. Zirkonium MK II

The newly reprogrammed version was named *Zirkonium MK II*[4] and it was designed as a collection of three separate relatively small software modules, named *Trajectory editor*, *Speaker setup*, and *Spatialization server* (Fig. 2). These three applications are responsible for three different tasks.

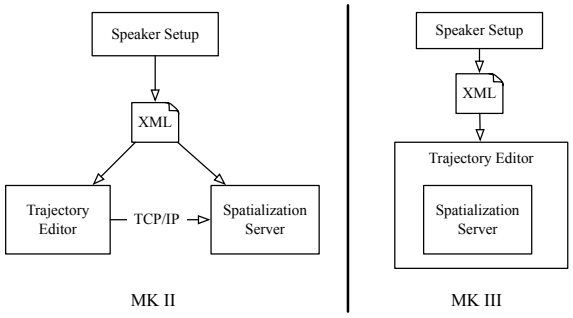

Figure 2. Difference between MK II and MK III structure

The *Trajectory editor* allows users to design trajectories with a GUI and to arrange them in time. The *Speaker setup* facilitates writing XML files, that define the position of loudspeakers in a 2D or 3D space. The *Spatialization server* loads the XML file, generated by the Speaker setup, receives TCP/IP packets from the Trajectory editor, and spatializes the actual audio signals. This modular approach allows users to combine individual software modules with their familiar tools, such as DAWs or physical interfaces, and enables them to integrate Zirkonium into

their own workflow. Moreover, the text-based score editor of the original version was replaced by a GUI in which the user can intuitively draw the movement of each sound source through simple mouse operations.

1.4. Motivation for the new version

With this distinctive modular approach and newly implemented GUI, Zirkonium MK II attained superior flexibility and extensibility. However, there would be still room for improvement, particularly in terms of usability and data accessibility. With MK II, it is necessary for users to launch at least two applications and setup various parameters before they hear actual sound from the software.

In addition, the separation between the Spatialization server and the Trajectory editor might decrease the accessibility of data held by each application. For instance, the Trajectory editor is not able to access the actual audio data, that the Spatialization server plays back. This separation may also restrict the development of functionalities based on audio data, such as audio visualization or audio-based spatial event or trajectory creation.

2. ZIRKONIUM MK III

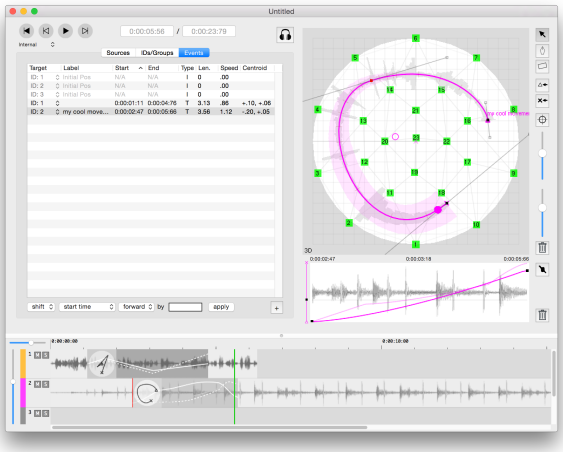

Figure 3. Zirkonium MK III Trajectory Editor

In order to refine the two aforementioned points, the structure of MK II was reassessed. The decision we made for the latest version, MK III, is to integrate the module of the Spatialization server to the Trajectory editor (Fig. 2). By uniting these two software modules, we attempt to decrease the operational complexity of the entire system and increase the inter-module data accessibility. In other words, MK III amalgamates the flexibility of the modular approach of MK II and the intelligibility of the original version.

2.1. Trajectory editor

The new Trajectory editor provides a superior GUI for designing audio trajectories and arranging spatial events in time (Fig. 3). In MK III, the Trajectory editor is also responsible for processing audio files or live audio signals from physical inputs, execute spatial rendering, and distribute audio signals to a maximum of 64 loudspeakers.

Figure 4 depicts the software architecture of the Trajectory editor. The software consists of three components: GUI, Data management, and Spatial rendering engine.

