IVES - Interactive Virtual Environment System: A modular toolkit for 3D audiovisual composition in Max

Damian Dziwis (damian.dziwis@th-koeln.de) [1,2], **Johannes M. Arend**[1,2], **Tim Lübeck**[1,2], and **Christoph Pörschmann**[1]

[1]**TH Köln - University of Applied Sciences**, Cologne, Germany
[2]**Technical University of Berlin**, Berlin, Germany

ABSTRACT

The *Interactive Virtual Environment System* (IVES) is a toolkit aiding the production of immersive audiovisual 3D virtual environments for screen-based or virtual reality (VR) applications with loudspeaker- or headphone-based spatial audio reproduction. It is developed within Cycling´74s Max programming environment and consists of a set of interface-based, higher-level building-block modules, similar to the BEAP and VIZZIE toolkits included in Max. IVES uses and unifies established programming libraries such as Jitter / OpenGL (Cycling'74), Spat (IRCAM), VR (Graham Wakefield) into ready-to-use abstractions with graphical user interfaces (GUIs). This allows simple patching of individual spatial audio and visual 3D rendering chains. IVES provides various blocks in a flexible modular patching system, suitable for audiovisual rendering in different application scenarios with different content, and manages the synchronization and conversion of data, control messages and coordinate-systems, used differently in the underlying libraries. Furthermore, the system also provides modules for the creation, interaction, motion, and transformation of audiovisual spatial elements within virtual environments. The toolkit allows users to concentrate on the composition and artistic content in audiovisual virtual environments rather than the programming of the complex systems behind them.

1. INTRODUCTION

In the fourth movement of his composition "Symphony No. 4" [1], Charles Ives uses a percussion ensemble spatially separated from the orchestra as a form of spatial composition technique. This type of, in this context, unconventional placing of sounds in space is only one of countless examples, representative of the interest of composers in the use of spatial features in their works. Placing sounds in space, like Ives did in the concert hall, moving them around and using the room acoustics and their perception, are common techniques used in spatial composition. In the context of electroacoustic/acousmatic music and spatial audio reproduction systems with multi-channel loudspeakers or headphone-based binaural rendering, these techniques have been further developed and also extended. Besides more complex options like the movement with three-dimensional trajectories, also new techniques such as spatial sound processing and the simulation of different room acoustics became possible [1]. This led to the emergence of various software aiding the spatial composition process (e.g., HoloEdit [2, 3], Zirkonium [4, 5]) and spatial audio reproduction (e.g., Spat [6, 7], ICST Ambisonics Tools [8, 9], AudioScape [10]) in common computer music programming environments (e.g., Max [11], PureData [12], SuperCollider [13]).

[1] Symphony No. 4, S. 4 (K. 1A4), 1925; Charles Ives (1874–1954)

1.1 Audiovisual Virtual Environments

Spatial audio plays an important role not only in music but also in other media such as movies and games. In interdisciplinary art, the interest of media artists and composers in combining these media to create spatial audiovisual works also increased. Especially the recent developments and availability of virtual and augmented reality (XR) systems led to new possibilities to explore audiovisual spatial environments for art production.
Emerging from the gaming industry, game engines (such as Unity or Unreal) represent powerful tools for creating and rendering these spatial audiovisual environments for screen- and XR-based applications, even beyond their primary application for games. While also being frequently used for audiovisual spatial composition, game engines can still have shortcomings in several application scenarios. Being very powerful especially in the visual domain, the possibilities in audio might be too limited for many composition requirements. Real-time multi-channel input, sound synthesis, and generative composition procedures can be challenging in game engines. Also, deeper control over the audio processing and the synthesis of spatial sound fields may be desirable for many applications. This has led to several developments that combine game engines and their extensive capabilities in 3D visual environments with computer music languages for composition, sound synthesis, and audio spatialization [14–16].

