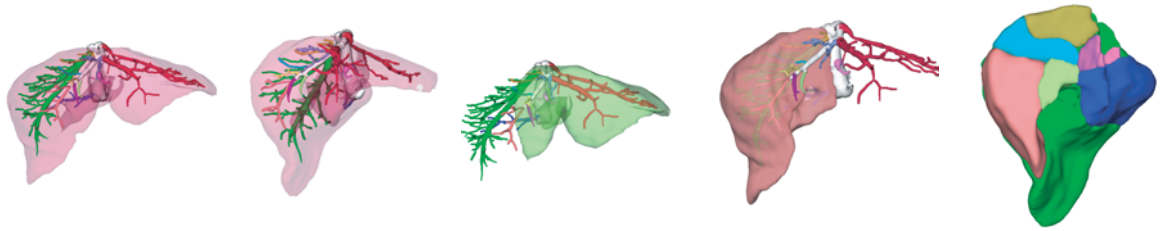
Adaptive script based animations for medical education and intervention planning

Konrad Muehler Ragnar Bade Bernhard Preim

Department of Simulation and Graphics
Otto-von-Guericke University of Magdeburg, Germany
{muehler|bade|preim}@isg.cs.uni-magdeburg.de



Abstract

We describe scripting facilities to create medical animations for intervention planning as well as for educational purposes. We developed a data independent script language to separate animation scripts from imaging data. Our scripting facilities are adaptive and allow to reuse one script to create animations for many different patients where the same anatomic structures and their spatial relations should be visualized. Decomposition rules map abstract high-level instructions to low-level instructions. We designed an object classification and assigned visualization parameters to each category to make the scripting process more effective. We also discuss the enhancement of interactive explorations with animations generated on the fly. In particular, the automated emphasis of objects benefits from appropriate animations.

1. Introduction

Advances in imaging technology are continuously increasing the amount and the complexity of data and information available to medical doctors. To take advantage of these large amounts of data for intervention planning, the effective presentation and interaction with these data has to be considered.

For intervention planning, usually slice-based visualizations of medical volume data, such as CT or MRI, and derived segmentation information, are used. In very specific applications, standardized static three-dimensional visualizations are provided as a service [MDSA05]. Two major drawbacks of those visualizations are (i) difficult to recognize exact spatial relations between the objects and (ii) occlusions of relevant objects or parts thereof. These disadvantages can be compensated by interactive 3d-visualizations. By navigating through a scene the viewer has a better impression of shape, size and spatial relations of objects. However, navigating in 3d-scenes is time consuming and the viewer can lose sight of important objects or lose orientation because of insufficient 3d-interaction techniques [BRP05]. Therefore, interactive exploration is rarely used in discussions of medical doctors, for example when a radiologist demonstrates a case for a surgical team.

Some of the problems of both static graphics and interactive scenes can be avoided by using carefully designed animations which effectively convey information in a limited and pre-defined period of time. Animations of medical volume data convey spatial relations, size and shape at the expense of reduced facilities to explore the scene. Animations are predefined image sequences designed by an *author*

for a specific group of users, such as medical doctors from different specialties in a tumor board discussion.

The creation of useful and expressive animations for intervention planning involves the specification of camera movements, movements of clipping planes, and the interpolation of transfer function parameters or parameters for surface rendering. This process is time-consuming, error-prone and requires deep and specialist skills. In intervention planning, appropriate visualizations and animations are required for many different patients. The therapeutic question, however, is often the same: for example the evaluation of resectability of a certain kind of tumor. Therefore, it is essential that animations designed for a particular case can be easily adapted or reused for another patient where the same therapeutic question is addressed. The reuse of animations is not only an issue of reducing effort but also essential to make the preoperative planning process more reproducible.

Reuse of animations is also desirable for anatomy and surgery education where an important objective is to convey anatomic variants and their consequences (variational anatomy). While some aspects are unique for a case, other anatomic structures are very similar with respect to their location and size. Due to these similarities, a reuse of animations is possible.

In order to allow an effective use of animations in intervention planning and in educational systems, we defined scripting facilities to *declare* an animation in a way which is independent of a particular case. We describe a scripting language which enables an author to describe the behavior of objects in a simple manner. To separate the description of animations from the individual patient data our animation system is adaptive.