For MK III, most of the GUI components are reimplemented with OpenGL and GLSL[5] in order to conserve the CPU resources for the execution of the spatial rendering algorithms.

The Data management component processes all the data regarding spatial compositions. It has functionalities of importing Speaker-Setup XML files, and exporting spatial events to SpatDIF-XML files (described later in detail).

In the Spatial rendering engine, the Spatialization server executes the spatial rendering algorithms and distributes the actual audio signals to each output channel. The spatialization server is entirely programmed with Pd (Pure Data)[6] and integrated into the Trajectory Editor with the aid of *libPd*, a C Library that turns Pd into an embeddable library[7]. With this integration, the Trajectory editor in MK III is capable of accessing audio content more efficiently than in MK II. The new Trajectory editor offers various new features that take full advantage of this integration.

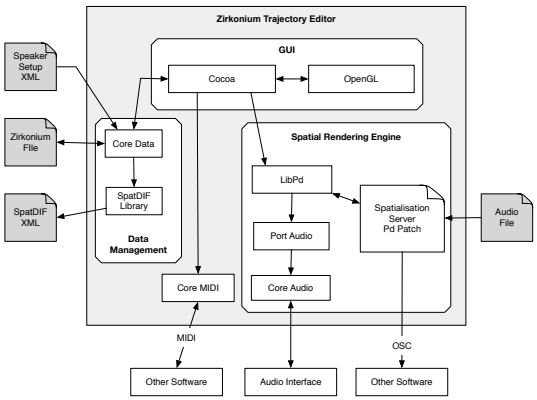

Figure 4. Software architecture of the Trajectory Editor

The following subsections introduce the most important additional functionalities, implemented in the Trajectory editor.

2.1.1. *Trajectory Creation in MK III*

In MK III, the graphical approach of trajectory creation, inherited from MK II, is further enhanced.

In the Trajectory editor, a single trajectory (i.e. a movement of a virtual sound source in a specific time frame) is determined by a pair of paths; a *Sound path* and a *Motion path*. These two paths are drawn in two different views: the *Dome view* and the *Motion view*. The Dome view displays a space for spatialization, observed orthographically from the zenith. In this view, a Sound path, a geometrical route, that a virtual sound source moves along, can be drawn with Bèzier curves. The Motion view and the Motion paths, on the other hand, visualize how a sound source moves along a Sound path in a specific period of time. In MK III, the Motion path can be drawn with a multi-segment curve. The steepness of each segment is

independently configurable by simple mouse operations and it controls the acceleration and deceleration of spatial movements.

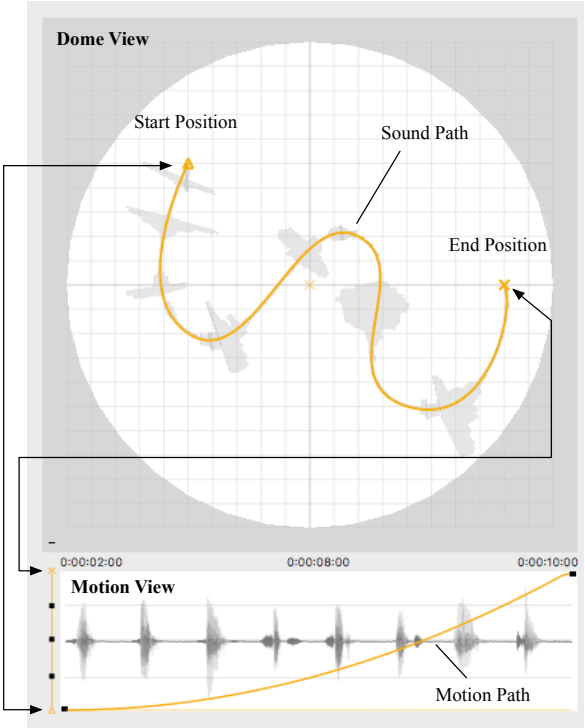

Figure 5. Sound path and Motion path

Figure 5 shows a possible combination of a Sound path and a Motion path in the Dome view and the Motion view. In the Dome view, a meandering Sound path is defined. A virtual sound source moves along this Sound path from the start point to the end point, marked by a triangle and a cross symbol respectively. The X-axis of the Motion view indicates the time line of an event, and the Y-axis represents the relative position between the start and the end point of the Sound path. The start point (triangle symbol) coincides with the bottom and the end point (cross symbol) coincides with the top of the Motion view. In this way, the Motion view displays the relationship between time and relative position of a sound source, moving along the Sound path. In figure 5, an exponential curve is used as a Motion path. Thus, the sound source accelerates its speed towards the end point (cross).