1.2 Audiovisual Rendering in the Max Environment

While game engines with their graphical editors and integrated physics and rendering engines are attractive tools for creating visual 3D worlds, their basic functionality can

also be programmed in Max. Max is a graphical programming environment, originally focused on audio and music programming and often used by computer music composers and media artists. The Max library Jitter for visual components has in addition to objects for graphics and video also an implementation of OpenGL for programming and rendering 3D environments.
The integration of virtual reality (VR) systems is realized with the VR library for Max [17]. The library allows implementing head-mounted-display (HMD) reproduction, tracking devices, and controller integration.
These libraries make the programming of virtual environments and their rendering within Max a potential alternative to the use of game engines. In combination with a spatial audio and composing library, such as the well-established Spat library by IRCAM, Max users can program their custom virtual environment systems and rendering pipelines using these libraries. This gives them deep control over the underlying processes in a programming environment dedicated to audiovisual composition.
Still, even with the help of these libraries, programming such systems can be very challenging. The implementation requires broad knowledge about spatial audio reproduction, 3D virtual environments, as well as rendering pipelines and their parametrization. Furthermore, exchanged data must be converted and adjusted, for example, because of different coordinate conventions used by the libraries.
Toolkits such as COSM [18] aid the programming of virtual worlds and rendering pipelines by offering high-level programming externals. It provides objects for spatial visual and audio rendering, as well as the creation of virtual worlds. Nevertheless, also with high-level externals, the process of creating audiovisual virtual environments is still quite programming intense.

1.3 The IVES Toolkit

In an attempt to minimize the programming effort required to realize audiovisual virtual environments within Max, we present in this paper the development of "IVES - Interactive Virtual Environment System", a modular toolkit for 3D audiovisual composition in Max. IVES is a set of higher-level building-block modules with graphical user interfaces (GUIs) aiding the patching of virtual environments and rendering pipelines. Within the toolkit, we use the Spat library to program the spatial audio rendering and composition modules, Jitter with OpenGL for visual 3D environments modules, and the Max VR package to integrate HMD-based VR systems. The conversion of data and control messages used in between these libraries is done inside the modules, ensuring consistent use of parameters and coordinate conventions.
As a result, the toolkit provides ready-to-use abstractions with interfaces for parametrization with no additional programming required, while it fully integrates in the Max development environment offering every kind of programmable extension. The modular system preserves a deep control over the signal and rendering processing and allows the adaptation to different source data and application scenarios. This implies the creation of a 3D world and simulated sound field with virtual sound sources and corresponding visual elements, or the use of recorded or real-time microphone array input in combination with audio-reactive 3D visuals. Composers and media artists can create virtual environments presented on VR systems with six-degrees-of-freedom (6DoF) tracking and dynamic binauralization over headphones, as well as works for a concert space with projection and loudspeaker-based sound reproduction. They can integrate various controllers, interfaces, and tracking-systems available in Max and program their own generative systems, algorithmic compositions, or sound synthesis used as content in their audiovisual environments.
The IVES toolkit aims to guide the creation of 3D virtual environments with a focus on audio and music. Unlike game engines, which are very powerful in the visual domain, IVES is used to embed a 3D virtual environment engine in an audio and music specific programming language such as Max, which is quite common among composers and media artists. It provides an easy-to-use solution for artists, especially those already familiar with Max or similar visual languages, and gives them deeper control and more options for real-time audio, spatial composition, sound and their spatialization compared to game engines, with less complexity than combined approaches. Thus, the toolkit allows artists to use the basic features of visual 3D environments, VR, and sound spatialization without having to learn a new environment or language, or programming of the principles behind rendering and spatialization.

2. DESIGN AND IMPLEMENTATION

The IVES toolkit's technical design follows the idea of a modular system. It consists of higher-level GUI-based blocks, as introduced with the BEAP and VIZZIE toolkits, which are integrated in the Max programming environment. This modularity allows the user to build up individual signal-processing chains by connecting modules with patch cords. The visual programming paradigm used in Max, with programming instructions as objects/nodes that are also connected to each other by patch cords, is very similar to this principle. This facilitates the adaptation and implementation of such a modular system paradigm. While the Max programming language can be considered to be already very high-level, the main difference of the here presented blocks is that they do not require further lower-level programming of functionality. Instead, they represent modules that can be connected in the order of the required processing chain and parameterized over their GUI. In its current development state, the available modules in IVES can be divided into 3 categories:

- Spatial audio modules to create and process a spatial sound field. Those modules provide the required rendering chain for loudspeaker- or headphone-based reproduction. Furthermore, the modules allow to transform and interact with the sound field in the spatial audio domain.

