Drile: An Immersive Environment for hierarchical live-looping

Florent Berthaut, Myriam Desainte-Catherine, Martin Hachet

▶ **To cite this version:**

Florent Berthaut, Myriam Desainte-Catherine, Martin Hachet. Drile: An Immersive Environment for hierarchical live-looping. New Interface for Musical Expression, Jun 2010, Sydney, Australia. page 192. hal-00530071

HAL Id: hal-00530071
https://hal.science/hal-00530071

Submitted on 27 Oct 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

DRILE: an immersive environment for hierarchical live-looping

Florent Berthaut
University of Bordeaux - SCRIME - LaBRI
berthaut@labri.fr

Myriam Desainte-Catherine
University of Bordeaux - SCRIME - LaBRI
myriam@labri.fr

Martin Hachet
INRIA - LaBRI
hachet@labri.fr

ABSTRACT

We present Drile, a multiprocess immersive instrument built upon the *hierarchical live-looping* technique and aimed at musical performance. This technique consists in creating musical trees whose nodes are composed of sound effects applied to a musical content. In the leaves, this content is a one-shot sound, whereas in higher-level nodes this content is composed of live-recorded sequences of parameters of the children nodes. Drile allows musicians to interact efficiently with these trees in an immersive environment. Nodes are represented as *worms*, which are 3D audiovisual objects. Worms can be manipulated using 3D interaction techniques, and several operations can be applied to the live-looping trees. The environment is composed of several virtual rooms, i.e. group of trees, corresponding to specific sounds and effects. Learning Drile is progressive since the musical control complexity varies according to the levels in live-looping trees. Thus beginners may have limited control over only root worms while still obtaining musically interesting results. Advanced users may modify the trees and manipulate each of the worms.

Keywords

Drile, immersive instrument, hierarchical live-looping, 3D interaction

1. INTRODUCTION

In the past decade, the live-looping technique has become more and more used in musical performances, either for solo singers who create their accompaniment or for instrumental or electronic improvisations. However, it has some limitations due to the use of hardware controllers, such as the impossibility of creating and manipulating complex musical structures or the impossibility of modifying all the parameters of recorded loops. We believe that 3D immersive environments can support the evolution of this technique into a more advanced system for musical performances. Indeed these environments provide interesting possibilities for 3D interaction with multiprocess instruments, i.e instruments that are composed of several sound synthesis processes. They also enable new way of building, visualizing and navigating complex musical structures as 3D shapes. Immersion, e.g using large stereoscopic displays, may be as valuable for the audience as it is for musicians. For example, comprehension of what musicians are doing and involvment in the musical performance may be improved by the use of stereoscopic vision and large screens. Finally, these immersive environments are often used in other fields for collaborative tasks, and thus they may be appropriate for collaborative musical performance.

In section 3, we describe an evolution of the live-looping technique, called *hierarchical live-looping* technique. Then, in section 4, we present *Drile*, our immersive musical instrument which relies on this technique. As it can be seen on figure 1, the musician wears 6DOF tracked stereoscopic glasses, and he interacts with the instrument, displayed on a large screen, using the *Piivert* input device. We define the components of our instrument in section 4.2, then we describe how we represent and interact with the 3D *live-looping trees* in section 4.3. Finally, we present the live-looping scenes in section 4.4 and the collaboration/learning possibilities brought by our approach in section 4.5.

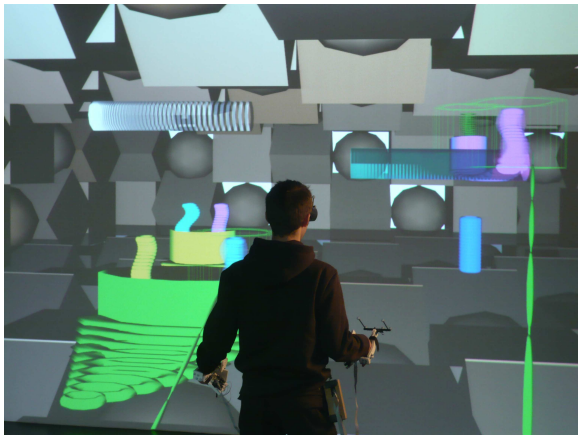

Figure 1: Drile used by one expert musician.The live-looping scene contains a three-level tree on bottom left, a two-level tree on top right, 1 leaf node/worm and 2 tunnels(scattering/reverb and color hue/pitch).