Moreover, as shown in the figure, the waveform of audio file is rendered along the Sound path and behind the Motion path in MK III. This feature enables users to grasp the relationship between the audio content and its position in space, and to adjust a certain audio content to a specific position in the Dome view.

2.1.2. *Parameter-based Trajectory Creation*

In addition to the manual drawing method with Bèzier curves, the software provides an algorithmic approach for Sound path creation. By entering a few parameters to the "add circle/spiral" pop-over panel, the software automatically draws a circular or spiral Sound path in the dome view, employing the minimum amount of Bèzier curves required (Fig. 6). These algorithmically drawn Sound paths can be further modified by mouse operations.

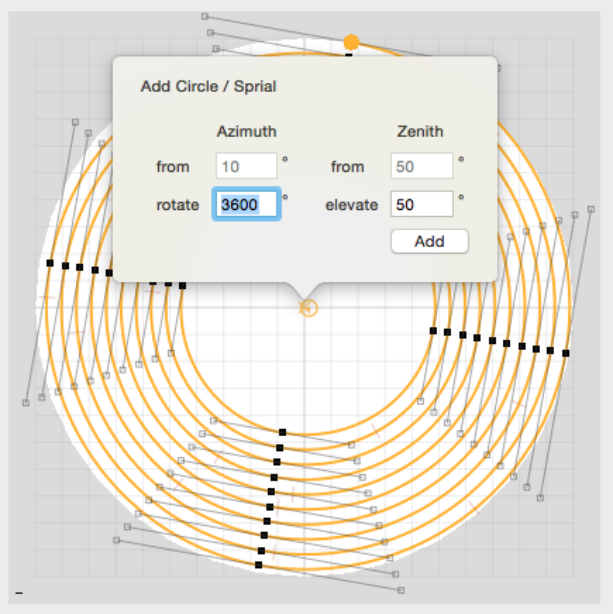

Figure 6. Algorithmically generated Sound path

2.1.3. *Spatial Rendering Algorithms*

The Trajectory editor currently provides two different rendering algorithms, VBAP (Vector Based Amplitude Panning) and HOA (Higher Order Ambisonics). These two different rendering algorithms are implemented as external Pd objects, named *ZirkVBA* and *ZirkHOA*. These objects execute the actual spatial rendering in the integrated Spatialization server at runtime. The ZirkVBAP object is implemented based on the *vbap* object by Ville Pulkki[8], while the ZirkHOA is implemented based on the HOA Library by CICM[9]. For the HOA, the order of Ambisonics is automatically determined by the dimensionality of the space and the number of loudspeakers. In addition, the HOA algorithm provides further optimization options for spatial rendering: *Basic*, *Max-Re*, and *In Phase*.

The two algorithms can be assigned to each individual virtual sound source; it is possible to utilize both VBAP and HOA simultaneously in a single musical work. In this way the user can deploy the sound characteristics of each algorithm as artistic means of composing.

2.1.4. *Functions for Event handling*

The more a spatial composition project evolves, the more spatial events are used. A powerful event handling tool would be indispensable for increasing productivity. In MK III, multiple new functionalities are implemented to enhance the efficiency of event handling.

The Event view is a newly introduced GUI component in MK III. As shown in figure 7, this view visualizes the waveform of imported sound files and the spatial events assigned to each virtual sound source in the manner of a typical DAW software. In the figure, the waveform of two sound files is displayed and spatial events are represented

as transparent dark rectangles, superimposed on the waveforms. On top of these rectangles, snapshots of Sound paths, edited in the Dome view, are shown as thumbnails, and the chronological position of the sound source on X and Y axes are represented as a solid and a dotted line respectively. This graphical representation enables users to quickly grasp the distribution of trajectories in time.