Besides the use of completely rendered animations the execution of animations during interactive exploration is conceivable. Depending on user interactions, such as selection of an object, animations can be executed which provide a scene overview or a scene introduction. The design of our scripting facilities also considers animations integrated in the interactive exploration.

2. Related Work

The pioneering work [Zel90] introduced intent-based animation using a high-level scripting language and decomposition rules, to map higher level commands to lower level commands. They defined objects and agents in a virtual environment with four levels: a task level, a functional level, a procedural level and a machine level. At the machine level, basic physical attributes of objects are defined. At the procedural level, mechanisms are defined to control object motion like collision detection and deformation. At the functional level, objects are associated with procedures to define behavior of single objects. The task level defines the behavior of complete agents in the environment. The advantage of this hierarchy is the ability to hide underlying structures and functions in each level. [KF93] used this approach to decompose communication goals to animation instructions. They separate a script for animation from the description of communicative goals. Additionally, they use knowledge about causal relationships between objects and build up their animations on a traditional hierarchy with sequences, scenes and shots.

According to [TMB02] animations must be slow and clear enough for viewers to perceive and understand changes. For that reason emphasis techniques are appropriate for animations. They can be distinguished in local, regional or global techniques according to changes influencing an object in focus, nearby objects or the whole scene [PTD05]. Emphasizing techniques are essential for information transfer in animations for both educational purposes and therapeutic issues.

For clinical purposes standardization is also an important issue in the design of animations. [IBRSE*01] generated clinical videos efficiently using standard views and visualizations parameters. [THIB*03] evaluated the benefit of standardized high quality videos in clinical routine and state that those videos are a viable approach for standardized and reproducible visualizations. Facilities to improve interactive dataset traversal by animation of transfer functions for volume rendering were introduced by [CS05]. They showed examples of flow in vascular structures. [KTH*05], in cooperation with medical doctors, defined a standardized process for visualization of patient data in Ear-Nose-Throat-surgery (ENT-surgery). They identified default colors, texturing and viewpoints for important structures in the neck region. [PRS96] introduced a script language for anatomy education. However, volume rendering and slice-based rendering was not considered.

Modern software libraries for scientific and medical visualizations provide facilities to create animations. In AMIRA [SWH05], animations can be created for single data sets by defining events to be processed or by writing scripts in a TCL extending language. The event lists or scripts are associated with the particular data set and can not be applied to other data sets. Furthermore, the script instructions are very complex and difficult to handle. The library MEVISLAB [HLP03] is specialized to analyze and visualize medical image data. It provides the possibility to execute simple actions like rotations automatically. Also, a low-level scripting facility is available. Moreover user interactions can be captured and recorded for creating an animation directly. The interaction capturing is very extensive and not reproducible. PARAVIEW [LHA01] is a visualization software for large data sets that does not support medical image data, but does support the creation of animations. Like AMIRA, it supports a TCL like script language and permits the creation of animations by defining key frames. The definitions of anima-

tions are restricted to one data set and not portable to similar data sets.

3. Script Concept

In this section, we discuss requirements for medical animations and present a script-based concept. High-level instructions are mapped to executable low-level instructions according to decomposition rules and an object classification. Furthermore we present the parsing process and the description of semantic relations to enhance animation descriptions. Finally, we present the integration of animations in the interactive exploration of a 3d scene.

3.1. Requirements for Medical Animations

The problems of medical animations are (i) the lack of portability of animations for different but similar cases with the same therapeutic question (e.g. the resectability in case of a liver tumor) and (ii) the absence of reproducibility in captured user interactions.