2. RELATED WORK

2.1 3D Virtual Instruments

Many existing immersive instruments focus on navigation in musical environments, like the virtual groove in the Phase project [10] or the audiovisual grains in Plumage [3]. These applications allow musicians to play precomposed musical structures, but they do not give access to the structure of the synthesis processes.

Other immersive instruments are single process instruments, i.e. instruments that allow to interact with only one synthesis process, such as the Virtual Xylophone, the Virtual Membrane, or the Virtual Air Guitar developed by Mäki-Patola et al. [4] and the sculpting instruments developed by Mulder [6]. The application devel-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
NIME2010, June 15-18, 2010, Sydney, Australia
Copyright 2010, Copyright remains with the author(s).

oped by Mike Wozniewski et al. [12] relies on users movements to either control the spatialization of pre-recorded sound sources, or apply effects on the sound of an accoustic instrument. All these applications aim at being ordinary instruments, with more control over synthesis processes than navigation tools, that one may use for musical performance. However, they do not take advantage of the possibilities brought by 3D environments in terms of multiple processes handling and structures creation.

An interesting application designed by Polfreman [8] takes advantage of 3D environment to build complex musical scores including hierarchical organisation of elements. Nevertheless it is not designed for musical performance but rather for composition.

2.2 Live-Looping

Live-Looping is a musical technique that consists in recording musical loops from an audio or control(e.g MIDI messages) input and stacking these loops to quickly build musical structures or sound textures. It has several advantages over other electronic playing techniques such as triggering sequenced loops, or applying effects to different tracks of a prepared song. First of all, there is usually no temporal quantization so it enables more natural musical patterns as opposed to midi sequences. It may fit any musical genre since any musical input and any time-signature can be used. Finally, one can rely on prepared loops or improvise every part of a musical structure. It is mostly used for live performances by different instrumentalists such as guitar players, beat-boxers, or electronic musicians[1]. It is also an interesting tool for music writing, allowing quick sketching of songs.

Live-Looping originates from tape-delay systems and was explored by contemporary composers such as Terry Riley or Steve Reich. This looping system was then implemented in hardware effects racks or guitar pedals as the Gibson Echoplex, Digitech Jamman, Boss RC20 and so on. Usual operations include record, play and stop/mute. One can also overdub, i.e add material to an existing loop, multiply the length of a loop, and reverse a loop. Live-looping has evolved with digital loopers, such as Sooperlooper[2] or Freewheeling[3], which include new possibilities such as synchronization, timestretching, graphical interfaces and so on.

Most live-looping systems deal with audio input, so that the loops are audio buffers. However, some of them, such as LiveLoop[4], deal with control inputs, such as MIDI or OpenSoundControl messages or other events. They record and playback series of events, which may be either sound triggers or synthesis/effects parameters controls. This control live-looping has many advantages over audio live-looping. First of all, loops can be easily modified since each event is separable. This enables timestretching and addition/removal of events. Notes and effects controls may be recorded independently, so that they can be modified separately. Finally the same recorded loops can easily be rerouted on different synthesis processes.

Some novel instruments are based on live-looping. For example, the BeatBugs [11] are small input devices that record rhythms played by users and repeat them. Recorded rhythms can then be modified and several devices can be synchronized, enabling collaborative musical interaction. Another very interesting application is Fijuu [7], in which users manipulate 3D audiovisual shapes using a gamepad. Each shape is associated with a specific sound synthesis process which is triggered when the shape is distorted. Audiovisual effects can also be applied on produced sounds and audio loops can be recorded for each shape. These loops can then be accelerated, muted and amplified. However, creation and modification of musical structures remain limited, as well as the interaction possibilities.

[1]http://www.livelooping.org/
[2]http://www.essej.net/sooperlooper/
[3]http://freewheeling.sourceforge.net/
[4]http://code.google.com/p/liveloop/

3. HIERARCHICAL LIVE-LOOPING

Basic live-looping does not enable the creation and manipulation of complex musical structures, but only lists of recorded sequences. It also reduces the possibilities of advanced modification of recorded loops, especially when the content is audio data. We propose to expand the advantages of control live-looping with what we call *hierarchical live-looping*. The tree structure used in *hierarchical live-looping* is based on the study of Marczak [5]. This section focuses on the concept of hierarchical live-looping while section 4 describes its implementation as a 3D immersive instrument.

