Digital art has become a respected art form, and interest in this very interdisciplinary field is rapidly growing as more and more contemporary artists start employing computers for implementing their own creative work. This course focuses on techniques for creating digital visual art; in particular on the case where custom, not off-the-shelf software tools are applied. The attendees of the course learn how distinct techniques are applied in order to achieve specific artistic goals. They will further learn what consequences and requirements result from given artistic ideas in terms of realizing effective digital art software components.

The course will begin by exemplifying the creative processes of a digital artist, and will explain how creative ideas can be formulated and realized by using traditional offline 3D modeling tools, image-manipulation software, and video-editing techniques. Then, we will give a ground-up perspective on how real-time mixed media processing can be realized in a generic manner, and how these processing capabilities can be integrated into a live digital art performance instrument that actively supports the artistic design flow.

The third part will give a live visuals tutorial. The basic aspects of this type of live performance, such as overall scene setup, dynamic content, automated media retrieval will be introduced and brought into context. The final part focuses on interactive generative systems and on how they are applied in interactive installation and projection environments. It will give an overview on artistic strategies dealing with real-time graphics and sound and describe the relevant software concepts.
Kenneth A. Huff: One Artist’s Work

- How does a contemporary artist formulate and realize creative ideas by using digital techniques?

Kenneth A. Huff is an independent visual artist working in a variety of media, usually involving digital technologies in some form. His body of organically-inspired work spans more than ten years and includes prints, sculptures, mixed-media works, animations, interactive installations, photographs and videos. His work has been recognized with over 120 visual arts awards and grants and been included in over 350 public showings throughout the world, including the last nine SIGGRAPH Art Galleries. Alias Systems named Ken a “Maya Master” in 2002. A frequent contributor to SIGGRAPH, he has given numerous presentations at the conference and was an Art Gallery juror in 2006.
Stefan Müller Arisona is lecturer and post-doctoral researcher at the Computer Systems Institute of ETH Zurich, Switzerland. His main interests are at the intersections of art and technology, and in particular in the domain of live media art. His research focuses on novel real-time multimedia systems and on live multimedia composition and performance techniques. He is co-author of the multimedia authoring software Soundium, which is frequently used for media art installations and live performances by himself and his collaborators. A recent work, the Digital Marionette, is currently installed at the Ars Electronica Center’s permanent exhibition.
Pascal Müller: Live Visuals Tutorial

- What are the standard practices of VJs?

Pascal Mueller is PhD candidate and research assistant at the Computer Vision Lab of the ETH Zurich, Switzerland. His main interests lie in the field of computer graphics: procedural modeling, generative design, animation, visual effects production pipelines and computer-aided media art. He developed the architectural modeling tool CityEngine and is co-developer of the multimedia engine Soundium. He has published various scientific papers including SIGGRAPH, and his body of artistic work includes videos, short movies, over fifty live visuals performances, and several interactive installations exhibited in museums like the Ars Electronica Center. Pascal Mueller received a master degree in computer science from ETH Zurich in 2001. For two years, he worked as a 3D artist and technical director for the Swiss production company Central Pictures.
Bernd Lintermann: Interactive Generative Systems

- How are interactivity and procedural techniques implemented in new media artworks?

Bernd Lintermann works as artist and scientist in the field of real-time computer graphics with a strong focus on interactive and generative systems. The results of his research are applied in the scientific, creative and commercial context. His body of work includes interactive installations, projection environments and stage performances and has been exhibited in museums and festivals all over the world. He has published various scientific papers including SIGGRAPH and is co-founder of the company Greenworks. Currently he is head of the Institute for Visual Media at the ZKM | Center for Art and Media and professor at the State University of Design, Media and Arts in Karlsruhe, Germany.
Intended Audience & Prerequisites

- The course is intended for interdisciplinary working artists and computer graphics scientists, especially those who are involved or interested in the development and use of software tools and digital techniques for the visual arts.

- Basic knowledge of computer graphics principles, audio and video signal processing, and human-computer interaction models is required.

- Level of Difficulty: Intermediate
## Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Session Title</th>
<th>Presenter(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30 - 8:45</td>
<td>Intro</td>
<td>P. Mueller</td>
</tr>
<tr>
<td>8:45 - 9:30</td>
<td>One Artist’s Work</td>
<td>K. Huff</td>
</tr>
<tr>
<td>9:30 - 10:15</td>
<td>Performance Tools</td>
<td>S. Müller Arisona</td>
</tr>
<tr>
<td>10:15 - 10:30</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>10:30 - 11:15</td>
<td>Live Visuals Tutorial</td>
<td>P. Mueller</td>
</tr>
<tr>
<td>11:15 - 12:00</td>
<td>Interactive Generative Systems</td>
<td>B. Lintermann</td>
</tr>
<tr>
<td>12:00 - 12:15</td>
<td>Q &amp; A, Conclusion</td>
<td>All</td>
</tr>
</tbody>
</table>
Part I: One Artist’s Work
Kenneth A. Huff, Independent Artist
The creative process is a journey of discovery
Seeking out the things that inspire me
Finding the solutions to creative problems
Being able to provide the viewer with something new to experience

Overview
- Background (briefly)
- Creative practice and technique
  - Inspiration, themes, fundamental aspects of the work
- Themes in the body of work
- A sampling of work along the way
“Sherman, set the Wayback Machine...”

Background
Creating work more than 10 years; Showing since October 1997
No formal training
Body of work mainly print-based until 2004
2004: Addition of sculptural, time-based works
Photo-based work for over twenty years
1981: Commodore VIC-20 and Commodore pen plotter
Creative practice

Behind the scenes of my creative process
Conceptualization, implementation and presentation
But it all starts with…
Anything can start an idea
Mathematics
Sciences (Biology, Botany, Physics, Chemistry)
Sometimes it is a shadow caught in the corner of my eye can trigger a chain of ideas
But most significantly…
Most inspiration from patterns and forms in nature
Archetypical patterns and forms which appear in a number of manifestations throughout the natural world
To that end…
    I maintain a collection of “natural curiosities”
Over 20 years of reference photography
Most recently, captured audio
Agave plant, a succulent
An example of multiple manifestations of a pattern in the same object
Three manifestations of same pattern
  *thorny edges*
  *impressions* left behind as the leaves grow out from a clustered center
  *shadows* (in this photograph)
Shadows could also be surface color variation
Manifestation of pattern in multiple forms: natural and human-made
Different scales
Skeletonized leaf from a Bo tree: aerial view of a subdivision

“Bo” tree or “Bodhi” tree. *Ficus religiosa*. Buddha is said to have gained enlightenment sitting under a bo tree.
The root/branch of a cypress tree
Ambiguity of scale, orientation and context
I will touch on some of the math and science-based inspiration later…
Moving from inspiration to the concepts and planning behind the work

Very deliberate process

Planning takes the form of sketches and notes
Sometimes the sketches are very specific
This work is meant to be presented on either side of a wall
Sometimes the sketches are bits and pieces that get merged together into the final work.

Over time, I have amassed a large collection of sketches. Mining the sketchbooks — taking ideas from across time and combining them into a new work — has become an important part of my process.
The ideas that are part of every work, the fundamental aspects
This should be the cornerstone of any creative process, the intent
behind the work.

Intent
A work from April 1995
A major aspect of the intent is a purposeful ambiguity
Articulating that ambiguity — The story of a young girl “seeing” a lion
One result of the ambiguity: titles are numerical, not verbal
Inspired by nature, but not recreating nature = ambiguity
   No photographic/scanned elements (in the 3D print work)
Ambiguity: not only subject matter, but also scale
   If physical, would this be large or small?
Structure/pattern/symmetry with variation
   Each of six arms is a unique construction, not a simple duplication
Subtle, subliminal importance
Maintaining illusion of physical reality
Repetition of form with unique variation
Each of 2,000 small objects is unique in its proportions, color and texture
Accomplished with procedural modeling tools I develop
Ambiguity of material — organic/inorganic
Fine, unique details of color and texture
Able to withstand close scrutiny
Allow the viewer to create their own depth of field
Discovery: Each time the piece is viewed, unique details can be found
Important aspect of the print work is detail in image and in print itself. It is one thing to print something large, another to have the image have an appropriate level of detail at that large physical size.

This recent work is 12,000 by 20,000 pixels (60 by 100 inch print) ~230 megapixels (for those with a digital camera around their necks).

Image is ~116x larger than HD television.

HD TV: 1080 x 1920 pixels = 0.864% area of 2004.5
Implementation

Virtual three-dimensional constructions
Sculptural rather than painterly process
    forms, color, texture, lighting, composition
Using software normally used for character animation and special effects
Alias *Maya* and mental images *mentalray*
Adobe *AfterEffects*
Sculpt 3D forms, apply color and texture, light, compose
No photographic or scanned elements (in 3D print works)
Rendering
   At least one month for each print work (single workstation)
Modeling: procedural tools
Oftentimes, rather than sculpting or constructing the forms individually, I create software used to build the forms. Allows me to focus more on the concept behind the work and the overall composition rather than spending thousands of hours creating thousands of forms which may or may not make it into a completed work.