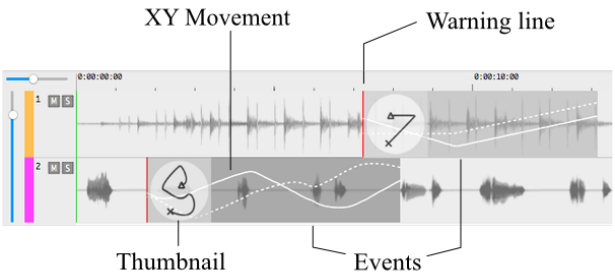

Figure 7. Event view

Besides, the Trajectory editor also offers the Event filtering panel (Fig. 8). This panel provides a way to automatically select specific events that match particular conditions. All selected events can then be shifted, scaled, copied or erased at once with mouse or keyboard operations.

Figure 8. Event filtering panel

2.1.5. *Automatic Interpolation*

A Sound path, assigned to each spatial event, has a geometrical start point and end point. If the start point of a Sound path does not match the end point of the previous event, the virtual sound source abruptly leaps from the end point of the previous event to the start point of the current event during the playback. These unnatural spatial leaps should be avoided in most of the cases. Zirkonium offers three practical solutions for this issue.

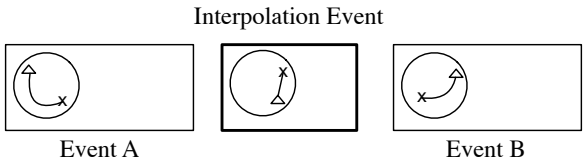

Figure 9. Interpolation event

Firstly, in order to let users instantly know where in the time line these leaps occur, vertical red warning lines are shown at the left edge of the rectangles (Fig. 7). These red lines indicate the moment of leaps; they disappear, when the start point of the event and the end point of the previous event match.

Secondly, two buttons, located near to the right edge of the window, allow users either to match the start position of a selected event and the end position of the previous event, or to match the end position of a selected event and the start position of the next event.

Thirdly, when an event is created between two events that have unmatched end and start points, the event is automatically filled with a Sound path, interpolating these two points (Fig. 9).

2.1.6. *Re-programmable Spatialization server*

The integration of the Trajectory editor and the Spatialization server does not prevent users from accessing the Spatialization server, running internally in the Trajectory editor. As described above, the server is programmed as a Pd patch (Fig. 10); advanced users with experiences in Pd programming are able to access the patch, modify the core spatial rendering algorithms, and apply arbitrary custom effects (e.g. Reverb or Doppler) to sound sources.

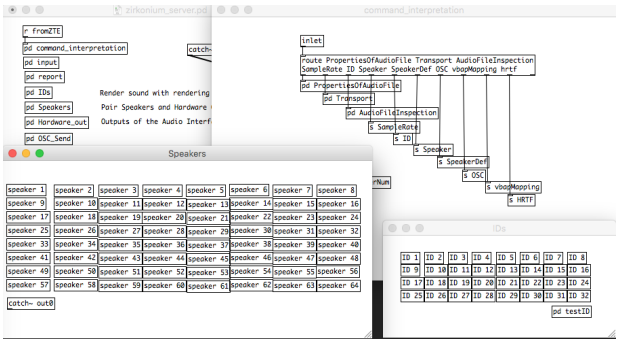

Figure 10. A snapshot of the Spatialization server Pd-patch, running in the Trajectory editor