3.1 Principle

We define *live-looping trees* containing leaves $l :< c_l, x >$ and nodes $n :< ch, c, x >$. x is a list of audio effects. c_l is a one-shot, i.e no sequences or loops, musical content which can be a soundfile or another synthesis process. ch is a list of children nodes. c is a musical content composed of live-recorded sequences of children events. These events are effects parameters and triggers of the musical content. The audio effects available for all nodes are pitch, volume, distortion and reverb. In addition to these audio effects, the tempo of sequences can be modified, and other high-level musical effects could be added. Each node and leaf also has a content play mode, which can be *trigger*, i.e the content is played until its end, *normal*, i.e the content is played until its end or the reception of a stop event, and *loop*, i.e the content is looped until it receives a stop event. The loop mode is not used for leaves,

Control and audio data follow different paths in *hierarchical live-looping*. Control data go top-down the tree because nodes contain their children sequences. Audio data go bottom-up the tree, which means that the sequences of sounds contained in the leaves will be successively modified by the effects of their parent nodes. The audio and control data flows and the structure of a *live-looping trees* can be seen on figure 2.

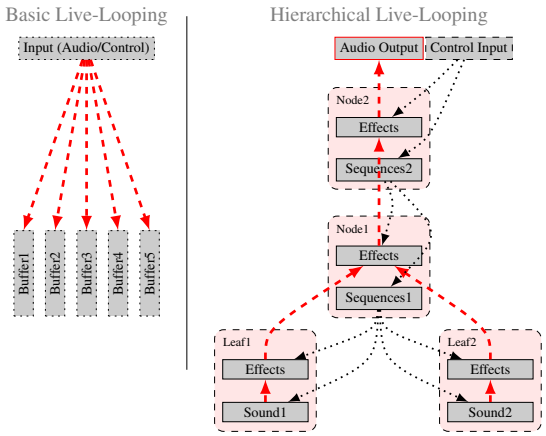

Figure 2: Basic live-looping data flow (left) and hierarchical live-looping control (in black) and audio (in red) data flows.

Hierarchical live-looping has several advantages over traditional live-looping approaches. First of all, it allows musicians to build musical sequences of any complexity by creating and manipulating trees. A simple example can be seen in figure 3.

It also provides direct access to any level of the trees. One may trigger only the snare or kick to produce a fill, or the drums node to completely modify the rhythm. Effects can be applied to any node independently, whether they are leaves or high-level nodes, in which case the effects will affect all their children. For example one may filter all the drums by modifying the drums node, or

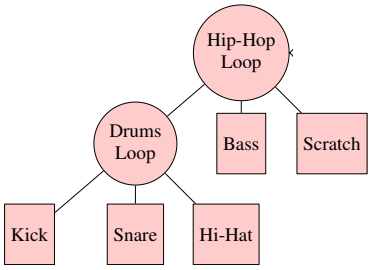

Figure 3: Hip-Hop live-looping tree example.

pitchshift only the hi-hat.

Live-looping trees also adapt to user with different levels of experience. Beginners can play with tree roots, thus triggering musically interesting parts and applying simple effects to it. Advanced users can go down the tree and manipulate each node. They may change any sequence by re-recording it, or modify the effects associated to the different nodes.

Furthermore, advanced users can easily collaborate to build musical structures. For example, they may add nodes to existing trees, or duplicate nodes to remix loops prepared by others.

Finally, several live-looping trees can coexist, allowing musicians to experiment with non-synchronized loops. This may be useful to create complex sonic textures.

3.2 Node operations

Two operations are available for the nodes. Nodes content can be triggered, and nodes audio effects can be modulated. Triggering a leaf node will play its sound whereas triggering a higher-level node will play its sequences of events and parameters. Audio effects parameters can also be modified. In leaves, this will directly affect the musical content, i.e the audio file, whereas in higher-level nodes it will affect the resulting audio output of all children. Musical effects, such as tempo modification, affect higher-level nodes sequences.

3.3 Tree operations

We propose a set of operations for the live-looping trees, which are shown in figure 4: *build, merge, duplicate, extract*.