Never purely algorithmic
As an example: *SurfacePlater*

Created using a programming language built into the software environment (MEL)

Tool used to construct numerous small forms that cover and conform to a given surface

*Steps*
1. The surface
2. A plan for the new forms
3. The forms; Sometimes the original surface remains visible in the final piece, sometimes not
The tools automate repetitive tasks
Also include a wide range of controls
The use is very deliberate and directed
The results produced with the tool are not the end
Objects are added or deleted, moved and adjusted
The tools evolve throughout use and with each use

*SurfacePlater* used in the completion of ~30 works
Examples scatter throughout presentation
The flat, rectangular forms were created with *SurfacePlater*
Another example: the woven mesh
The mesh was constructed with a series of smaller tools.
First, woven curves were constructed and used to create lofted surfaces, forming the wires.
The wires were chopped into sections and portions were culled.
The ends of the wires were tapered with a third tool.
Detail
In addition to the fundamental aspects that run through all works, there are themes, groupings of works related on a conceptual level, that weave in and out of the body of work.
Tend to overarch large numbers of works and large periods of time
Multiple themes also intersect or overlap in a given work
Sébastien Truchet, 1657-1729
Monk, hydraulic engineer, inventor
Early 1700s, working on a canal project and saw quarried tiles split diagonally
Two examples from Truchet's catalog (reconstructed)
Someone (???) removed the diagonal color difference and added the quarter circle ornamentation.

From a simple building block to a very complex, aperiodic pattern.
Two Truchet tilings of that base pattern were woven together for this piece.
Truchet curves were used as motion paths in this time–based work.
Three-dimensional Truchet pattern
Knot theory from mathematics
  Piece of string, tangle, glue ends, untangle
  Young branch of mathematics
    early 1900s, faded and recent comeback
      Biology and chemistry: the formation of proteins
      Physics: string theory
      Quantum computing
For me, intriguing symmetries and intertwined forms
Originally conceived as three panels showing the entire knot form
Another knot form
This work is another example of the use of the *SurfacePlater* tool.
Practice: pre-visualization in an important part of my process. 
Pre-visualization = Virtual sketches

Abstract knots are cataloged using a number of different schemes
In this scheme,
  9: The number of times the knot crosses itself
  8: The eighth such knot with 9 crossings
Another theme
Concentric lines patterns of the arches, loops and whorls of fingerprints
Unique for each person and each digit
This theme became the basis for prints and
  first sculptural
  first time-based, animated works
Seen throughout the physical world from geological strata…
...to burled wood...
… to patterns of cultivation…humanity leave its fingerprint on the planet

The photo was taken somewhere south of Minneapolis
Innermost crescent forms were seed elements for the concentric line pattern
The concentric line pattern was constructed using a manual procedural process
Red plates were constructed with SurfacePlater
There were many more layers to the growth pattern than appear in this piece…
…While the innermost crescents disappear, many more layers of the concentric line pattern appear in this work
Same base pattern as 2002.12b (previous)
Rotated 90° Clockwise from 2002.12b
Different implementation
Inspiration: electron micrographs of sintered ceramic powders
Thin undulating forms instead of plates
Based on same original pattern as previous two works
The center portion of the previous works is seen here
Approached by staff at General Motors Design Center in Warren, Michigan in mid-2003 to speak at the center and participate as a guest artist in an exhibition.

Part of arrangement gave me access to their rapid prototyping laboratory, providing me with the technology to turn virtual constructions into physical forms.
Renderings from above of two of the resulting sculptures. The forms were manually “grown” from the innermost, simplest forms
Left: seed elements were innermost closed loops
Right: seed elements from one of the larger elements from the left
If the position/rotation of a seed element was change the entire structure would be affected.
Installation view from a 2005 exhibition at Savannah College of Art and Design with included self-lit platforms. The sculptures are ~6.5 by 20 by 20 inches.
RP process — stereolithography
Photosensitive liquid polymer resin
Each layer is 0.05mm (1/20th of millimeter)
   31 layers to make thickness of US penny (1.55mm)
Selection of this particular rapid prototyping process was based on a desire to work with a translucent/transparent material. Interestingly, the sculptures continue to evolve as the ultraviolet sensitivity is inherent in the material.
A scanning electron micrograph of top edge detail of one of the forms (scanned during a residency at Juniata College, Huntingdon, Pennsylvania in 2005)
I embraced and exploited the physical characteristics of the process, using the layering process of stereolithography to manifest a concentric line pattern along the top edges of the forms.
After ten years of using animation software to create still images, I started working with time as an element in my work. Important to maintain contemplative and exploratory nature of the work. Mechanisms of perception further exploited.
2004.9a — First 33 seconds of a 20-minute loop
First and last frames of the 33-second excerpt, showing the very slow shift in color and position.
Every 1,000th frame of the 20-minute animation, showing the one major cycle of color and the overall slow rotation of the entire structure.
The second work in the series.
While the forms are moving up and down and the colors are shifting rather quickly, the rotation of the overall structure is very slow. Inspiration for the series came from the time scale visible at a seashore. The rapid breaking waves, the ebb and flow of the tides, the seasonal changes and the changes occurring on geological time scales. This piece goes through six major color cycles.
In this final work from the series, the concentric line pattern is implemented in two counter-rotating layers, with a third, more subtle layer of the pattern manifest in the shadows cast by the other two layers.
A time-based work from 2005, 40-minute, high definition, seamless loop
An interesting shift in the implementation of my work came about with this piece in the removal of surface relief from the forms. The detail of texture destroyed the beauty of the piece.
At the same time, the work makes a full circle of the intent of all of my work — mindful active engagement. In this case, anticipation of motion brings the engagement.
A time-based work from 2006, 40-minute, high definition, seamless loop

What’s next for my work
  - Carrying ideas forward into interactive installations
  - Physical mixed media for print based works
  - Incorporation of sound elements
  - Continuation of time-based works
  - Continuing my explorations, my journey of discovery
Acknowledgments

Alias Systems
mental images
Savannah College of Art and Design
School of Film and Digital Media
General Motors Design Center
Acknowledgments

Autumn Color
Apple Computer
Sean Rush
Yvonne Barlog and Mike Barlog
Many current digital art works operate in multiple media domains, for example computer graphics, video processing, audio signal processing, or arbitrary sensory input. Often, analysis and synthesis are cross-combined, such as applying real-time audio and music analysis results for computer graphics generation. In addition, digital art performances are typically interactive, and the computer is used as a performance instrument. From a human-centered viewpoint, and as in traditional music performance, this instrument must be controllable in a expressive and virtuous manner.

This course session gives an overview of live performance tools with a focus on those that operate in the visual domain. In particular, we address computer scientists involved in the construction of performance tools and artists with an interest in the underlying technology they are using. The fundamental question is how we can successfully design tools that satisfy the artist’s needs. We will start with general observations of tools with respect to their purpose and goals, and we will look at the typical performance artist’s workflow. Part 2 will give a classification and an overview of existing visual performance instruments. Part 3 and 4 will deal with two specific issues (real-time mixed media processing and design-oriented composition). The session concludes with example works, both in terms of a existing live composition and performance platform, and in terms of art works realised with the platform.

The title image shows an example live visuals performance (2002). Free jazz pianist Guerino Mazzola was attached to EEG sensors, which were sampled in real-time. The data stream was used by the visual artist (not on photo) in order to interactively compose live visuals.
How can we successfully create tools that cover the artist’s needs and requirements?

- General Observations
- Existing tools for live visual performance
- Two specific issues in detail:
  - Mixed-media processing
  - Design oriented composition
- Example works
The tools that evolved over the past years are the result of an intense interdisciplinary work of computer scientists, software engineers, and artists. Notably, these collaborations often resulted in computer scientists and engineers getting seriously involved in the process of creating artworks, and vice versa, artists starting to strengthen their technological skills and taking an active part during the course of software design and implementation.

The realisation of a concrete software tool was typically launched by a given artistic idea or problem. However, exploring and using the final result often resulted in novel artistic ideas that went beyond the tools capabilities and therefore induced another development phase. Besides of being very exciting, this cycle was the main drive for collecting common principles and practices, and to implement them as part of general-purpose tools.
Part 1

General Observations
Performance tools purposes and goals

- Assist the artist in an intuitive and expressive manner
- Attempt to address the whole artistic workflow, from an artistic idea / goal towards the actual performance
- Be able to adapt to artist-chosen variations of the workflow
- Provide high-quality output
- Be highly reliable

As the primary purpose, a tool should actively assist the working artist. By intuitive and expressive, and with a reference to music, we imply the use of a tool as an instrument designed for performance. As in music, skills for using the instrument effectively need to be learned, and ultimately mastering the instrument allows for virtuous performance.