Thus our design is guided by four requirements for an animation system for medical animations:

1. **Adaptiveness:**
   The general location as well as the number of many anatomic structures, such as organs, is the same for virtually all patients. However, they differ in their specific shape and size. Thus the mapping of high-level to low-level commands has to adapt to these differences automatically. It also has to adapt to differences in pathologic structures which are unknown in size, location and shape.
2. **Convenience and specialisation:**
   Originating from different fields of application (intervention planning, education) the process of creating animations must strike a balance between convenience, speed and specialisation of animation.
3. **Medical animation techniques:**
   In addition to rotation, zoom or translation it must be possible to animate particular medical visualization techniques, such as volume rendering, clipping planes and slice-based visualizations. Volume rendering is often used in addition to surface rendering of segmentation results. A scripting language for medical animations should include dynamic changes of transfer function parameters. Animated clipping planes can move slowly through the scene dissecting structures of interest (e.g. a tumor) step by step. The viewer is thus guided from an overview to a pathologic structure directly.
4. **Standardization:**
   Medical animations should be standardized to afford comparisons and to facilitate the recognition of structures in different animations [THIB*03] , [IBRSE*01]. Thus similar objects should be colored in the same way in different animations and the viewpoints in a scene should be limited to a few appropriate views per object. [KTH*05] introduced a concept of defined views and specific colors using ENT-visualizations as an example.

3.2. Basic Concept

In this paper, we introduce a script language to describe animations of medical volume data and derived segmentation information. This script language is dedicated to the requirements of animations for intervention planning and educational systems.

A script instruction consists of four parts: *time*, *object name*, *instruction name* and *parameters* (see Listing 1 for an example). A script is executed on a visualization i.e. all instructions are applied to their associated objects over time. This can be a structure like an organ or a vessel but also the scene or the camera. The order of instructions in a script need not be the order of execution. Also an overlapping of time ranges are possible.

```
time object instruction parameters

[0,8] 'All' clipPlane on left 0.8 'Vessels'
```

Listing 1: *Design of a script instruction with an example instruction for moving a clipping plane through the scene. The parameters stand for a clipping plane from the left side which move trough the* `All`*-object for 80% and do not clip* `Vessels`*. See Fig. 1 for sample frames.*

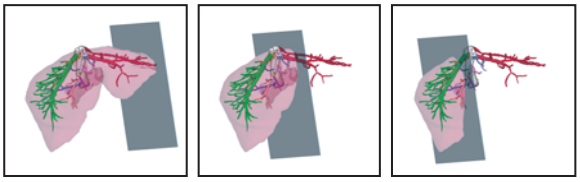


Figure 1: *Sample frames for clipping plane instruction in Listing 1.*

3.2.1. Adaptiveness

Instructions for camera positioning occur in relation to the corresponding object and are geared to the size and location of its bounding box. Thus the animations adapt themselves to size, shape and location of objects. This adaption is dedicated to compact structures, such as tumors, lung nodules and lymph nodes as well as to organs. Figure 2 presents an example for this behaviour. For elongated or branching structures, the bounding box is not a good approximation.

3.2.2. Decomposition of high- in low-level

To afford an enhancement of the script language for particular therapeutical questions, we use a hierarchical approach similar to [KF93]. In contrast to [KF93] we represent (therapeutic) questions by instructions and do not abstract parts

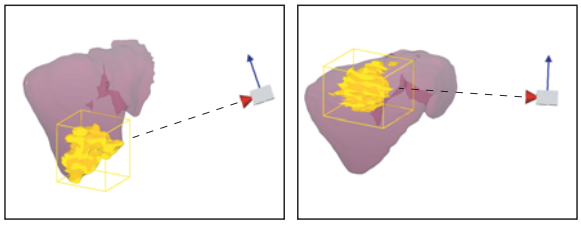


Figure 2: *Alignment of the camera with the bounding box of the corresponding object. Two different livers with one tumor in each of them are shown.*

of animations like scenes or shots. By using decomposition rules, script instructions can be abstracted. We distinguish between low-level (elementary) and high-level (abstract) instructions. Low-level instructions are e.g. `move`, `rotate` and `setColor`. High-level instructions are e.g. `emphasize`, `groupOverview` and `setAsFocusObject`. By abstracting instructions, we achieve a convenient way of describing whole animations with few instructions. To support fine-grained modifications and specialized specifications high-level and low-level instructions can be mixed.