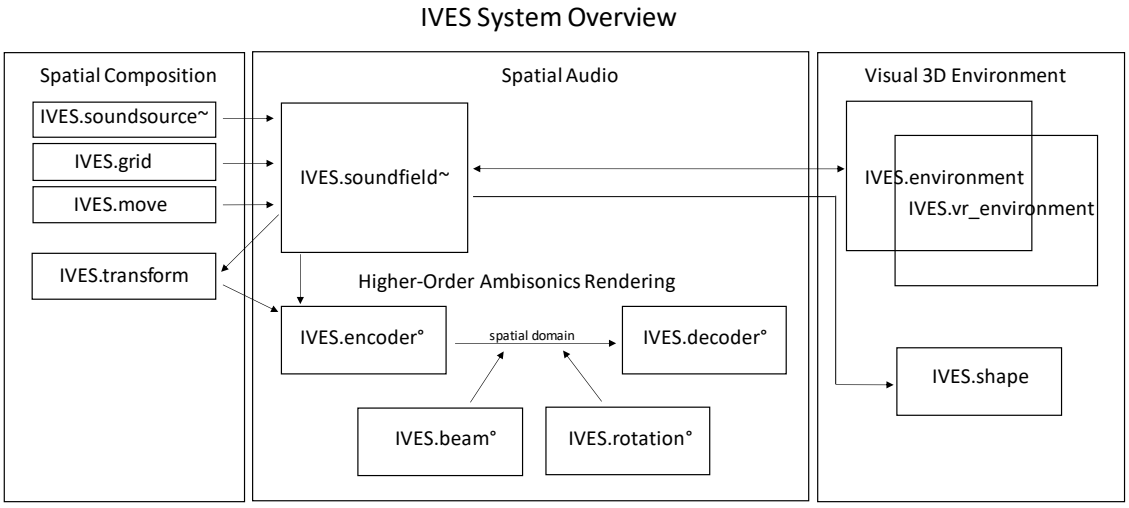

Figure 1. IVES system overview. The diagram shows the architecture with the spatial audio rendering as the center of the toolkit. It shows the connections of spatial composition, as well as visual 3D environment modules to the spatial audio rendering.

- Visual 3D environment modules that set up a virtual world with the needed rendering chain implemented. The modules allow to create 3D objects in the virtual environment and link them to virtual sound sources.
- Control and interaction modules, for example, to generate grids/patterns of object positions and trajectory movements, transform object parameters, and simulate/control head rotation parameters.

The system architecture of IVES (see Fig. 1) is centered around the virtual sound field. This indicates the slightly stronger focus on audio in IVES, as opposed to the game engines discussed. The visual environment is linked to the data of the virtual sound field. Virtual sound sources are automatically represented as visual objects in the virtual environment and can be linked to added 3D objects. For example, parameters such as the listener's position and head orientation are synchronized as position and view of the camera in the virtual environment. Still, the visual environment, as well as the spatial sound field, can be used and parameterized independently and self-sufficient.

In the current development state, the externals of three libraries are used to program the modules for creating, interacting, and rendering audiovisual virtual environments in Max.
The OpenGL [19] objects of the Jitter library by Cycling'74, integrated in the Max 8 (v.8.1.10) programming environment, are used to render a basic visual virtual environment and place 3D elements in it. The VR library by Graham Wakefield (v.1.0.1) is used to integrate PC-VR systems, compatible with SteamVR, Oculus, or Vive drivers, and render the virtual environment on HMDs and use tracking for 6DoF rendering of the virtual environment.
The sound spatialization is implemented using the Spat 5 (v.5.2.1) library by IRCAM. It is used to set up the virtual sound field and an Ambisonics-based [20] processing chain to manipulate and render the sound field for loudspeaker-based or binaural headphone reproduction. Whereas Max 8, as well as all implemented libraries, are compatible with Microsofts Windows and Apples MacOS platforms, the VR functionality has only been tested on Windows 10 operating systems.