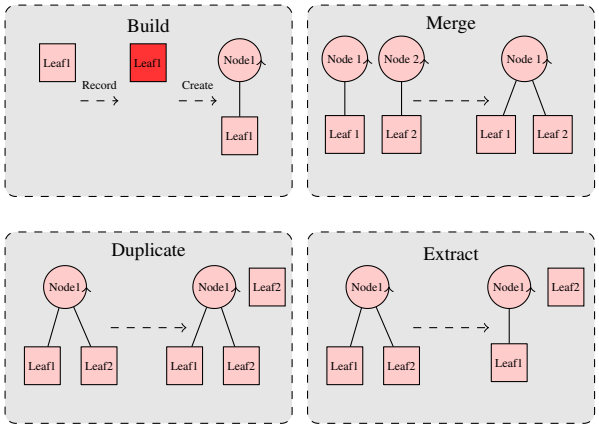

Figure 4: Hierarchical live-looping operations.

The *build* operation has two steps. The first step starts the recording of the events of a node. The second step stops the recording, creates a parent node and define the recorded sequences of events as its content. The playing mode of the parent content is automatically set to *loop* so that its children sequences keep playing, as it would occur with a traditional live-looping application. The playing mode can then be changed to *trigger* or *normal*. When a build operation is performed on a node that already has a parent, this operation simply records sequences over existing ones.

The *merge* operation consists in linking two trees by merging two parent nodes into a single parent node. The oldest sequence of these nodes, i.e the one which was recorded first, is taken as a pulse. Another sequence could be chosen as the pulse, but the usual playing technique in live-looping is to first record the pulse, then the other loops. All the other sequences are then synchronized to multiples of this pulse according to their length, as depicted in figure 5.

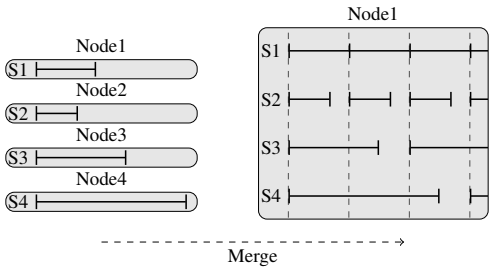

Figure 5: Example of sequences synchronization when merging several nodes. Here the oldest sequence is S1.

Thus the parent node synchronizes all its children together, which allows musicians to build regular rhythms. The *build* and *merge* operations can be combined if multiple sequences are recorded at the same time. In that case, all simultaneously built parent nodes are merged into a single node. One must note that this operation is not possible if both nodes have parents, since it may lead to too much changes in the live-looping tree.

The *duplicate* operation consists in duplicating a part of a live-looping tree. This is a useful operation because hierarchical live-looping results in nodes being locked to their parents content, as one node can not have two parents, for musical structure understanding reasons. Thus nodes can not be used in two different sequences at the same time. Using the *duplicate* operation, one can copy a node in order to use it to build another sequence, with different effects parameters for example.

With the *extract* operation, one may remove any node and its children from a live-looping tree, making it the root of a new tree, or a leaf if it has no children.

3.4 Implementation

Hierarchical live-looping adds new theoretical possibilities to basic live-looping. However, its relevance heavily depends on its implementation. Indeed operations on trees and manipulations of nodes need to be efficient. 3D immersive environments provide new possibilities in terms of interaction, visualization and immersion. They are thus fitted to the control of complex musical structures, especially in a performance context.

4. DRILE

Drile implements the *hierarchical live-looping* technique in a 3D immersive environment. In this implementation, leaves contain multisampled sounds which can be played normally or using granular synthesis. Live-looping trees are represented by 3D virtual structures. Musicians interact with these structures in front of a large screen. They are immersed in the 3D environment thanks to stereoscopic glasses and head-tracking. A video of Drile can be watched on http://vimeo.com/9206485.

4.1 Technical setup

Drile is composed of three applications which run on Gnu/Linux: drile-audio, drile-ui and drile-ui-display. Drile-audio handles the

sound processes and it outputs audio via the Jack sound server. It also uses LV2 audio effects plugins, and VAMP audio analysis plugins. Drile-ui handles the scene graph of the 3D environment. They may run on two separate computers and communicate via OpenSoundControl messages. The stereoscopic display is done by two instances of drile-ui-display synchronized with drile-ui and connected to separated projectors, one for each eye. In addition, one user (or several with split screens) can be equipped with tracked glasses, to improve immersion and interaction with the environment. Users wear passive stereoscopic glasses and the sound is played using an external usb soundcard and two active speakers.