3.2.3. High-level commands for medical animations

In addition to instructions for object property changes and camera changes, we implemented instructions with special regard to medical visualization techniques. As well as the `clipPlane` instruction in Listing 1 we created instructions for slice-based animations and simple animations of volume rendering. For sliced-based animations, the slices containing the corresponding object are identified automatically. The `showSlices` instruction has a parameter for margin slides above and below objects' slides. The following instruction show the slices containing a liver tumor with 5 slices margin:

```
[0,10] 'Tumor' showSlices 5
```

We implemented also facilities to create animations of volume rendered scenes. Because volume rendering exists in the context of surface rendering in our visualizations, simple manipulations of transfer functions are necessary only. With the `volumeRendering` instruction the transfer functions for gray values and transparency can be changed over time. For each transfer function there are two parameters for *center* and *width*. By replacing parameters with standard values they can be abstracted to parameters like *bones* or *soft tissue*:

```
[0] 'All' volumeRendering Bones
[0,5] 'All' volumeRendering softTissue
```

Examples for low-level and high-level instrcutions can be found in Appendix A and Appendix B.

3.3. Script Parsing

Before its execution, a script will be parsed, i.e. high-level instructions, composed objects and parameters will be decomposed (Fig. 4). High-level instructions can be replaced by high-level as well as by low-level instructions. The substitution of instructions can be declared in general or depending on the anatomic type of the object the instruction refers to (see Listing 3). Objects can be combined to composed objects so that instructions referring to a composed object are applied to all elementary objects after decomposing the composed object. For example `LeftLung` and `RightLung` can be combined to `Lung`. Parameters can be replaced by values, e.g. `red` by its rgb-value `255,0,0`. The decomposition rules are declared in XML, so they can be easily edited by the author of an animation.

3.4. Enhanced Script Concept for Convenient Scripting

To enhance the scripting process, we designed a two dimensional classification for objects. On the one hand, objects are grouped by their anatomic systematics (Fig. 3), e.g. muscles, vessels, bones etc. This classification is likewise declared in XML and editable by the author (e.g. extendable by pathologic objects). On the other hand, we differentiate focus objects (*FO*), near focus objects (*NFO*) and context objects (*CO*) based on the classification of objects according to their relevance as introduced by [TIP05]. Focus objects are objects, which are in the center of interest for a therapeutic question (e.g. a lymph node). Near focus objects are objects, which are important for the clarification of the issue and which are in close spatial proximity to the focus object (e.g. a muscle infiltrated by a lymph nodes). Context objects are in contrast to [TIP05] all objects that are essential to the global orientation as anatomic context. Other objects are not included in an animation.

To standardize our animations we considered defined colors and transparency values for anatomic types of anatomic objects, regarding applications in orthopedics, ENT-surgery and abdominal surgery. In addition we defined default views for important structures.

3.5. Interactive animations

Besides the use of animations as videos, animations can be combined with interactive manipulation. In educational settings but also for the individual process of intervention planning, it is essential to give the user the freedom of exploring the scene and support him or her by animations in the scene, e.g. for an introduction of the scene or an automatic camera areal view. We consider the following scenarios for such interactive animations:

1. **Selection**
   By selecting an object, by picking it or selecting it from a list, an animation emphasizes this object by moving the