In order to systematically collect requirements demanded by the artist, it is important to understand the artistic workflow, either generally (in the case of building general-purpose tools), or for a specific case (in the case of a custom built tool). Structuring the workflow allows to define the primary properties and requirements for the concrete software and interaction design. However, we should keep in mind, that a workflow is typically not absolutely fixed (a simple example is when the person using the tool changes), and therefore a tool should be able to adapt to such variations.

Besides of these assistive properties, computational tasks play an important factor of performance tools: A tool should always attempt to provide high-quality output, but depending on the complexity, there must be a trade-off between quality and being able to produce results interactively and in real-time. In addition, high reliability is essential, as not only crashes during performance are fatal, but already missing or erroneous details will be noticed by the audience.
Clearly, even the best tool can not support every artistic idea and workflow. Most important, the artist needs to be aware that she is working with a software tool, and that her workflow need to be made accessible to the computer. This implies that the workflow needs to be reasonably structured. In addition, and as with a music instrument, the artist should know how the tool works, how it should be used and how it can be broken.

While a tool can provide for a great output quality (e.g., high resolution and frame rate, colour quality, etc.) it can generally not guarantee for quality of the artwork itself. After all, its purpose is to assist the artist, and not to domineer over the artist.

Finally, software engineers must be aware that “their” tool will most certainly be applied in a manner that it was not thought for in the first place. However, and as stated earlier, this fact can be seen as a positive effect to drive the evolution of both artistic expression as well as a tool’s capabilities.
Devising a suitable workflow

- Can an existing model serve as a basis?
- Yes: Music composition and performance
  - Very elaborate, evolved over centuries
- Typically: Separation between composition and performance (but not strictly, e.g., improvisation)
- The musical score serves as a mediator between composition and performance

In order to devise a suitable structured workflow, we may have a closer look at the process of music composition and performance. Here, the workflow is very elaborate as it has evolved over centuries, at the same time it has always been very flexible to adapt to different periods.

One characteristic has always been part to the musical workflow: the separation of composition and performance (even this one not always rigid, for instance in terms of improvisation). This separation has allowed composers to work in a purely mental space and to apply rules or systematic methods (e.g., counterpoint or Xenakis). In contrast the performer attempts to retrieve the original compositional idea (by studying the score, using his musicological knowledge, and through rehearsal), and to express it accordingly during performance. Not surprisingly, compositional models were adapted to other art forms in the 20th century (Kandinsky, Eisenstein).

References

Deriving from music composition and performance, we can devise a typical performance workflow: Starting with an artistic idea or concept, which is typically not expressed in a formal manner, the formulation of design goals results in a more precise description of the original idea. As we shall see later, it is possible for a tool to provide support for dealing with design goals. In a next step, material that will support the design goals is collected. From here, experimentation often plays an important role in order to determine whether something works or not. After that, systematic composition tasks realise the original artistic idea. The resulting composition is available for performance.

Rehearsal typically has two purposes: first, the original composition should be approximated. Second, skills required for the performance need to be trained. The actual performance expresses the composition. It transforms the original idea from a mental to a physical space.

Usually, the dominance of individual topics in the workflow varies and there may be multiple iterations between tasks involved. In addition, the overlap between composition and performance is becoming increasingly important, and we will show later, how compositional tasks can be applied in the performance domain.
Observations (1)

Typically, there are a number of tools involved

- Techniques not only for the performance itself are required, but also for:
  - Organisation of ideas
  - Organisation of materials (e.g., media content)
  - Authoring / composition / design
  - Experimentation and rehearsal

- Resulting in a complete “Toolchain”
Observations (2)

**Representation is important**

- As in music notation, a suitable representation helps to glue between different aspects of the workflow
- In contrast to music notation, we are free to decide the detail of the representation
- The representation allows to employ computer-assisted composition and performance methods

Exercise from Carl Czerny’s *Vollständige theoretisch-praktische Pianoforte Schule, Op. 500*
Part 2

Example Tools for Live Visual Performance
The diagram classifies tools for live visual performance in terms of programmability and complexity.

On one end, the “custom” side, tools with a fixed functionality are found. Typically, these tools address specific needs for the realisation of a specific artistic idea, or they realise historic pre-computer counterparts, such as analogue video mixing consoles.

The intermediate class provides more flexibility by allowing for interactive composition and extensibility through scripting languages. Example tools are those that share original video mixing ideas, but employ the computer’s power to reconfigure the system according to the artist’s needs.

The last class refers to tools that constitute complete programming environments. Here, often the “graphical programming” paradigm is applied, in order to provide extensive freedom for the realisation of works. In that sense, custom “tools” can be created on top of these environments, closing the loop to custom tools, but typically with some compromises regarding computing performance.

It is important to note that the borders between the given classes are generally fluent, as for example most tools provide means for extensibility. In addition, programmable tools are not “better” by default. For instance, for a given problem, sometimes a small custom tool can provide a solution more effectively.
Classes of visual performance tools

- Digital Video Mixers
  - Root in analog video mixing and effects
  - Fixed processing engine and GUI

- Visual Synthesisers
  - Interactive composition of processing flows and visual sources / effects
  - Customisable GUI

- Graphical Programming Environments
  - Allow for creation of programs, which represent the composition
  - Often also allow for general-purpose computing

The following slides will illustrate a few sample tools belonging into each category. We selected representative example and the list is by no means exhaustive.

A good source for further information on live visuals tools, in particular with respect to VJing, is:

http://www.vjcentral.com/
ArKaos is a tool designed for VJ performance. Its design is relatively static, and it focuses on video mixing / processing. Compositions are basically modelled via looped video clips and selected effects that stored as patches, which can be activated via computer keyboard or MIDI input.

In contrast to a traditional video mixer, the software provides real-time audio analysis in order to match clip loops and other automation to musical features.

http://www.arkaos.net/
Similarly to ArKaos, Vdmx focuses on real-time video performance, but allows for more complex setup. The video processing paths as well as the user interface can be flexibly configured, and loop and effect parameters can be automated using animation curves. At the same time, the skills required for controlling the program are higher.

http://www.vidvox.net/
Touch is a live visuals applications with a strong background in computer graphics, and as such, it is not limited to video processing. The basic concept are visual synthesisers, similarly to electronic music synthesisers. Touch Designer (in contrast to the feature-limited Touch Player tool) is an authoring environment that allows for creation and editing of synthesisers, and provides complete 3D animation and video compositing features. In addition to content, the designer allows for creation of customised GUI control panels.

http://www.derivativeinc.com/
Max/MSP member of a family of graphical programming environment that has its origins in sound and music computing. Graphics processing is enabled via a an additional framework called Jitter. The basic concept behind Max are patches, which contain graphical programs. In contrast to graphical composition of building blocks, as it is known from many other software tools, the patches may contain program segments down to an “atomic” level, e.g., such as the addition of two numbers. Max has become the de facto standard for many media artists, but we shall also mention that the graphical programming paradigm is not indisputable among computer scientists as well as users.

http://www.cycling74.com/

References

The following two parts of this session focus on two issues that invariant among the given classification scheme: In part 3, we will have a closer look at real-time multimedia processing issues and how the processing logic can be de-coupled from the application logic. The basic concepts presented there will also serve as the basis for part 4, where we will highlight our approach to design-oriented composition. In addition, a third issue, media management, is becoming increasingly important as media libraries used for performance are rapidly growing. This issues will be dealt with in detail in session 3 of the course.
Part 3

Real-time
Mixed Media Processing
Realising tools from a multimedia processing viewpoint

- Reliable real-time multimedia processing is a precondition for successful live performance
- Not just graphics, but also music, audio, network, sensors, etc. - Typically loosely coupled
- How can we provide adequate processing facilities that satisfy the artist’s needs?
- In particular:
  - How do we deal with media coupling?
  - How do we address live performance issues?
  - Can we provide access to the system at different levels?

Today, the problem of real-time processing in complex multimedia systems has been solved in many domains, and existing research and software sources provide a good basis for the implementation of novel tools. However, many existing frameworks (e.g., scene graphs for graphics, audio processing libraries, etc.) focus on a dedicated media type, or are dedicated to operate in a specific application domain (e.g., a game engine). Notably, generic approaches to link parameters between different media types are relatively rare.

Here, we will highlight some observations that arose from implementing digital art applications which required real-time processing of multiple media types by default. The reference below deals with the problem in detail.

References

Many media processing applications are realised in a monolithic way: The system is a specific solution for a given problem. However, as new types of work are created, this leads to a cumbersome iteration: Essential parts of the system need to be re-implemented over and over. That step too often implies valuable time and cost, and, more importantly, interrupts the creative flow.
How can we simplify implementation in terms of the processing part?

- Insert an additional layer for processing logic
- Provide a unified view of the processing state to the application
- Reuse existing processing libraries and frameworks
- What should this layer look like?