4.2 Worms, Tunnels and Piivert

In Drile, nodes of live-looping trees are represented by what we call *worms*. These worms, which can be seen on figure 1, have several graphical parameters such as color hue, size, transparency, scattering and so on. These parameters are associated to the nodes audio effects parameters, so that modifying the worms appearance modifies the nodes audio effects. Current mappings are size/volume, color hue/pitch, transparency/distortion and scattering/reverb. They were chosen as a result of a user study, following user preferences and trying to preserve the independance of graphical perceptual dimensions. One must note that the mappings are relative, not absolute. For example, two worms with the same color hue do not necessarily have the same pitch, but their initial pitch is modified by the same amount. Graphical parameters thus give the current value of the effect, like a cursor on a graphical slider, which is essential for efficient interaction.

Worms shapes are associated to the analysed spectrum of nodes audio outputs, to allow users to identify which worm is associated to a node. Finally, each worm rotation on the y-axis is associated to the position of the read pointer in its associated node content. This allows musicians to follow the playback of nodes sequences.

Worms graphical parameters, and thus their associated sound parameters, can be modified by grabbing and sliding the worms through what we call *tunnels*. Two of them can be seen on figure 1. Each tunnel is associated to one or several graphical parameters, and one or several scales for each parameter. The tunnels are made of a succession of thin cylinders which reflect the values of the parameters scales. The scales can be continuous or discrete. For example, if one associates size with pitch, one may define musical notes with an array of sizes. One may modify only one worm at a time by passing it through a tunnel, but one may also grab and move a tunnel to modify several worms at the same time, for example to mute several loops.

To interact with the environment, we use an input device called Piivert [1]. Piivert is a bimanual input device that combines 6DOF tracking, using infrared targets detection, with high-sensitivity pressure sensing. It allows musicians to benefit from the possibilities of graphical interaction while preserving accurate musical controls. Force-resistive sensors are positionned under the thumb, index, middle and ring fingers, allowing us to detect different gestures. These include low-level gestures such as hit and pressure, and high-level percussion gestures such as flam, three-strikes roll and four-strikes roll. This device allows us to select and grab worms and tunnels using a ray-casting technique, i.e a virtual ray that sticks out of the device and goes through the 3D environment. To grab a worm or tunnel, users perform a pressure gesture with their thumb. Low-level gestures transmit vibrations to the worms with the virtual ray, and thus trigger the associated node. Hit gestures play the entire content of the node while pressure gestures play the content using random small grains of the content, which results in granular synthesis for the leaves. Tunnels scale presets can also be changed by selecting the tunnels and performing a hit gesture. Finally, high-level gestures are used to trigger hierarchical live-looping operations. A user holding this device can be seen in figure 1.

4.3 Hierarchy

Common 3D selection and manipulation techniques such as the virtual ray technique are more efficient for objects at short distances, as evaluated by Poupyrev et al. [9]. Furthermore, as the size parameter is used to control the volume of worms, allowing continuous modification of worms depth may disrupt the perception of worms size. Thus we propose to keep most worms and tunnels on what we call the *interaction plane*, at a fixed distance. This preserves spatial and temporal accuracy for musical actions such as selecting and manipulating worms and tunnels. Depth and navigation in the environment may then be used for actions that require less temporal accuracy.

4.3.1 Live-looping trees

As depicted in figure 6, live-looping trees are represented by connected worms organized by depth. Each level of the tree is displayed at a discrete depth. The level that is the closest to the users is on the *interaction plane*. Worms children are placed in kind of appendages. Musicians may only select and grab the worms on the *interaction plane* and their direct children. By physically moving in front of the screen, musicians can easily point directly at children with the virtual ray, thanks to head-tracking. When grabbed, the children jump to the depth of the *interaction plane* so that they can be modified using the *tunnels*. To access lower-level and higher-level worms, musicians respectively pull and push one of the worms, as depicted in figure 6. When going down the tree, high-level worms move towards users and disappear. Their appendage remain visible, to indicate that displayed worms are children. When users grab and move the root worm of a tree, all the tree follows its translations. When users grab a children worm, move it and release it, it jumps back to its parent appendage.

The immersive environment allows us to display the live-looping hierarchy without overloading the interface and disrupting the interaction. Indeed, manipulation of nodes is done within the *interaction plane* whereas access to the trees levels relies on the depth of the environment and on the use of head-tracking. These simultaneous manipulations of trees and nodes would be more complicated with a 2D interface, since they could not be done in separated dimensions.