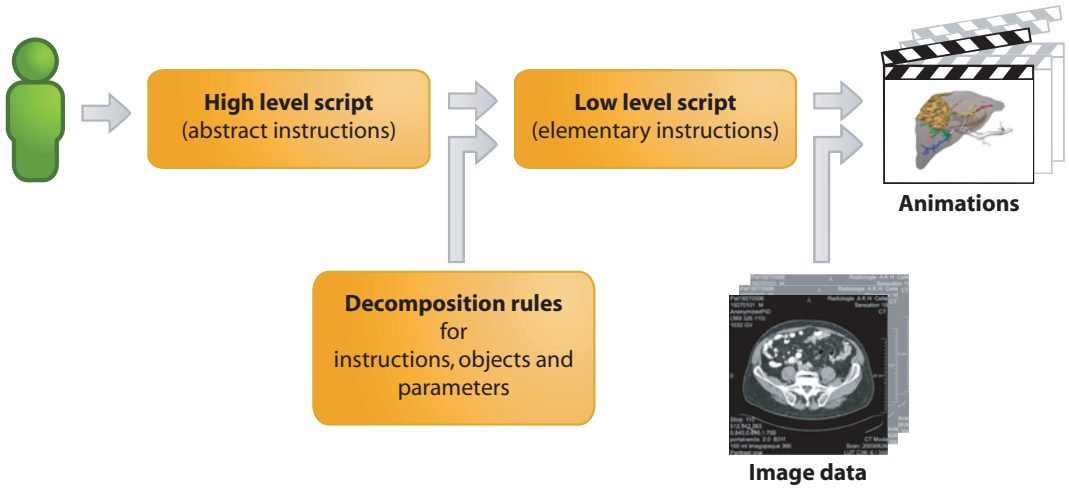


Figure 4: *Concept of transfering a high-level script to a low-level script by decomposing instructions and composed objects and by replacing parameters. The low-level script is executed on visualization of data sets to create an animation.*

- Muscles
  - M. sternocleidom. left
  - M. sternocleidom. right
  - ...
- Vessels
  - Artery
    - A. carotis communis left
    - A. carotis communis right
    - ...
  - Vein
    - V. jugularis right
    - V. jugularis left
    - ...
- Lymph nodes
  - Lymph node 1
  - Lymph node 2
  - ...
- ...

Figure 3: *Classification of objects by their anatomic type.*

camera to a predefined viewpoint for this object. Currently the camera path between two viewpoints is a path on an object surrounding sphere. The style of emphasis depends on the type of selected object as described above.

2. **Object overview**
   Instead of emphasizing a selected object an overview of this object can occur. This animation can be a rotation around the object or a serial emphasis of some near focus objects.
3. **Scene overview**
   To give the user an impression of his or her current position in the scene, a short animation can provide a camera pan to a global viewpoint and return back to the starting position.
4. **Scene introduction**
   For a first overview of the scene, an animation can present the scene with a rotation and an emphasis of the most important objects.
5. **Fade in and out**
   If the viewer explores a selected object from different viewpoints occluding objects can be smoothly faded out when they appear in the field of view and faded in when they no longer hide the selected object.

We integrated our scripting facilities to create interactive animations. This is why a script exists for every possible interaction.

4. Examples

Our concepts for generating medical animations are based on the analysis of numerous medical animations and fruitful discussions with medical professionals.

4.1. Clinical example

In ENT-surgery, the spatial relations in the vicinity of enlarged lymph nodes is a common question. To emphasize a lymph node for this question, it has to be set as *focus object* whereas objects with a certain distance have to be set as *near focus objects*. Because of the importance of bones for global

orientation they have to be set as *context objects*. Thus the high-level script instruction

```
[0,5] 'lymphN1' emphasize
```

will be decomposed by the first rule in Listing 3 to the instructions in Listing 2. The second rule in Listing 3 results in the instructions in Listing 4. More decomposition steps lead to an executable low-level script and an animation from the current state to the final state (Fig. 5).

Many of these `emphasize` instructions can be summarize to a `groupOverview` instruction to show all lymph nodes in neck region consecutively, e.g. in a clinical board discussion. This practice is also applicable for other medical questions, e.g. in oncologic surgery with many metastasis to explore.

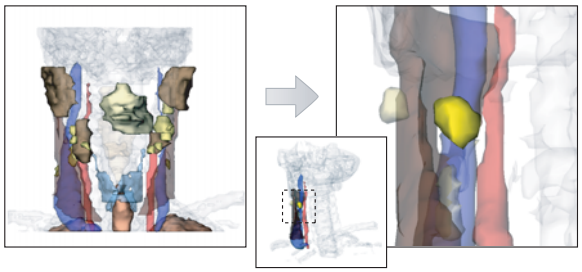


Figure 5: *Start and final state of an animation to emphasize a lymph node in its spatial context. The detail context-view is included to clarify the spatial relation.*

```
[0,5] 'lymphN1' setAsFO
[0,5] ('lymphN1' getObjects dist 20) setAsNFO
```

Listing 2: *Script after a first decomposition of the instruction* `[0,5] 'lymphN1' emphasize`.