We separate the GUI and application logic from the processing logic, with the goal to implement the latter in a generic manner, available for later reuse. To the layer above, the processing layer provides a unified view to applications. To the layer below, it allows for reuse of existing processing libraries and frameworks.
Typical abstraction model: Processing graphs

- The model widely employed in many existing tools and frameworks
- Formally founded
- Variations depending on purpose
  - CG Scene graphs
  - Audio flow graphs

Graphs have been applied in many domains of media processing, e.g., in the form of scene graphs for computer graphics, or as data flow graphs for audio and video processing. In addition, graphs are a well-known construct for formal approaches, e.g. for optimisation transformations. For these reasons, we propose to unify existing graph-based frameworks in one global processing graph, which is then presented as a single view to the application layer.

A closer look at the graph reveals processing nodes with ports, which can be interconnected. Depending on the situation, connections may be untyped or strictly typed. Examples for the latter case are simple values such as numbers or strings, but also vectors, or complete audio or video streams. In addition, connections may also represent parent-child relationships, as they appear in graphics scene graphs. In that case, the connection denotes an underlying state mechanism, and no data is directly transferred.

References

The Decklight media processing framework

- Underlying frameworks and libraries are unified in a *global* processing graph
- The graph’s structure is presented to application
- Graph can be modified consistently and at any time through transactions
  - Essential for the application of compositional tasks during performance
- The graph’s nodes represent underlying processing entities
  - Easy integration of existing libraries and frameworks

Graph modification may take place during operation, which implies the need for consistent graph updates. Examples are modifying multiple transformation parameters in a scene graph, or multiple filter parameters in an audio graph. In both examples, changes must be applied at once at a frame boundary, otherwise undesired jitter effects will result. Therefore, graph modifications are collected in transactions, which can be executed atomically for instance at frame boundaries.

There is a problem we have not dealt with: As a result from unifying multiple graph-based processing entities in a single graph, we have to deal with different graph processing semantics: For example a audio graph, which is typically flow-based) is processed in a different manner than a scene graph (which is basically an object hierarchy). The question is, how do we deal with the coexistence of different semantics in the same graph?

Our approach is to segment the global graph into individual subgraphs, which correspond to different semantics. Of course, this segmentation will not be made visible to the application layer. For each graph type, the segmentation process starts at dedicated root nodes, e.g. at audio sinks for pull-based audio flow graph, or at the viewport for a scene graph. From there, well-known graph walking algorithms are applied to complete the segmentation.

The graph segmentation not only allows us to deal with multiple processing semantics, but facilitates the parallelisation of the whole system, as each subgraph may run in an individual threads. This concept of “globally asynchronous, locally synchronous” systems is for example well-established in hardware design (“GALS” systems). Of course, there is a need to correctly deal with data flows at asynchronous borders. Here we can apply automatic methods such as averaging or maximising in order to maintain data consistency.
The figure gives an overview of the Decklight media processing framework. It indicates the interfaces to the application layer (basically the inquiry of the node repository and the transaction interface for modifying the processing graph), and how underlying frameworks are integrated in terms of concrete processing nodes.
Media processing summary

A generic framework for real-time mixed-media processing enables:

- A unified and consistent representation of media processing entities
- Easy integration of existing multimedia processing libraries and frameworks
- Provides access at different levels of expertise:
  - low-level scheduling and processing
  - processing node implementation
  - application development

This part presented a software component that allows us to integrate existing multimedia processing libraries and frameworks, and present a unified and consistent view to applications. The processing layer allows for automated parallelisation and independently copes with data consistency issues.

From a software engineering viewpoint, the layer also provides access do different levels of expertise, e.g. media processing experts can directly interact with low-level scheduling and processing issues, integrate their own frameworks, and fine-tune the system for their needs. These results become automatically available to application developers, who may just reuse integrated frameworks.
Part 4

Design-oriented Composition
Composition

- We present a technique that augments multimedia composition techniques in general
- In particular, the technique supports the application of compositional tasks during performance
  - As shown previously, solved for the processing part
  - Here, we deal with interaction techniques
- What do we mean by *composition*?
  - Very general: the process of creating artwork

In the previous part, we have shown strategies to deal with real-time processing of different media types: A global graph incorporates various processing entities. Here, we introduce an interaction technique that supports the composition digital art works, in particular in a live context, where composition during performance takes place.

While there are many different viewpoints on composition, here, we use the term in a very general way, and refer it to any activity that creates or modifies artwork.
Live composition

- “Composing” + “Live” = Something that happens during performance
- Often, there is a gap between composition and performance:
  - Can we edit a movie while it is playing?
  - Can we model a scene while its animation is running?
  - Can we compose music while performing?
- The gap as seen from a software viewpoint:
  Separation of composition vs. performance
  - Separate tools, or
  - Inside tools, “edit” and “perform” modes

When attempting to apply composition methods during live performance, it shows that many tools separate between composition and performance. Often, compositional elements are applicable only in a limited way: Structural changes to the composition (e.g., modifying a scene graph or a flow graph) only available in “edit” mode, or even more radically, the composition tool is completely separated from the performance tool.

Here, we follow the approach of not differentiating between composition and performance, i.e., the two areas earlier shown in the workflow overlap to a large amount. A generic processing layer as shown in the previous part allows us to apply composition tasks without interrupting the performance. In the extreme case, we may completely re-configure the composition as a whole.
Compositional tasks from a software (or tool) perspective

- Allow for creation and transformation of artwork
- Collect and organise artwork
- Provide means for navigation in and exploration of a collection of artworks
- In order to enable live composition:
  - Organisation and exploration become very important
  - We need effective interaction models
  - We need to embed composition in the performance process

With the general definition of composition we gave above, the task of “creating” artwork is what tools generally provide (obviously with wide variations in the actual implementations). By “transformation” we refer to modification of material, this may occur in an interactive manner or through computational methods (e.g., based on rules).

The ability to collect and organise artwork is becoming increasingly important in a live context: using a standard file browser during a performance already becomes cumbersome (e.g., with respect to the time required for finding something). When artwork is organised, we can easily provide means for navigation and exploration.
The “design tree”

- The design tree is a design editing software component
- It collects and organises the artist’s work
- Design operations provide high-level design editing methods
- Exploration, combination, reuse, and evolution of designs...
- ...before and during performance

Our approach to composition, either in the traditional sense (i.e., composition as an separate task before performance takes place) or for live composition is called the “design tree”. We use the term “design” both for denoting to individual compositions as well as in its verbal form to refer to the activity of composing.

Besides of collecting and organising the artist’s work, the design tree provides a data structure with is available to interactive and computational methods.

References

Deducing the design tree

- Have a closer look at how a well-established application concept evolved
- **Undo / redo!**
  - First, there was undo only
  - Then, there was redo
  - And then, multi-level undo / redo

In its simplest form, the design tree can be deduced from the standard multi-level undo/redo mechanisms as we know it in most of today’s applications. A sequence of user actions can be split into individual versions, and the application of undo or redo allows to move the current version among the sequence.
The linear sequence of user actions is made explicit in Photoshop’s “History”: It gives a detailed view of previous actions, as well as those that are disabled due to the undo function. In addition, the history allows the creation of snapshots of the current version.
The fundamental limitation of multi-level undo / redo

- Every operation that changes a design invalidates the redo path
- What takes us around this limitation?
  - Do not invalidate redo path, instead keep it and branch to create a new design variant
- A tree structure emerges
- Automated versioning
  - What we sometimes do manually or with the help of versioning tools

Obviously, the fundamental limitation of standard undo / redo is that every change to the current version invalidates the redo path, and previously existing versions (those that have be subject to repeated undos) become inaccessible. A simple mechanism takes us around this limitation: instead of invalidation of previous versions, we branch at the current version in order to reflect the change. As a consequence, a tree structure of “design versions” emerges: the design tree. One important property of the design tree is its automatic version capability: changes to the current design (design variants) result in new designs, keeping the original design available for later reference or reuse. In addition it organises a collection of designs by keeping the evolving design path and therefore putting the designs into relation.
Underlying Representation

- We need an underlying representation for the description of the design tree’s individual nodes
- In our case, we employ a simple processing graph description language
- The design tree’s nodes encapsulate differential changes to the graph

```javascript
viewport = gl.viewport;
quad = gl.quad;
viewport.child = quad.parent;
quad.size = "100x100";
quad.size = "200x100";
```

So far, we have not dealt with the actual content of individual design nodes. Here the generalisation of the processing layer from the previous part comes into play: Every design represents a corresponding processing graph. For a more compact realisation, individual design nodes only store differential changes to the graph, as it is indicated in the example figure. The concrete representation inside each design node is a simple graph description language, which allows for the creation and destruction of graph nodes, for creation and deletion of graph edges (“connections”), and for the assignment of properties (e.g., values) to the graphs nodes.