2.1.7. *SpatDIF Export*

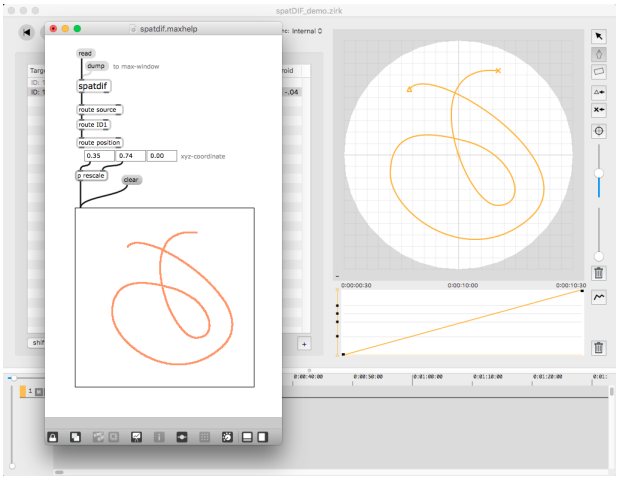

Figure 11. A SpatDIF external object reads a SpatDIF-XML file and renders a trajectory, designed in the Trajectory editor, onto a LCD object in a Max patch

SpatDIF (Spatial Descriptor Interchange Format) is a universal syntax for describing spatial movement [10]. By the aid of the SpatDIF Library, the Trajectory editor is able to export all the trajectories, designed in the Trajectory editor, to a XML file, conformed to the SpatDIF v0.3 specification.

The data of the trajectories, exported from the Trajectory editor can be reused in other SpatDIF-compatible software. Notably, by using SpatDIF external objects provided by ICST Zürich, trajectories can be loaded and employed in common real-time audio programming environments, such as Max or Pure Data (Fig. 11).

This feature would significantly facilitate composers and sound artists to reuse trajectories, designed with the Trajectory editor, in live and interactive music contexts.

2.2. Speaker setup

The Speaker setup application is employed for creating a XML file that defines the position of the loudspeakers in a 2D or 3D space. This XML file is required by the Trajectory editor for the spatial audio rendering and the visualization of the speakers in the Dome view. For MK III, the Speaker setup application is entirely reimplemented (Fig. 12). It visualizes a virtual listening space with a listener, and allows users to position loudspeakers intuitively.

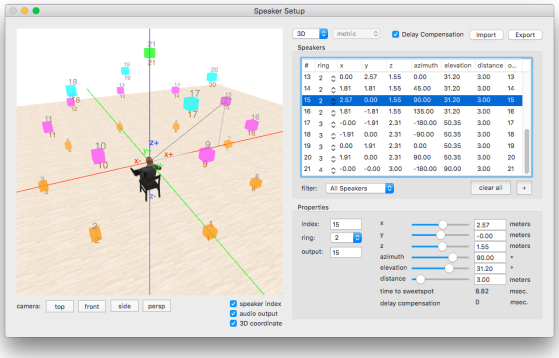

Figure 12. Improved Speaker setup

2.2.1. *Delay Compensation*

Although it is often technically impracticable, the position of all loudspeakers should be equally distant from the sweetspot in order to attain the best acoustic result, since uneven distance between the listener and speakers causes unwanted acoustic phenomena, such as comb-filtering.

A common solution for this problem is to apply a slight delay for compensation to loudspeakers located near to the listener compared to other speakers. The Speaker setup application in MK III automatically calculates the proper delay time for each loudspeaker, based on the metrical position of the speakers, provided by the user, and stores the delay time in the exported XML file. The Trajectory editor then applies the designated delay time to each output channel automatically, when the XML file is loaded.

3. CONCLUSION AND FUTURE PLANS

For the new release of Zirkonium, we focused on improving usability, visualization, and compatibility. We attempted to attain these objectives by modifying the modular structure, introducing the new graphical components, and integrating the SpatDIF Library. For the next development phase, enhancements in the real-time capability, archive functionality, and backward compatibility are planned. In addition, more algorithmic approaches and audio-based approaches for trajectory creation will be explored.

4. AVAILABILITY

Zirkonium MK III public beta is available on the website of ZKM | IMA. http://zkm.de/zirkonium/.

5. ACKNOWLEDGMENTS

This software development would not have been possible without the generous support of researchers and composers, especially, Jan Schacher, Trond Lossius, Junya Oikawa, and the colleagues at ZKM.

6. REFERENCES

[1] C. Ramakrishnan, J. Goßmann, and L. Brümmer, “The ZKM Klangdom”. in *Proceedings of NIME* Paris, France 2006 pp. 140-143.