2.1 Spatial Audio with Spat 5

The Spat library by IRCAM is a sophisticated library providing many externals for sound spatialization and spatial composing. It can be used for many different spatialization scenarios, such as reproduction on loudspeaker- or headphone-based systems. It offers all necessary functions and algorithms for spatial en- and decoding, reverberation, sound processing, plotting and visualization, as well as tools for motion and trajectories of sound sources. Even though the Spat library is already providing an all-in-one sound spatializer (*spat5.spat~*) with a graphical interface, reverberation, and a fully integrated rendering chain, it can be useful for many situations to implement separated or different processing chains. As all elementary spatialization algorithms and functions are also provided as separated externals, one of the main purposes of the Spat library is to enable the programming of individual spatializers, tailored to different application scenarios (see, for example, different Spat-based spatializers as Max4Live devices [21, 22]).
IVES is also built on a custom implementation to achieve the flexibility of a modular system and to allow the reproduction of simulated sound fields or multi-channel audio from spherical microphone arrays. It allows transforming the sound field, as well as the reproduction on various multi-channel loudspeaker systems or headphones. IVES uses the Spat externals with the MC multi-channel system

introduced in Max 8, providing bundled multiple signal chords to simplify patching and connection of the different audio modules. The number of channels is automatically adapted by IVES modules to the used input channels or the spatial sound field order.

2.1.1 Sound Fields in IVES

IVES.soundfield∼ represents one of the core modules of the toolkit and can operate in two different modes:

- Virtual Sound Sources:
  This mode handles spatial parameters of the listener as well as virtual sound sources and their signal processing. It uses the *spat5.viewer* object to manage the position of sound sources and the listener in the virtual space. Additionally, the listener orientation is considered to enable full 6DoF roaming with dynamic binauralization in the appended rendering process. Sound sources can also be freely placed in the sound field coordinate system. To enable distance perception of the virtual sound sources, gain attenuation is implemented and applied to the source signal. The distance between the listener and each source is calculated, resulting in a gain factor based on the inverse-distance law.
  In addition to the distance attenuation, the source pre-processing provided by the Spat library's *spat5.source∼* object is implemented to generate a continuous pre-delay based on distance to produce a Doppler effect. The processed signals of the sources, as well as the spatial parameters of all elements in the coordinate space, are subsequently passed to the spatial encoder.

- Multi-Channel Audio:
  The IVES toolkit also supports rendering of sound fields recorded with spherical microphone arrays. Thus, this mode of the sound field module aids the playback of multichannel audio files containing such recordings. It is dynamically generating MC signal streams according to the corresponding channel number. Alternatively, for example for real-time input, the input stream can directly be connected to the spatial encoder.

2.1.2 Higher-Order Ambisonics En- and Decoding

The spatial audio rendering is performed with a Higher-Order Ambisonics (HOA) processing chain (see Fig. 2). It consists of an encoder and decoder module that can be applied to the described above modes and used for different reproduction scenarios.
HOA represents the Ambisonics 3D-audio format in spatial resolution orders greater than one. Ambisonics is an object-based full-spherical spatial sound format, that, unlike channel-based systems, is agnostic to channel numbers or speaker positions. While higher numbers of channels and loudspeakers lead to higher orders and resulting spatial resolution, the decoding can be adopted to various reproduction scenarios. The HOA spatial format was chosen as it can encode single virtual sound sources as well as microphone array recordings. The encoding results in the B-Format sound field representation, offering high flexibility of decoding to various loudspeaker layouts as well as binaural headphone reproduction. The Spat library offers HOA encoders for simulated sound fields and several common microphone arrays. Decoders are available for loudspeaker reproduction or as binauralizer for headphones.
The encoder module (*IVES.encoder°*), includes several HOA encoders provided by Spat 5. One HOA encoder for simulated sound fields and three encoders for microphone arrays. The Zylia ZM1, Eigenmike EM32, and 1st order A-Format microphones, such as the Sennheiser Ambeo VR mic, are supported. The encoders can be selected and parameterized with the module's GUI. Parameters required through the whole rendering chain, such as spatial order, dimension, or normalization, are passed automatically to subsequent modules.
The decoder module (*IVES.decoder°*), includes a HOA decoder for multi-channel loudspeaker reproduction as well as a HOA binauralizer for spatial headphone reproduction using head-related transfer functions (HRTFs).
For the loudspeaker decoder, a grid generator was implemented. It can generate loudspeaker positions for common grids and loudspeaker layouts as provided by the Spat library. To enable dynamic binauralization also for recorded sound fields with microphone arrays, head rotation parameters need also to be considered in the decoding process.
Because the HOA binauralizer provided in Spat 5 does not support parameters of head rotation for dynamic binauralization, a sufficient sound field rotation needs to be implemented. For this, IVES rotates the sound field around a static listener's head position (in contrast to a rotation from the listener's point-of-view). As the HOA rotation implemented in Spat 5 is by default intrinsic (following the 'ZYX' convention), the rotation parameters are converted to an extrinsic rotation order (following the 'XYZ' convention). This ensures an adequate sound field rotation for a listener's point-of-view with given Euler angles (yaw/pitch/roll) as head orientation parameters.