Individual designs may emerge through GUI-based generation, or through manual scripting, or through generative methods.
The following slides will give an overview of available design tree operations. Besides of the basic ones, such as creating new nodes based on changes to the processing graph, and activation of a current node, we will also show, that higher-level design operations, such as design merging and inserting can be realised on top of the design tree.
Initial design

The initial design consists of a Camera and a Viewport processing node. Nothing visible yet.

The frame on the left shows the design tree as it emerges, the centre frame shows the current processing graph, and the frame on the right shows the “output” resulting from the current processing graph.
Adding a design

A quad is attached to the Viewport. Commiting the changes results in a new design node named Quad.
Modifying a design

Two new processing nodes (Transform and Color) are inserted into the processing graph. Committing the changes results in a design node CQuad.
By activating Quad again, the previous changes are undone. At the same time, CQuad remains available for future use.
The artist decides to deactivate the design Quad and create a new design CDisk. A branch is automatically created.
The design node CQuad is extracted and merged with CDisk, resulting in a design +CQuad. The output now consists of CQuad and CDisk. Name conflicts (Color to Color1) are automatically resolved.
The designs CDisk and +CQuad are fused to a new design CDiskQuad. The processing graph remains unchanged.
The splitting of design nodes requires user intervention as we need information which part of the change set remains in the original node, and which part is split off.
The Disk node of CDiskQuad is replaced with extracted Quad and Transform nodes of TQuad, resulting in a design mix *CQuads.

Requires user intervention.

again, user intervention is required here in order to determine the extracted processing nodes to be applied to the mixed design.
Design transformation example. The design on the left is transformed into the one on the right by applying design operations. Two possible alternatives are illustrated.
The design tree in practice

- **Organisation**
  - Facilitates preparation of a performance
  - Structures a collection of designs; sharing of common properties

- **Exploration**
  - Designs are easily activated
  - Allows for navigation in an unknown design space
  - Design insertion allows for previewing during performance

- **Transformation**
  - Designs are created and transformed in order to reach an artistic goal
  - Designs may be reused and adapted as needed
Ongoing work

- The underlying representation must provide appropriate semantics
  - Design tree operations can become more powerful
- A highly effective user interface is required
Part 5

Example Works
Soundium is a multimedia composition and performance platform that implements the concepts highlighted in parts 3 and 4. By employing a generalised processing model, Soundium deals with arbitrary media types in a uniform manner and allows for easy integration of existing media processing libraries and frameworks. In addition, it does not distinguish between composition and performance, the basic idea is to allow for “do anything, anytime”.

The Soundium performance platform

- A multimedia composition and performance platform
- Deals with any media type in a uniform manner
- Evolved as a research platform
  - Aims at the integration of new models for multimedia processing and HCI
  - Rapid realisation of digital art and entertainment applications
- Implements the processing model (part 3) and the design tree (part 4)
Soundium central interaction environment is its graphical management console, which provides interactive access to the design tree, the media manager, and the processing graph. Media processing is performed in a separate engine, which can be controlled through a networked interface (graph changes and inquiry are invoked using a remote procedure calls).
The figure gives an overview of the basic control views inside the graphical management console: The design tree, the processing graph, and the node inspector for specific nodes in the processing graph. Not shown in the figure is Soundium’s media manager.
The interactive installation Digital Marionette, currently exhibited at the Ars Electronica Center in Linz (Austria), shows the audience the look and feel of a puppet in the multimedia era: The nicely dressed wooden marionette is replaced by a Lara Croft - like character; the traditional strings attached to puppet control handles emerge into a network of computer cables. The installation consists of a projection of a digital face, which can be controlled by the visitors. The puppet can be made talking via speech input, and the classical puppet controls serve as controllers for head direction and face emotions, such as joy, anger, or sadness. The whole artistic concept was designed and realised in an interdisciplinary manner, incorporating art historical facts about marionettes, the architectural space, interaction design, and state of the art research results from computer graphics and speech recognition. Speech input is realised via speech recognition, where the recognised phonemes are mapped to a set of facial expressions and visemes. The different media components have been implemented and integrated in Soundium.

References

Live visuals / VJing

Soundium has been intensively applied for live visuals performances in club scenes and at digital art festivals. Details of how concrete live visuals performances have been realised will be given in the following session of the course.
Part III: Live Visuals Tutorial
Pascal Mueller, ETH Zurich

In this part of the course, we give a tutorial on how to compose live visuals. First, we talk about hardware and software setup needed for a live visual performance. Then we present concepts to prepare and organize huge media libraries allowing for instant access during performance. Afterwards, we give an overview of the basic content elements and how they can be edited, timed or animated according to musical properties (provided by real-time audio analysis). Finally we explain the workflow during a live visuals performance and show how graphic designs can be successfully embedded in a real-world environments. During the talk, we will make several live demonstrations by using the Soundium system as case in point - but most concepts can be applied to other performance tools as well.

The background image shows a typical live visuals setup.
Outline

- Background
- The Soundium System
- Media Management
- Dynamic 2D Content
- Dynamic 3D Content
- Performance Workflow
- Conclusions
The term “VJing”, referring to the VJ who composes visual art at music concerts, electronic dance music events, public spaces or media art performances in general, is nowadays becoming increasingly popular. The expression VJ originally referred to Video Jockey (as a counterpart to the DJ), but its more accurately Visuals Jockey. Nowadays, VJs have more in common with musicians than with DJs.

A (very) brief history of VJing:

- Became popular in the 60s (Pink Floyd, Andy Warhol’s Exploding Plastic Inevitable shows with Velvet Underground, Nam June Paik, etc).
- Dedicated hardware systems (analog video synthesizers and mixing consoles) became affordable in the 80s and 90s.
- Today’s VJs use multimedia PCs or have a wide choice of off the shelf hardware products, covering every aspect of visual performance, including video sample playback, real-time video effects, scratchable DVD players and 3D visual generation.

Further reading:

Definition: *Real-time graphics composed live by a performing visual artist*

Typical characteristics:
- Includes Flash-like 2D and state-of-the-art 3D graphics i.e. more than just mixing and manipulating video or film
- Audio-driven dynamics
- Improvised non-scripted performances
- Mostly generated with personal computers

The snapshot sequence represents a 5 minute excerpt from a typical live performance consisting of abstract imagery, video loops, font layouts and a live video stream.
Soundium has been developed mainly at ETH Zurich since 2001 by the members of the art collective Corebounce (S. Müller Arisona, S. Schubiger-Banz, M. Specht and P. Müller). The tool has been extensively tested in live performances by its developers and collaborators. More information about the software can be found here: http://www.corebounce.org
The Soundium System

Hardware Requirements

- Minimal setup: Notebook with fast 3D graphics
- Typical setup:
  - Notebook with GUI (client)
  - PC(s) with fast 3D graphics (render engines)
  - DualHead2Go: 4 instead of 2 projectors with one PC
    e.g. one OpenGL viewport of 4K resolution on 4 projectors side-by-side
  - Preview displays and MIDI input devices

Links:
The Soundium System
Graphical User Interface

Manipulation on all abstraction levels in preparation phase and during performance:

- Interactive design editing (high-level graph operations)
- Interactive processing graph manipulation
- Interactive processing node parameterization
- Interactive multi-value changes (value sets)
- Runtime Scripting

The screenshot shows the GUI of Soundium. Top left: Design tree which permits high-level editing. Bottom left: Current system state of the processing graph consisting of (1) OpenGL nodes like transformations, distributions, images, primitives etc, (2) audio nodes like equalizers, beat detection etc, (3) shader nodes like analog video effects etc, and (3) generic nodes like curves or video editing nodes etc. Right: Node inspector for modification of a selected processing node (curve editor in this example).
The core, our generic live visuals setup, evolved over 5 years and is the result of more than 100 performances.

It comprises of an overall scene lighting, such as a traditional three-light setup, of global transformations (e.g., camera movement, or audio-sensitive dynamics), of real-time NLE nodes, and of full-scene post effects, such as motion blur, glow, or masking. The setup is placed in the root node of the design tree, or in nodes close to the root. Specific or novel designs are usually built upon the core. Consequently, every design deeper in the tree’s structure will inherit the particular global configuration.
A typical media library of a live visuals artist contains several gigabytes of pictures, video footage, geometric models, and so on. Hence, an efficient media manager is needed to allow for quick interactive access, which is crucial during live performance. In SOUNDIUM, the media manager extracts metadata in a “pre-production” phase by automatically scanning media files according to their type. In addition, the media manager supports manual annotation of media files (e.g., for semantic content descriptions which cannot be acquired automatically). The resulting metadata is stored in XML format based on the MPEG-7 description standard [Manjunath et al. 2002]. The digital library with its incorporated metadata can be accessed by using fast and intuitive high-level retrieval mechanisms.
Media Management
Organization and Annotation

- Management of multimedia metadata via MPEG 7
- MPEG 7 provides a wide range of functionality to generate and manage metadata of a broad range of different media types. It consists of the following parts:
  - MPEG 7 Systems - Basic tools for transport and storage
  - MPEG 7 Description Definition Language (DDL)
  - MPEG 7 Visual - Description Tools dealing with (only) Visual descriptions
  - MPEG 7 Multimedia Description Schemes - Description Tools dealing with generic features and multimedia descriptions
  - MPEG 7 Audio - Description Tools dealing with (only) Audio descriptions
  - ....