[2] C. Ramakrishnan, “Zirkonium: Noninvasive software for sound spatialisation”. *Organised Sound*, Vol.14 No.3 2009 pp 268-276.

[3] A. Schmeder, *Everything you ever wanted to know about Open Sound Control.* Tech. rep. Mar. 2008.

[4] D. Wagner, L. Brümmer, G. Dipper, and J. A. Otto, “Introducing the Zirkonium MK2 System for Spatial Composition”. in *Proceedings of ICMC-SMC, Athens, Greece* 2014 pp.823-828.

[5] “OpenGL” *https://www.opengl.org*.

[6] “Pure Data” *http://puredata.info*.

[7] P. Brinkmann, *Making Musical Apps.* O’Reilly Media, 2012.

[8] V. Pulkki, “Virtual sound source positioning using vector base amplitude panning”. *Journal of the Audio Engineering Society*, Vol.45 No.6 1997 pp.456-466.

[9] S. Anne, G. Pierre, and P. Eliott, “The HOA library, review and prospects”. in *Proceedings of ICMC-SMC, Athens, Greece*, 2014 pp.856-860.

[10] J. C. Schacher, C. Miyama, and T. Lossius, “The SpatDIF library Concepts and Practical Applications in Audio Software” in *Proceedings of ICMC-SMC Athens, Greece*, 2014 pp.861-868.

7. AUTHOR'S PROFILES

Chikashi Miyama

Chikashi Miyama is a composer, programmer, video artist, interface designer, and performer. He received a MA (Sonology/2004) from Kunitachi College of Music, Tokyo, Japan, a Nachdiplom (Komposition im Elektronischen Studio/2007) from Music academy of Basel, Switzerland, and a Ph.D (Composition/2011) from University at Buffalo, New york, USA. His compositions have received a first prize from Kunstfestival Strom (2015/Germany), an ICMA award (2011/UK) from the International Computer Music Association, a second prize in SEAMUS commission competition (2010/St. Cloud, USA), a special prize in Destellos Competition (2009/Argentina), and a honorable mention in the Bourges Electroacoustic Music Com petition (2002/France). Several works of him are included on the DVD of the Computer Music Journal Vol.28 by MIT press, and ICMC official CD/DVD(2005/2011). In 2011, he received a research grant from DAAD (German Academic Exchange Service) and worked as a visiting researcher at ZKM, Karlsruhe, Germany. He has taught computer music at University at Buffalo, USA and College of Arts Bern, Switzerland. He is currently teaching at College of Music and Dance Cologne, and working as a software developer at ZKM Karlsruhe and ICST Zrich. (http://chikashi.net)

Götz Dipper

Götz Dipper, born 1966 in Stuttgart, studied cello at the Hanover Academy of Music and at the Mozarteum, in Salzburg. Afterwards he turned his interests to computer music, with a focus on sound installations. Since 2001 he has been working as a member of the artistic/scientific staff at the ZKM | Institute for Music and Acoustics. Since 2015 he has been also teaching at the Karlsruhe University of Art and Design (HfG).

Ludger Bruemmer

Born and educated in Germany. Studied psychology/sociology in Dortmund and composition with Nicolaus A. Huber and Dirk Reith at the Institute for Computermusic und Electronic Media [ICEM] Essen. Visiting Scholar at Centre for Computer Research in Music and Acoustics, Stanford University (1991-1993). Lecturer at the ICEM, Folkwang Hochschule Essen (1993-2000). Research Fellow at Kingston University (2000-2002). Composer Residence and Guest Composer at the Centre for Art and Media, Karlsruhe (ZKM 1994-2002), Lecturer at Sonic Arts Research Centre, Queens University Belfast (2001-2002). Currently director of the Institute for Music and Acoustics at the Centre for Arts and Media Karlsruhe (www.zkm.de).

ZKM | Institute for Music and Acoustics

The goal of the ZKM | Institute for Music and Acoustics (IMA) is to provide new impulses to electronic music through its activities in research and development. For this reason, guest artists are regularly invited to work at the institute and develop new works on site. The ZKM | IMA hereby acts as work, production, and performance space. The results of the artistic work are regularly made accessible to an interested public in performances, publications, and editions.