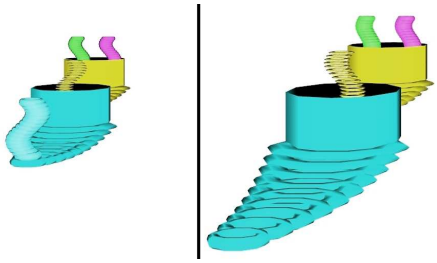

Figure 6: Three-levels live-looping tree (left). Pulling a worm goes down the tree (right).

4.3.2 Tree Operations

The *build* operation needs to integrate closely in a sequence of musical gestures since it records a musical loop. Thus we have defined a high-level gesture with Piivert, actually a ring-middle finger flam. First the user needs to select a worm with the virtual ray. When the record step is triggered, the worm changes shape to highlight its rotation around the y-axis. When the create step is triggered, the created parent node appears and the child worm moves behind it, in its appendage, as depicted on figure 7.

The *merge* operation is a less temporally critical operation. This it is done by colliding worms. When two worms collide for more than 1 second, the system will try to merge them. All children of one of the worms are transferred to the other, and this empty

node is deleted from the scene. Children worms are automatically arranged behind their parent worm in its appendage, as depicted in figure 7.

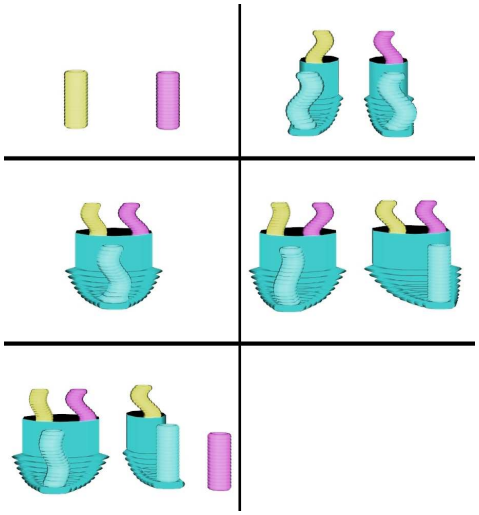

Figure 7: Live-looping trees operations (from right to left, top to bottom): Build, Merge, Duplicate, Extract

The *duplicate* operation is also done graphically by grabbing a worm with two rays, i.e two hands, and stretching it, as it can be seen on figure 7. A copy of the stretched worm and all these children, with all contents, is then added to the scene. The stretched worm thus become the root of a new live-looping tree, and is brought to the *interaction plane*.

Since the *extract* operation is the opposite of the build operation, it is also triggered by a high-level Piivert gesture. This gesture is a flam with the same fingers but in the opposite direction, i.e middle finger before ring finger. The extracted worm is brought to the *interaction plane*, its children remaining behind it, as depicted in figure 7. This results in a new live-looping tree.

4.4 Live-looping scenes

As explained in section 2.2, the live-looping technique often includes the idea of *scenes*. A scene is composed of several loops which are synchronized, or which simply fit together musically. With hardware systems, like guitar pedals, scenes are selected using buttons or rotary encoders. Thus each scene is only associated with a number. With 2D looping software, like Freewheeling, scenes are selected with a menu, and are associated with a name. Their musical features, such as tempo or mood, may thus be better described.

In Drile, we define scenes as 3D rooms containing several worms, which can be leaves or high-level nodes of pre-built live-looping trees, and several tunnels.

When the user walks backward from the screen and reaches a specific distance, the camera moves backward, revealing the grid of scenes, as depicted in figure 8. A scene can then be selected simply by looking at it. The head movement needs to be a little exagarated though, as the selection is computed from the position and orientation of the user's head. Finally, the user move to the selected scene simply by walking towards the screen.

Such an approach has several advantages. Visual properties of each scene/room, such as colors, 3D objects, walls and so on, may be used to reflect the musical "mood" of the live-looping trees. Users can refer to these properties to choose the musical content they want to play with. The appearance of the scenes is defined in a configuration file with a XML syntax.

Different tunnels can be chosen for each scene, as audio effects needs may be different for different musical content.