MPEG 7 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group), the committee that also developed the successful standards known as MPEG-1 (1992) and MPEG-2 (1994), and the MPEG 4 standard (Version 1 in 1998, and version 2 in 1999). The MPEG 7 standard, formally named "Multimedia Content Description Interface", provides a rich set of standardized tools to describe multimedia content. Both human users and automatic systems that process audiovisual information are within the scope of MPEG 7. MPEG 7 offers a comprehensive set of audiovisual description tools (the metadata elements and their structure and relationships, that are defined by the standard in the form of descriptors and Description Schemes) to create descriptions (i.e. a set of instantiated Description Schemes and their corresponding Descriptors at the users will), which will form the basis for applications enabling the needed effective and efficient access (search, filtering and browsing) to multimedia content. MPEG 7 has been developed by experts representing broadcasters, electronics manufacturers, content creators and managers, publishers, intellectual property rights managers, telecommunication service providers and academia.
Since most live visual performances include video footage, the video clip is the most important media type a visual artist is working with. Traditionally, footage is prepared manually using video editing tools, which is a very time consuming task. Hence, the media manager analyzes footage using video segmentation techniques for shot boundary detection and video abstracting techniques for scene determination. The implemented methods base on:


As a result, the original shot list of can be reverse engineered which lead to a structured organization of the video clip into scenes, groups and shots. Definitions:

- Keyframe: Representative of video shot (multiple keyframes per shot possible). Important for browsing.
- Shot: Delimited by shot boundaries (hardcuts, fades or dissolves)
- Group: Visually similar video shots (shot may belong to multiple groups if content changing strongly over time)
- Scene: Collection of semantically related groups

MPEG-7 consist of video-related structures and descriptors that cover the following basic visual features: color, texture, shape, motion, etc. Each category consists of elementary and sophisticated descriptors. More information:

Media Management

Movie File Formats

- Uncompressed, lossless or almost lossless formats:
  - Uncompressed Avi/Quicktime, Huffyuv (fast!), DV, …
  - Fast access, but requires to much storage space

- Formats with high compression factor:
  - MPEG 4, DivX, H264, …
  - Non-linear access in such compressed video files critical due to keyframe encoding i.e. real-time random access in highly compressed video (PAL or NTSC resolution) is almost impossible on today’s standard PCs
  - Decoding strategies like caching or pre-fetching required

Some formats/codecs like DV (Digital Video) or Huffyuv contain a keyframes every frame. Hence, random access as required by NLE is possible, but the compression factor is tremendously smaller. For example, a H264 movie of 3 minutes (PAL resolution) has the size of 41 MB. If we transcode this movie to DV, it results in a size of 1 GB (in H264 we have a compression factor of 220, in DV format only a compression factor of 9).
Dynamic 2D Content

- Imagery
- Video footage
  - Clip (‘stream’, w/wo shot list)
  - Prepared loops
- Parameterized 2D effects
  - Standard effects like color-correction, glow, etc
  - Customized effects like twirl, glass-breaking, etc
- 2D vector graphics (including fonts) - addressed in next section

How to animate it in a live real-time environment?
Keyframing too expensive, we need procedural techniques…
Dynamic 2D Content
Moving Images

- Map audio levels on transformations
  - E.g. smoothed bass level on z-translation
  - Difficult to do in real-time and to retain control

- Predefined animation curves
  - Pan, zoom and fade (à la screen saver slideshows)
  - Can be triggered by audio

- Procedural imagery nodes
  - User can assign set of images to e.g. slideshow node
  - User controls parameters of the customized nodes e.g. images per minutes
In film music theory, a visual action point is usually associated with a classical film cut, but it can also be within a continuous shot or refer to arbitrary types of dramatic events. Most film music has been conducted according to such visual action points.

Further reading:

To make computer-aided non-linear editing (NLE) possible in real-time, Soundium includes dedicated interactive NLE processing nodes which inverse the film music approach by generating visual action points to given audio. We implemented different editing techniques according to Eisenstein:

These nodes decide which shots are played and align and time transform shots according to music characteristics e.g. extracted beat borders may enforce cuts. Furthermore, these nodes automatically rearrange selected shots or whole (annotated) video clips in real-time. For instance, the user can assign a whole video scene or multiple groups to the NLE node, and then tune editing parameters such as 'cuts per second'. Of course also completely audio-independent editing is supported.

The figure illustrates such an interactive real-time video editing example. Different shots of two scenes are aligned to different music parts using similarity analysis. The individual shots are non-linearly stretched to beat intervals.
Dynamic 2D Content

Animated Shader Effects

- Usually applied in a ‘post-processing’ step (per frame)
- Photoshop-like effects controlled via parameters (implemented as fragment shader nodes using GLSL)
- Dynamic behavior via parameters, e.g.:
  - Control continuous values via audio level or animation curve mapping
  - Discrete values via (modulated) beat or audio patterns
The inclusion of 3D content is a mixed blessing: On one side, almost everything is possible concerning rendering performance of the next-generation platforms, on the other side, almost everything is possible concerning compositing of content i.e. computer-aided, procedural or high-level methods are crucial to be able to design such content in real-time.

In this section of the tutorial, we will specify the most general 3D content elements, give examples how they can be animated, and we present concepts to organize and control the graphical design of highly emergent 3D scenarios.
Dynamic 3D Content

Abstract and Objective Content

- **Abstract**
  - + Can be efficiently modeled and parameterized, parts are reusable
  - - Difficult to animate attractive

- **Objective**
  - + Interactive 3D or virtual worlds impressive, ‘unlimited’ possibilities
  - - Lighting and rendering difficult, tends to look 80s VR style
Audiovisual associations:

- **Continuous values (driven by audio levels):**
  - Non-linear mapping to transformations
  - Speed, acceleration, intensities, densities, etc…

- **Triggers (driven by beat detection):**
  - *RotFX* (rotation):
    - stop, change direction, change axis
  - *Swing* (sinus rotation):
    - stop, change direction
  - *TransX* (sinus translation):
    - stop, change direction
  - …

- **Audiovisual associations:**
  - Discrete changes in movements are aligned to form and rhythm: This principle is backed by our observations on dance, where steps and other bodily movements are synchronized to music. Even a notorious non-dancer typically starts nodding to the rhythm of a song he or she likes. Examples for discrete changes in movements are switching the rotation direction of a rotating object, or re-triggering an animation curve in order to restart the curve.
  - Switching of different visual scenes and parameterizations is aligned to form and rhythm: This principle is an extension to the approach of aligning video footage to musical form and rhythm. In addition to footage, we align arbitrary visual scenes, or parameterizations thereof, to intervals between two events in the time domain. Examples are switching of a scene when a music piece changes, or changing the number of objects in a scene within a particular rhythmical structure.
  - Continuous parameters of objects and movements are associated to spectral and dynamical characteristics of an audio signal: While the spectral composition of an audio signal does not directly reflect an underlying musical principle (such as melody or harmony), it can be used to characterize an actual piece. An example is the assignment of a band’s energy to an object’s size or color, or to the rotation speed of a transformation.
Dynamic 3D Content
Interactive Camera & Lighting Setup

- Interactive camera placing
  - Procedural encoding of cinematographic rules

- Lighting setups
  - Global lighting setup (three-point lighting) in core
  - Design specific spotlights etc are difficult to generalize i.e. lighting is generally done on a per object/design basis

Three-point lighting:
Performance Workflow

- Two stages: preparation phase and performance
- Preparation phase:
  - Analyze event, location and target audience
  - Prepare media and designs according to the needs, wishes, and constraints of the ‘client’
- Performance:
  - Application of designs (rare: live modeling of novel designs)
  - Parameter tuning, media selection & modification of designs
  - Improvise accordant to music, crowd & environment!

The interactive design process can be split into two major stages: In a preparation phase, designs are created in terms of an upcoming performance. In the second stage, the performance, the predefined designs are selected and further modified. How detailed a performance is prepared is not technically constrained but rather the artist’s choice, and often influenced by external factors such as the performance’s duration. Because several parameters (e.g. those resulting from real-time music analysis) are unknown during the preparation phase, these designs are typically anticipatory – their final visual rendering is partially unknown and only emerges in function of the environment during the live performance. This property is particularly effective for VJ performances that usually last several hours with musical content that is not predictable in advance. In addition, anticipatory designs provide enough space for improvisation during the performance.
Performance Workflow
Live Compositing of Layers

- Usual workflow:
  1. Select/adjust the novel layer in the preview display
  2. Activate the prepared layer in main screen(s) e.g. fade in

- Preview tasks:
  - Adjust dynamics (map to audio): speed, ranges & rhythm
  - Test layer composition with currently active layers (blend modes etc)
  - Color corrections (brightness & contrast)
Throughout the performance, the artist selects, fine-tunes, parameterizes, and applies designs, thereby countervailing the unknown and adding his perception of the moment.