4.2. Educational example

A target application of our animation facilities is the LIVERSURGERYTRAINER [BRS*06]. The objective of this program is to convey differences and pathologic characteristics of the human liver. The screenshots in Figure 6 are extracted from an animation which gives an overview of planning a living liver donor transplantation (see Listing 5 for corresponding animation script). First, the entire liver is shown. The vessels are color-coded. The selection of colors is guided by visualizations generated for liver surgery planning (e.g. [LRL*04]). Afterwards, the specific resection is shown, the remnant is emphasized and the camera is moved to a closer look at the cutting plane. Following an overview is given to obtain the orientation for the viewer. To explain which liver segments must be resected for the graft, the single segments are accentuated. The animation closes with an overview look.

```
<command cmd='emphasize' type='lymphnodes'>
  <command>T O setAsFO</command>
  <command>
    T (O getObjects distance 20) setAsNFO
  </command>
</command>

<command cmd='setAsFO' type='lymphnodes'>
  <command>
    T O setColor lymphColor
  </command>
  <command>
    T O setTransparency opaque
  </command>
  <command>
    T O view coronal
  </command>
</command>
```

Listing 3: *Decomposition rules to resolve the* `emphasize` *and* `setAsFO` *instructions for lymph nodes.*

```
[0,5] 'lymphN1' setColor lymphColor
[0,5] 'lymphN1' setTransparency opaque
[0,5] 'lymphN1' view coronal
[0,5] 'A. carotis_right' setAsNFO
[0,5] 'M. sternoclei_right' setAsNFO
```

Listing 4: *Script instructions after decomposing the* `setAsFO` *instruction and determining the objects in a distance of 20mm (see second instruction in Listing 2).*

```
[0,9] 'Liver' objectOverview
[9,18] 'Remnant' showResection
[18,25] 'Graft' sceneOverview
[25,60] 'Graft' showSegments
[60,70] 'All' sceneOverview
```

Listing 5: *Script to give an overview of a living liver donor. See Figure 6 for screenshots of animation frames.*

More animation examples can be found at our website http://isgwww.cs.uni-magdeburg.de/cv/projects/animation/

5. Conclusion and future work

We introduced scripting facilities which enable authors to create animations for therapeutic questions as well as educational practice. We developed a data independent script language to separate the description of animations from data sets. Because our scripting facilities are adaptive, we can use one script to create animations for many different but similar cases.