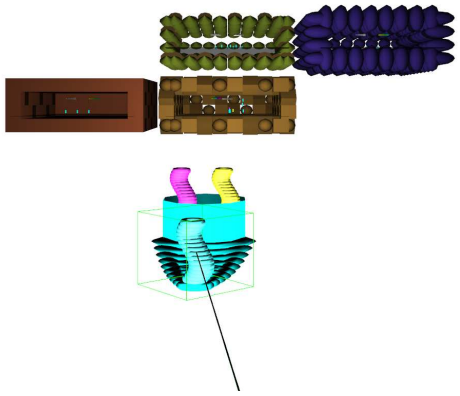

Figure 8: View of all scenes (here 4). Here one tree is being moved from one scene to another.

A worm can be brought from a previous scene, to make smooth transitions between musical parts. This worm can be a leaf, with a single sound, so that musicians may play it a last time while starting the new musical part. It may also be the root of complex tree, so that the new musical part will consist in modifying the previous tree by progressively adding new sounds.

Performances may follow very different paths, as any transition between scenes is possible and may be done at anytime.

Finally, each scene has a floor. Musicians can bypass worms or complete trees by moving them below this semi-transparent floor. Their audio data is then sent another stereo Jack output, which can be connected to earphones for example. This may be used as a monitor system to prepare trees and then play them, or it can be simply used to quickly mute trees.

4.5 Learning and Collaboration

Thanks to *hierarchical live-looping*, learning Drile is progressive. Indeed, following Birnbaum et al. classification of new instruments in dimension spaces [2], Drile happens to be between two different configurations, as depicted in figure 9. New users may begin by interacting only with the root worms of pre-built live-looping trees. This allows them to manipulate musically interesting sequences without much expertise. To push this idea further, we have developed an interaction technique in addition to the ones available with Piivert, which we call the *bucket*. Users manipulate a virtual bucket using a 6DOF tracked device with a single button. They may then grab and move only worms on the *interaction plane*. They are not allowed to trigger the sequences, but only pass these worms through tunnels, or bypass them. Interaction is quite simple, i.e grabbing and moving worms, and the musical possibilities are reduced to high-level processes manipulation. Thus required expertise, musical control and degrees of freedom are low.

When users get accustomed to this subset of Drile interaction techniques, they can switch to Piivert. By going down the tree, musicians progressively access rawer musical content, i.e simpler sequences and finally single sounds. Symmetrically, the degree of interaction needed to produce an interesting result, i.e the required expertise, rises. At first, they may interact only with the root worms, this time being able to trigger the musical content. When gaining expertise, they may access lower levels of pre-built live-looping trees, to trigger their nodes and make more complex manipulations. Finally, after more learning, they may modify trees structures by using the available operations, and even build new trees. Obviously required expertised increases as hierarchical live-looping operations and low-level worms manipulations are complex. The number of degrees of freedom also increases because Piivert enables new musical gestures and 3D interaction techniques.

Symmetrically, the musical control evolves from high-level control of a musical process to a complete control of the sound and musical parameters of each worm.

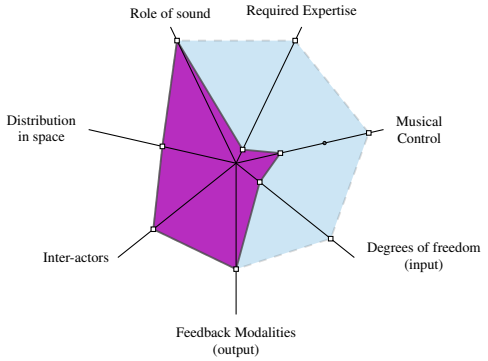

Figure 9: Drile Dimension Space, in purple for beginners with buckets and only access to root worms, in blue for advanced users with piivert and access to all worms of the live-looping trees.

Collaborative performances may rely on these different expertise levels. An advanced user can easily collaborate with one or several beginner users, as seen in figure 10. In this configuration, the expert builds live-looping trees, and pass them to the beginners so that they can manipulate worms at the foreground, i.e interesting musical sequences. As the bucket does not allow to manipulate the tunnels, the expert may also choose which tunnels should be used by the beginners.