The figure demonstrates the application of valuesets (from left to right: original state to fully activated valueset). The user defined a valueset called “bright” as part of the root design and applied it to a design consisting of a geometry and a video clip. The valueset affects several processing graph nodes of the root design: The colors of geometry nodes, the luminance-based selection of video footage in corresponding NLE-nodes, the three-light setup and the glow node. Since these nodes and the valueset are part of the root design, the valueset is applicable to all other designs (as long as they make use of the nodes defined in the root design).

All changes to the system state are tracked and can be committed to build new design tree nodes in order to be reused. Using design operations an interpolation from the current design to a target design is easily achieved.
Performance Workflow

Brightness, Contrast & Environment

- Problem with projectors: Low contrast
  - In clubs, only footage/designs with high contrast can be used

- Problem with plasmas/LCDs: Too bright

- Problem with multi-screen setups: Sync
  - One screen per PC (with standard gfx-cards) or DualHead2Go

- A good (multiple) screen setup is crucial
  - Visuals on multiple screen are difficult to control
  - Take into account the architectural environment & the changing real-world lighting (spotlights etc)
  - Compose visuals according to screen size & aspect ratio
Acknowledgments

- Stefan Müller Arisona
- Simon Schubiger-Banz
- Matthias Specht
- Philippe Wüger, Mortiz Oetiker, David Stadelmann and Curdin Grond
- Prof. Luc Van Gool, Prof. Jürg Gutknecht and Prof. Christoph P. E. Zollikofer
The ZKM | Center for Art and Media has an over 15 years history of production in the field of art and media. A strong focus of the Institute for Visual Media has been on real-time interactive installations and projection environments. Various productions have been realized for different contexts such as installations for museums, stage performances and operas, dome projections for entertainment parks and real-time projections for concerts.

In order to meet the requirements of the creative process on the one hand and the stability of the presentation on the other, we developed our own software systems implementing certain key concepts which have been the basis for several major art works.

In this part we give an overview on artistic strategies dealing with interactive real-time graphics and sound and describe the relevant software concepts.

The image shows the installation “confIGURING the CAVE” by Agnes Hegedues, Bernd Lintermann, Jeffrey Shaw and Leslie Stuck (Music), 1997, conceived for the CAVE™ environment and part of the permanent collection of the NTT InterCommunication Center in Tokyo.
Introduction

Addressed Topics:

- Software Strategies
- Strategies to Create Complexity
- Interaction Strategies
- Presentation Strategies

This talk reflects the author's 10-years experience in creating and producing interactive and performing art works. It addresses strategies of software systems which are accounting for the production process of a work. It shows some strategies for creating visual complexity, and it displays some strategies of dealing with users in an exhibition or in mass public environments.
Xfrog is a procedural modeler for organic objects like trees, flowers but also for abstract organics like radiolaria.

The user edits a component hierarchy which gives a structural description of the model. Components are designed to reproduce form principles observed in nature, for example the proportion of the golden section is used for the simulation of spiral phyllotaxis. The animation of parameters allows the simulation of growth processes.

Components are independent entities and freely combinable, the foliage of trees for example can consist of any geometry, for example of teapots instead of leafs. Any parameter can be animated independently resulting in a great variety of real and abstract shapes.

The method is described in:

The Tree of Knowledge is an early example of a realtime controlled video sequence. The installation consists of a 17m corridor with an animated tree projected at the end. The tree changes in relation to the visitor’s presence in the environment. As the person approaches the tree, it undergoes a process of aging, changes color, and grows old. When the visitor arrives at the culminating image the tree is dead and the visitor lost control.

The material of this work is the experience of time itself, as it is engaged in personal reflection by the symbolism of the experience of walking through one’s life.

The animation of the tree is prerendered using Xfrog organic modeling software. A laser tracking system located at the entrance of the corridor detects the visitor’s location and controls an Abecas Video Disk Recorder.
MTK is a software which has been used at ZKM for various interactive installations. Originally conceived as just a software layer implementing the control logic inbetween input devices and graphics or acoustic rendering systems (MTK stands for Mapping ToolKit), it became quickly a complete rendering system for visual (3D) content replacing Xfrog as production tool.

A mixture both of the dependency and control flow paradigm has been chosen as graph interpretation scheme. Control flow based systems are very popular in the interactive media art community since they allow for state changes in the application and subgraph graph execution respective to user interactions or performance narratives. MaxMSP with its addition for video and 3D graphics processing (Jitter) is still one of the most widely used tools in interactive media art. On the other hand in many modelling systems dependency graphs have been established, e.g. in Maya, relieving the user from the responsibility to understand the temporal logic of the graph’s execution. In MTK both concepts are incorporated. The control flow paradigm is used to control subgraph execution to acknowledge state changes and force the (repeatedly) execution of a subgraph within one graph execution step. Dependencies are used for assuring consistency of the incoming data. Graph connections visually encode whether they define data dependencies or control flow relations. In order to incorporate the generative capabilities introduced in Xfrog, a node can force the target graph of a connection to be executed. Thus, propagating scenegraph connection points, a node can force a subgraph to repeatedly generate geometry with different parameter sets and different geometric transformations.

A plugin architecture allows programmers to extend the system by developing custom processing modules (Nodes), data types (Attributes) and GUI representations of data types for interactive data manipulation during the graph editing.
MTK provides a library of nodes for standard tasks, e.g. mathematical operations, spline mapping, control value generation, timers, constraint random values, device handling, scene graph management and others. Custom nodes are written in C++ for specific applications, thus naturally extending the system capabilities over time, e.g. by image and sound processing, live video input, web server functionality, particle systems, multiple viewport rendering, movie playback etc.

Application logic which is either difficult to represent as a graph or would result in a very complex graph, is usually implemented as a separate computation module (node). If encapsulated in a node, it can be easily reused within another context. In a certain perspective these complex nodes can be seen as „frozen“ ideas and in consequence the library of nodes can be regarded as a library of ideas.
Even though graph based systems have very useful properties, we do not want to mask out certain problems. Graph based systems may also be handled by non programmers, and enables them to implement an application logic of a certain complexity. Complex tasks, given a small granularity of node semantics, easily result in large graph representations. To address this problem, graph systems usually offer the possibility to encapsulate subgraphs in abstractions which are represented as single nodes with incoming and outgoing connections (interface). As soon as the number of data dependencies between nodes within the network increases, this approach fails since the interface of the abstraction has to reflect the complexity of the data dependencies.

During the development of an installation or performance certain ideas can quickly be tested using the existing library modules (rapid prototyping). But it turned out that at a certain level of complexity, one usually begins to encapsulate prototyped functionality by reimplementing it in application specific nodes. As a side effect the execution speed of the application increases.
Sketches of Utopia is an example of geometric structures changed by a genetic mechanism. It is implemented with proprietary modifications to the organic modeller Xfrog. Sketches of Utopia is a derivation of the original stage design for the concert „Utopia Triumphans“. In this concert these genetically evolving shapes were projected above a choir consisting of up to 42 persons in stereo.

In Sketches of Utopia the user switches between 8 different setups of gene pools each of which is constituted by 8 to 18 single components. Starting from a specific basic structure, randomly selected operations either insert into or remove components from the existing structure, or change partial hierarchies in their position. The initial set of genes and the starting configuration determines the aesthetic system.

The slide shows on the left examples of two different setups, and on the right a third setup with four evolved structures and their corresponding component hierarchy.

The genetic installation "SonoMorphis" is realized with proprietary modifications to the organic modeller Xfrog. A three dimensional stereoscopically depicted image of a ramified organic object is projected on the front wall of a room. A control unit is placed in the middle of the room so the viewer can change the composition of the object and determine it’s visual and sonic elements, their shape and sound variation by using the buttons and slide controls.

Visitors evolve the three-dimensional organic object via an evolutionary metaphor. The organic is defined by a genome, a set of components, which is successively mutated by the users. Out of six randomly generated mutations users select one, which in the succeeding steps becomes the starting point for new mutations. This way users choose a thread through the space of possible forms. Via the user interface box users additionally change the shape and dynamic behavior of the life-like organic object.
Generative Systems

- Evolutionary Metaphor used for Navigation in the Configuration Space
- Operations: Mutation, Reduction, Crossover result in small steps in the Configuration Space
- Gene Pool with 32 components
  - Predefined Parameter Variations applied randomly
- Procedural Visuals mirrored by Procedural Sound
- Varying Parameters are shared by Visuals and Sound

In "SonoMorphis" the user modifies a Xfrog component hierarchy. Single components with associated information on constraint parameter variations are defined individually and stored as single components in the so called gene pool. The operation Mutate either selects one component of the gene pool by random and inserts it somewhere in the component hierarchy, deletes a component, or exchanges subhierarchies in the component hierarchy (cross over). The operation Simplify reduces the number of components in the hierarchy.