2.1.3 Spatial Domain Manipulation

The IVES toolkit also provides sound field manipulation modules operating directly on the spatial sound field. This sound field interaction, transformation, and manipulation are done in between the en- and decoding spatial processing, in the so-called spatial domain (also called Ambisonics or spherical harmonics domain). In its current development state, a rotation and beam-former module for transformations in the spatial domain are implemented in IVES. The rotation module provides a spherical Euler rotation of the given sound field around the listener. The *IVES.beam°* module provides an implementation based on the *spat5.hoa.focus* external, allowing a selective fragmentation of the sound field based on orientation and selectivity of the beam. It is based on the concept of beamforming, a spatial filtering technique that can be used to filter spatial

Figure 2. Spatial audio pipeline using IVES. The example shows a simple simulated sound field with one virtual sound source. It is based on 3rd order HOA rendering with binaural reproduction. It consists of a virtual sound field module with a single sound source module as audio input and the head rotation module for 3DoF motion tracking. The audio stream and position data from the sound field module is then spatialized using the en- and decoder modules.

sound with a given directivity.
Both modules can be parameterized in the GUI and also linked to external interfaces and controllers, which allows designing new forms of interactions and spatial compositions [23]. In contrast to spatial composition techniques such as the positioning and movement of single sounds in a simulated sound field, the techniques based on the interaction and transformation in the spatial domain can also be applied to recorded sound fields with microphone arrays.

2.2 Visual 3D Environments with Jitter and VR

The processing chain for visual 3D environments is based on Max Jitter and its implementation of OpenGL. Jitter is Max's integrated library for image, video and 3D graphics processing. Similar to the MIDI and audio processing objects with Max and the MSP library, Jitter offers objects for programming signal processing with visual elements. This includes 3D graphics using OpenGL, an open-source application programming interface (API) for 2D and 3D graphics rendering. To enable PC-VR systems, including the presentation on HMDs and integration of tracking data for 6DoF movement, the VR package for Max was integrated. VR is a third-party library made by Graham Wakefield and the Alice Lab for Computational Worldmaking at York University to integrate VR hardware and stereoscopic rendering on HMDs into the Max programming environment.
The core module for handling visual environments in IVES is *IVES.environment*. It is the visual equivalent to *IVES.soundfield~*. When connected to the *IVES.soundfield~* module, it synchronizes with the elements in the virtual sound field (number and position of sound sources, listener position and orientation for the