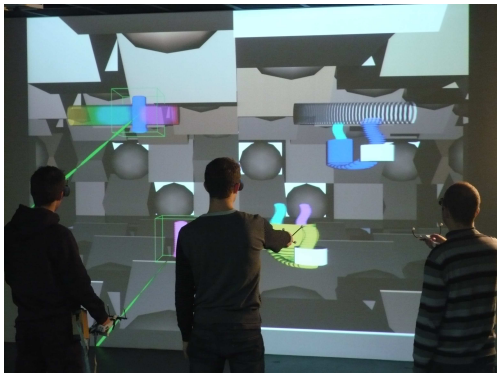

Figure 10: Collaboration between one advanced user with Piivert (left) and two beginner users with buckets

5. CONCLUSION

In this paper, we presented Drile, an immersive multiprocess instrument especially suited for musical performances. It relies on the concept of hierarchical live-looping, which enhances basic live-looping by allowing musicians to create and manipulate complex musical structures. Drile enables efficient use of this technique thanks to the possibilities brought by immersive environments. In particular, musicians make use of head-tracking, navigation in 3D environments and combination of hierarchy visualization with 3D interaction. Moreover, Drile is well adapted to learning and collaboration between users with different levels, because live-looping trees include different steps of musical interaction complexity.

Collaboration between several beginners users and a single advanced user is simple to setup with one screen, as beginners do not need head-tracking to use the *bucket* technique. Collaboration between advanced users, i.e using virtual rays and head-tracking, on the same screen could be implemented with a more complex hardware setup.3D avatars may also be used for distant collaboration, as multiple environments could be synchronized using OpenSoundControl messages together with OpenSG internal synchronization features. Following Birnbaum et al. classification, distribution in space would then be maximized. Another interesting issue is the immersion of the audience. They may obviously be equipped with stereoscopic glasses, but one can imagine various stage setups. Musicians and the audience may both face the same screen, though musicians may occlude parts of the screen. 3D avatars may also be used to display musicians while they interact in front of another screen. Future work will investigate these different issues.

6. REFERENCES

[1] F. Berthaut, M. Hachet, and M. Desainte-Catherine. Piivert: Percussion-based interaction for immersive virtual environments. In *Proceedings of the IEEE Symposium on 3D User Interfaces*, 2010.

[2] D. Birnbaum, R. Fiebrink, J. Malloch, and M. M. Wanderley. Towards a dimension space for musical devices. In *Proceedings of the 2005 International Conference on New Interfaces for Musical Expression (NIME05), Vancouver, BC, Canada*, pages 192–195, 2005.

[3] C. Jacquemin, R. Ajaj, R. Cahen, Y. Ollivier, and D. Schwarz. Plumage: Design d'une interface 3d pour le parcours d'échantillons sonores granularisés. In *Proceedings of the Conférence Francophone sur l'Interaction Homme-Machine(IHM'07)*, 2007.

[4] T. Mäki-Patola, J. Laitinen, A. Kanerva, and T. Takala. Experiments with virtual reality instruments. In *Proceedings of the 2005 International Conference on New Interfaces for Musical Expression (NIME05), Vancouver, BC, Canada*, 2005.

[5] R. Marczak. Etude d'une représentation hiérarchique liant micro et macro-structures musicales. Master's thesis, University of Bordeaux, 2007.

[6] A. G. Mulder. *Design of virtual three-dimensional instruments for sound control*. PhD thesis, Simon Fraser University, Canada, 1998.

[7] J. Olive and S. Pickles. Fijuu: http://www.fijuu.com/.

[8] R. Polfreman. Frameworks 3d: composition in the third dimension. pages 226–229, Pittsburgh, USA, Carnegie Mellon University, 2009.

[9] I. Poupyrev, S. Weghorst, M. Billinghurst, and T. Ichikawa. *Egocentric object manipulation in virtual environments: Empirical evaluation of interaction techniques*. 1998.

[10] X. Rodet, F. Gosselin, P. Mobuchon, J.-P. Lambert, R. Cahen, T. Gaudy, and F. Guedy. Study of haptic and visual interaction for sound and music control in the phase project. In *Proceedings of the 2005 International Conference on New Interfaces for Musical Expression (NIME05), Vancouver, BC, Canada*, 2005.

[11] G. Weinberg, R. Aimi, and K. Jennings. The beatbug network: a rhythmic system for interdependent group collaboration. In *NIME '02: Proceedings of the 2002 conference on New interfaces for musical expression*, pages 1–6, Singapore, 2002. National University of Singapore.

[12] M. Wozniewski, Z. Settel, and J. Cooperstock. A spatial interface for audio and music production. In *Proceedings of the International Conference on Digital Audio Effects (DAFx), 2006*, 2006.