With the first implementation 1998 running on a SGI Onyx, the number of possible configurations of components - considering the speed of the machine and the average costs of the creation and rendering process – calculated to about 10 to the 80 possible forms. The evolutionary metaphor of the user interface allows for an intuitive navigation in this huge space of possible configurations. The system ensures a smooth transition of the appearing and disappearing components. Component parameters like curvature, shape and growth factor are continuously varied for a smooth organic animation.

While the visuals are based on an organic modeller implementing form and shape principles in nature, the sound is using physical modelling as generation model. Each gene in the gene pool has an associated sound model. All visual parameters like curvature, shape etc. and as well the positions in space are mapped to parameters of the sound generation resulting in a strong dynamic correlation between image and sound.

In this way automatic compositions arise, the results of which are functions of their components and are variable in the details of their contours, complexity, and their behaviors. The overlapping of visual levels and sound levels produces an open structure that can be continually and endlessly configured in new ways by each viewer.
Since the installation SonoMorphis is conceived as a audio visual instrument the purpose of which is certainly to create convincing sensual dynamic structures, it has been reimplemented in spring 2007 in an extended immersive environment. The visuals are projected on a 180 degree stereoscopic projection screen with 6 SXGA+ projectors in an effective resolution of 8000x928 pixels per eye. The so called audio dome consists of 41 speakers in a semi-spherical arrangement. This environment implements a more accurate dynamic spatial correlation of visual elements and their acoustic correspondents compared to previous versions.

The software Circonium developed at the ZKM | Institute for Music and Acoustics allows programmers and composers to distribute an arbitrary number of audio channels in a transparent way. It appears to the programmer as a black box fed by audio streams which are spatially distributed in real time via OSC controls. For system simplicity the visuals are calculated on a single PC with a NVIDIA Quadro and a NVIDIA Quadro X2 board delivering 6 high resolution video signals. A correct omnidirectional stereoscopic projection is achieved by recalculation the vertex positions of the object in 3D space, dependent on their relation to the viewer and the screen before passing them to the OpenGL rendering pipeline.
The artwork’s algorithmic emergent tapestry of audiovisual and thematic correspondences is activated and modulated by the patterns derived from the palms of visitors’ hands that are being scanned and entered into the system from the local and remote input terminals. These varied and always uniquely individual palm lines appear on the installation’s screen, with a caption indicating the name of the location where that person’s hand was scanned, e.g., New York, Tokyo, Berlin, Melbourne, etc. These lines then merge into and activate a singular sequence of transformations on the screen, and also in the musical score that accompanies the imagery. In this way each visitor connects with and breathes new life into this networked artwork – with a ritual handshake they awaken a world of visual and thematic correlations and make themselves protagonists of the Web of Life.

The audio-visual environment is formed by an immersive conjunction of projected three-dimensional computer graphic and video images, together with a fully spatialized acoustic experience and a specially conceived architectural surrounding.

For more detailed information see:

http://www.web-of-life.de/
Web of Life is configured as a distributed network of installations, one large-scale environment situated permanently at ZKM, and four others designed to travel to various locations around the world during the period of the project from 2002 to 2004. User interaction at any location communicates with and affects the audio-visual behavior of all the installations.

Interaction is effected via a hand scanning user interface. A camera within the user interface captures an image of the user’s hand. The hand lines are extracted and converted into a vector representation using usual image processing algorithms. The hand line vectors are then distributed via internet to all connected stations.
The visuals consist out of two layers. A continuously changing cell structure in the background displays movies from a database of 192 single movie clips. The database is organized as a 8x8 matrix in three layers, each row in the matrix representing a network domain, e.g. transport, telecommunication, etc, and each column representing aspects of networks, e.g. users, data, velocity. Each movie in a column has an aesthetical treatment such that they form an aesthetical system making relations between different types of networks.

The front layer shows a dynamically changing network of lines, which constitutes from the user’s hand lines distributed via internet. The behavior of the network is inspired by biological neuron growth. Regions of hormones surrounding the neuron’s growth buds attract or reject other neurons. An additional ambient hormone distribution creates a continuous external disturbance.
In the installation Cupola the projection is undertaken inside a suspended semi-spherical projection screen, in which the viewers are invited to stand in or lie under to look up at the imagery. The thematic focus of this work was the ceiling architecture of buildings in the city of Lille, ranging from churches, to municipal sites, factories, shops and houses, and showing an appreciation of their formal beauty, everyday ordinariness and decaying structure. Over 200 images present a variety of architectonic structures of various scales and identities, both historical and contemporary. A subsequent version has been created for the city of Kyoto.
The imagery is projected by four SXGA projectors. Two computers synchronized via ethernet project fisheye images of 2048x2048 pixels resolution onto the dome’s inner surface. Image warp and edge blending are done in software, thus being independent of the projector capabilities and allowing a quick adjustment of the image distortion on location.

The images in the database are grouped in categories out of which images are displayed in a random permutation order to prevent early repetitions. Two pairs of images are cross-faded by 18 transitions applied in random order. Image transitions are calculated on the GPU customizing vertex and pixel shaders and make extensive use of image filtering techniques like edge detection, e.g. to generate image content dependent pixel warp vectors.
conFIGURING the CAVE is an application using the original CAVE technology, a four sided cube with stereo projection on the left, right and front wall and the floor. The problems of attaching cabled interfaces to the viewer was felt to be inappropriate in a museum context, so we developed a unique interface concept - a life size wooden puppet which could be handled by the visitors and where the movement of its head, body and limbs controlled all the audiovisual transformations in the VR environment. Seven different environments, which the user changes by closing the puppet’s eyes with the puppet’s hands, explore different relations of body and space.

Xfrog has been extended by a proprietary software camera to render the four sided stereo projection. Potentiometers in the wooden puppet’s limbs allow for a virtual reconstruction of the puppet’s posture. The orientation of the puppet’s head is always controlling the orientation of the virtual camera. In some environments potentiometer values are directly mapped to parameters of the procedural modelling to control for example the distance and orientation of objects, whereas groups of potentiometers are calculating higher levels of control values. Postures are used to trigger events in the environment. Constraint random variation of parameters create a dynamically animated environment.
In a later version of reConFIGURING the CAVE at the Louisiana Museum in Denmark, just a two screen CAVE was built, but with a very wide aspect ratio (3m x 5m). The result was clearly more appropriate for large audiences, with little loss of the immersive 3D experience.
Commissioned by Skoda, a czech car company for the Volkswagen Autostadt in Wolfsburg (Germany), the demand for this installation was to communicate the notion of czech culture history. Four projectors are used to cover the surface of a 12 meter diameter dome. The visitors can explore seven different audio visual environments focussing on different aspects of czech history of reaction and invention. The Panoramic Navigator, conceived by Jeffrey Shaw, was felt to be an appropriate solution as a user interface for a location that attracts thousands of visitors per day. The Panoramic Navigator consists of a revolving column with an integrated touch screen and a video camera mounted on top, embodying the concept of ‘augmented virtuality’. Depending on the orientation of the navigator, a portion of the dome is captured by the video camera and displayed on the touch screen. Sensitive areas with pictograms layered on top of the camera image allow the visitor to trigger visual and acoustic events in the environment.

MTK has been used for both the user interface and the application software to create and animate the real time virtual scenery.
Christopher Marlowe's drama “The Jew of Malta” provided the subject matter for André Werner’s opera. The composer condensed the text, so that the main plot lines are highlighted: the relationship between the three monotheistic world religions and their intertwinements with power and politics. Werner also rearranged the sequence of the scenes, and thus the plot does not unfold in a linear fashion. Instead there is a system of anticipations, references, cross references and flashbacks. Machiavelli, who only speaks the prologue in Marlowe’s play, constantly intervenes in the course of the events as the “director” in Werner’s opera. He also controls the virtual stage sets in the beginning, which are as well as the costumes created by means of projections, but he loses control over the virtual stage more and more during the course of the opera.

Costume projection is realized with a camera tracking system. Two projectors placed in front of the stage project textured masks onto the singers who are wearing white cloth. Depending on the role within the original play which Machiavelli assigns to the singers, the singers are projected with the costume which corresponds to the role. Two cameras are located at the projector’s positions. With additional four cameras a voxel model of the singers is created to compensate for ambiguities when singer’s contours are overlapping.

Six large screens are positioned on the stage establishing the virtual scenery. Three networked PCs generate architectonical content with different methods, for example geometry attached particle systems, space subdivision techniques or dynamic texturing.

A infrared camera tracks the protagonist’s position and hand gestures at a pedestal at the right side of the stage. From this “control center” Machiavelli influences the architectonical space by means of motion, rotation, wiping and pointing.

For Credits and a more detailed description see

http://www.bernd-lintermann.de/CV/JVMCredits.html
Q & A