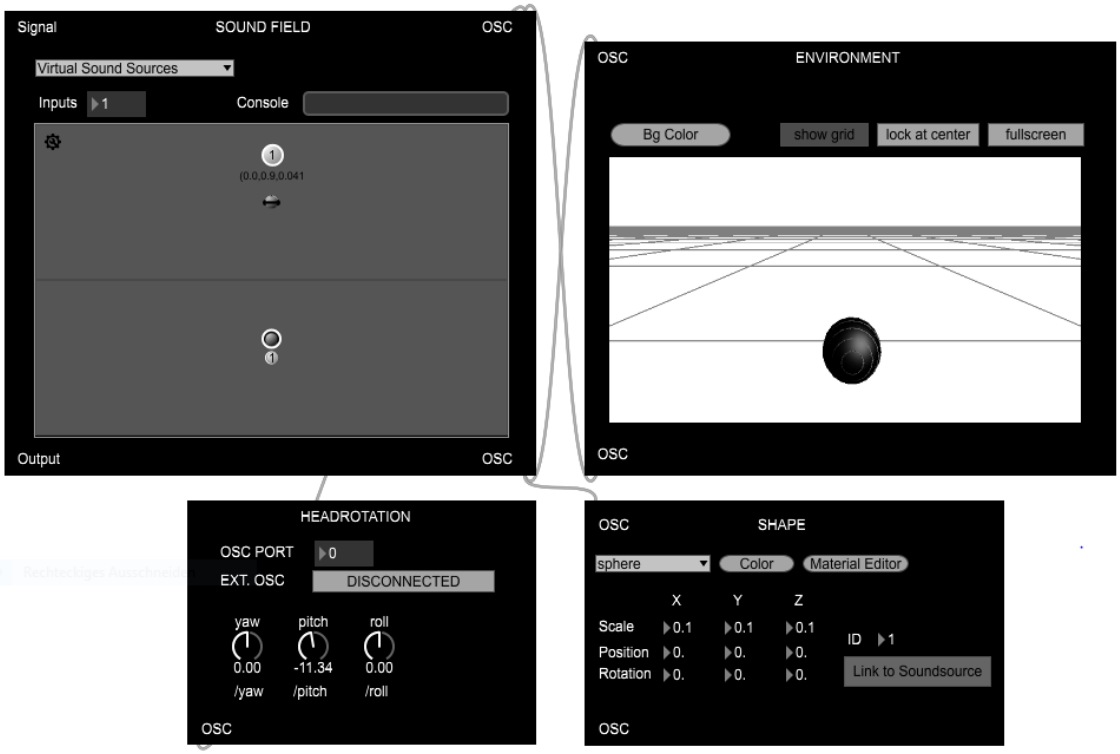

Figure 3. Visual 3D environment in IVES. The example shows a simple 3D environment with one sphere object linked to a virtual sound source. It consists of the sound field module as the core of the virtual environment. The environment module creates a visual representation of the sound field, a shape module renders a black sphere at the position of the given virtual sound source. The head rotation module is connected to the sound field and used to enable 3DoF motion in the audiovisual virtual environment.

viewer camera). The main difference to *IVES.soundfield~* is the implementation of the rendering chain inside the module. A basic environment with a world and physics context is set up, and the rendering of an empty world showing a grid. When sound sources are added to the sound field, they automatically show up as visual objects in the environment. The *IVES.shape* module allows to add further 3D objects. The module offers basic parametrization of geometry and material as well as transformation of the objects. The objects' position can also be linked to sound sources of the sound field (see Fig. 3).

In addition to the conventional environment module, the toolkit also offers the *IVES.vr_environment* module for using PC-VR systems. This module offers similar functionality as *IVES.environment*, but extends the rendering to HMDs using the VR library. The implementation is using the VR external with multiple cameras for each eye as documented in the package. The VR external also provides position and head-tracking parameters from the VR system. These are converted to the coordinate convention used in Spat and can be transferred to the listener's position and orientation by connecting the module to the *IVES.soundfield*.

2.2.1 *Spatial Parameter Conventions and Conversion*

Because the Spat library uses a different coordinate system convention than Jitter/OpenGL, all position data and parameters must be converted for the communication and data exchange in-between these libraries. The VR library uses the convention implemented in Jitter/OpenGL. Thus, for 6DoF audio spatialization, the tracking data must be converted to the Spat conventions accordingly.

All this conversion is done by the IVES modules automatically and all GUI elements of the modules use the convention implemented in the Jitter library to ensure consistency throughout the modules. The externals provided by the Spat library support parameters in spherical/navigational (azimuth, elevation, distance) and Cartesian coordinates (x-,y-,z-axis). Jitter also supports Cartesian coordinates, but the used convention is different.

Spat uses a right-handed coordinate system with the positive x-axis to the right, y-axis front to back, and z-axis up and down from the listener. The OpenGL Jitter externals are using the coordinate convention of the OpenGL API, a system with the positive x-axis to the right, y-axis up and down, and z-axis back and front from the listener. The Jitter/OpenGL convention was chosen as the leading coordinate convention. As Spat uses the spherical/navigational

Figure 4. Spatial composition in IVES. The example shows a grid generator (IVES.grid) creating circular positions for four virtual sound sources, and the IVES.move trajectories generator controlling the movement of the first sound source. Both modules are connected to the sound field module managing the data of the virtual sound field.

coordinate system by default, a conversion from spherical to Cartesian coordinates into Jitter/OpenGL convention, and vice versa, is implemented. Because Spat is interpreting Euler angles (yaw/pitch/roll) in relation to their Cartesian coordinate convention, they are also adapted to the convention used by Jitter/OpenGL.

2.3 Control and Interaction

To support the process of composition and interaction with the audiovisual environment, IVES also provides modules to create and transform spatial parameters of virtual objects by integrating further externals from Spat aided for spatial composition (see Fig. 4). The *IVES.move* module provides an implementation of Spat's trajectory external. The module allows the selection and parameterization of trajectory algorithms and apply them to sound sources, 3D objects, or the listener/viewer.
IVES.grid provides an implementation of the *spat5.grid* external, providing algorithms to generate different grids of position coordinates for sound sources as used in the *IVES.soundfield~* module. Spatial position coordinates of the elements described in the sound field can be transformed with the *IVES.transform* module, providing an implementation of the same-named Spat external.
The *IVES.soundsource~* module allows the handling of individual sound sources. It provides a sound player with different parameters for playback of sound files. Alternatively, an input signal can be passed through or a noise test signal can be generated. Furthermore, the position in Cartesian coordinates can be parameterized.

3. DISCUSSION AND CONCLUSION

In this paper, we described the first pre-release version of the IVES toolkit [2] . At the current state, the toolkit already provides all basic modules necessary to create a simple audiovisual virtual environment for screen-based or virtual reality applications with loudspeaker- or headphone-based spatial audio reproduction.
While the basic 3D environment, sound spatialization and VR features are already implemented in IVES, the visual world creation modules are still quite limited, especially compared to game engines. Although the future development aims to further develop and expand visual capabilities, the goal is not to create a full-scale alternative to game engines. Instead, the IVES toolkit will concentrate on single scene environments for artistic application, generative 3D visuals and sounds, as well as genuine audiovisual interaction concepts, rather than complex multiple-scene worlds, sophisticated rendering and simulation algorithms or scripted interactions. The idea is to encourage the development of new artistic concepts focused on sound and music in the field of audio-visual 3D environments and VR, without overwhelming the artists with standard features from game development.
As is the case with most software of this type, continuous expansion and further development is intended. Be-

[2] Available: `https://github.com/AudioGroupCologne/IVES/releases/tag/0.1-smc`

sides the usual bug fixes, improvements and simplifications as well as enhancements with different focuses are planned and/or already in development. To enable the creation of more complex visual environments, additional modules for composition with 3D objects are in development. Also, additional interfaces for further controllers and tracking devices, as well as for software for spatial composition (e.g., Iannix [24]) will be created. Future development will focus on modules for composition with the sound fields in the spatial domain (cf. Sec. 2.1). This approach is less explored compared to the composition of simulated sound fields (positions, motion) and offers the potential for new techniques for sound processing, composition, and the development of spatial musical instruments [25]. The most recent and future versions of IVES are available at: `https://github.com/AudioGroupCologne/IVES`

Acknowledgments

The work has been carried out within the project EarKAR, funded by EFRE (European Funds for Regional Development) under the funding reference code EFRE-0801444.

4. REFERENCES

[1] M. A. Baalman, "Spatial composition techniques and sound spatialisation technologies," *Organised Sound*, vol. 15, no. 3, pp. 209–218, 2010.

[2] "Holoedit," http://dvlpt.gmem.free.fr/web/static.php?page=HoloEdit_main, accessed: 2021-03-03.

[3] C. Bascou, "Adaptive spatialization and scripting capabilities in the spatial trajectory editor Holo-Edit," in *Proceedings of the 7th Sound and Music Computing Conference (SMC)*, 2010, pp. 1–6.

[4] "Zirkonium," https://zkm.de/en/about-the-zkm/organization/hertz-lab/software/zirkonium, accessed: 2021-03-03.

[5] C. Miyama, G. Dipper, and L. Brümmer, "Zirkonium 3.1 - A toolkit for spatial composition and performance," in *Proceedings of the International Computer Music Conference (ICMC)*, 2016, pp. 312–316.

[6] "Spat," https://forum.ircam.fr/projects/detail/spat/, accessed: 2021-03-03.

[7] T. Carpentier, "A new implementation of spat in Max," in *Proceedings of the 15th Sound and Music Computing Conference (SMC)*, 2018, pp. 184–191.

[8] "ICST Ambisonics Toolkit," https://www.zhdk.ch/forschung/icst/software-downloads-5379/downloads-ambisonics-externals-for-maxmsp-5381, accessed: 2021-03-03.

[9] J. C. Schacher, "Seven Years of ICST Ambisonics Tools for MAXMSP - A Brief Report," in *Proceedings of the 2nd International Symposium on Ambisonics and Spherical Acoustics*, 2010, pp. 1–4.

[10] M. Wozniewski, Z. Settel, and J. R. Cooperstock, "AudioScape : A Pure Data library for management of virtual environments and spatial audio," in *Proceedings of the Pd Convention*, 2007, pp. 1–6.

[11] "Cycling'74 Max," https://cycling74.com/, accessed: 2021-03-03.

[12] "PureData," http://msp.ucsd.edu/index.htm, accessed: 2021-03-03.

[13] "SuperCollider," https://supercollider.github.io/, accessed: 2021-03-03.

[14] I. I. Bukvic and J.-S. Kim, "μ Max-Unity3D Interoperability Toolkit," in *Proceedings of the International Computer Music Conference (ICMC)*, 2009, pp. 1–5.

[15] R. Graham and S. Cluett, "The soundfield as sound object: Virtual reality environments as a three-dimensional canvas for music composition," in *Proceedings of the AES International Conference on Audio for Virtual and Augmented Reality*, 2016, pp. 1–7.

[16] G. Santini, "Composing space in the space: An Augmented and Virtual Reality sound spatialization system," in *Proceedings of the 16th Sound and Music Computing Conference (SMC)*, 2019, pp. 229–233.

[17] "VR library for Max," https://github.com/worldmaking/vr, accessed: 2021-03-03.

[18] G. Wakefield and W. Smith, "Cosm : a Toolkit for Composing Immersive Audio-Visual Worlds of Agency and Autonomy," in *Proceedings of the International Computer Music Conference (ICMC)*, 2011, pp. 1–8.

[19] "OpenGL," https://www.opengl.org/, accessed: 2021-03-03.

[20] F. Zotter and M. Frank, *Ambisonics: A Practical 3D Audio Theory for Recording*. Cham: Springer, 2019.

[21] "Talk on Spat4Sat, by Fraction / Éric Raynaud," https://medias.ircam.fr/x80fd02, accessed: 2021-03-03.

[22] "binaural_spat, by Kasper Fangel Skov," https://github.com/kasperskov/binaural_spat-v1.0, accessed: 2021-03-03.

[23] D. Dziwis, T. Lübeck, and C. Pörschmann, "Body-controlled sound field manipulation as a performance practice," in *Proceedings of the 148th AES Convention*, 2020, pp. 1–6.

[24] "Iannix," https://www.iannix.org/en/, accessed: 2021-03-03.

[25] A. Pysiewicz and S. Weinzierl, "Instruments for spatial sound control in real time music performances. a review," in *Musical Century in the 21st Instruments*, T. Bovermann, A. de Campo, H. Egermann, Sarah-Indriyati, Hardjowirogo, and S. Weinzierl, Eds. Berlin: Springer Nature, 2017, pp. 273–296.