We use decomposition rules mapping high-level to low-

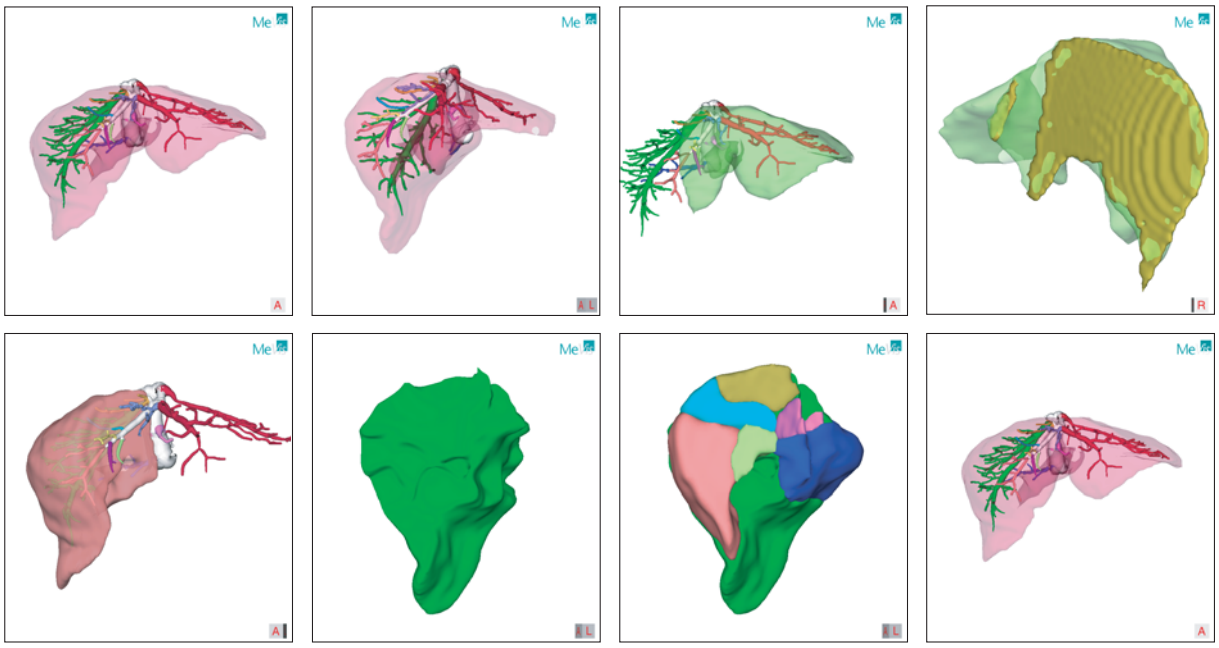


Figure 6: *Screenshots of an educational animation to give an overview of planning a living liver donor transplantation. The entire liver is shown, the specific resection is shown by emphasizing the remnant and liver segments are accentuated. See Listing 5.*

level instructions, decomposing composed with elementary objects and replacing parameters. We presented an extension to classify objects by their anatomic type and importance for a question to enhance high-level instructions and thus script design for the author.

Our script-based approach is also usable for interactive animations. The scripted animations can be used to enhance important interactions with animations e.g. a scene overview or an object emphasis directly in the scene.

Examples were presented in an educational context (liver surgery training) and in clinical routine task to examine neck lymph nodes. The scripting concept can be easily extended to other therapy questions such as biopsy needle placement.

We intend to expand our animation scripting facilities by using functions to identify good viewpoints for objects automatically and by providing the possibility for camera path planning for efficient movement of the camera, e.g. on emphasis changing. Our presented classification approach is only a first step. A more detailed classification enlarged by the functional relationships of structures is desirable (comp. [RSB98]). This would allow us to animate whole systems like the lymph system in their functional relation by few script instructions.

When the scripting facilities should be used without segmentation information, more elaborate facilities to control volume rendering parameters are desirable. The animation of further medical visualization techniques such as cut-away and ghost views [VG05, KTH*05] is also left open. The concept may be extended to address more viewers simultaneously e.g. for different views or a synchronized 2d-3d visualization.

Our animation scripting facilities are part of the educational system LIVERSURGERYTRAINER [BRS*06]. Integrating interactive animations and videos more simply in those systems with an authoring system is preferable.

Acknowledgment

The support of Milo Hindennach and Olaf Konrad-Verse from MeVis Bremen is greatly acknowledged. Many ideas are inspired by sophisticated animation sequences created for tumor board discussions manually by Milo Hindennach. We also thank the AKH Celle and the University Hospital of Leipzig for providing the data sets presented in this work. In particular the authors wish to thank Jeanette Cordes for the segmentation and Christian Tietjen for his supporting work programming the animation system. This work was supported by the BMBF in the framework of the SOMIT-FUSION project (FKZ 01|BE 03B).

References

[BRP05] BADE R., RITTER F., PREIM B.: Usability comparison of mouse-based interaction techniques for predictable 3d rotation. In *Smart Graphics: 5th International Symposium, SG 2005, Frauenwörth Cloister, Germany, August 22-24, 2005. Proceedings* (2005), Butz A., Fisher B., Krüger A.„ Olivier P., (Eds.), vol. 3638, pp. 138 – 150.

[BRS*06] BADE R., RIEDEL I., SCHMIDT L., OLDHAFER K. J., PREIM B.: Combining training and computer-assisted planning of oncologic liver surgery. In *Bildverarbeitung für die Medizin 2006 (BVM 2006)* (2006).

[CS05] CORREA C. D., SILVER D.: Dataset traversal with motion-controlled transfer functions. In *Proceedings of IEEE Visualization 2005* (Minneapolis, Oct. 2005), pp. 359–366.

[HLP03] HAHN H., LINK F., PEITGEN H.: Concepts for rapid application prototyping in medical image analysis and visualization. In *SimVis2003* (2003), pp. 283–298.

[IBRSE*01] ISERHARDT-BAUER S., REZK-SALAMA C., ERTL T., HASTREITER P., TOMANDL B.: Automated 3d video documentation for the analysis of medical data. In *Bildverarbeitung für die Medizin (BVM)* (2001), Handels H., Horsch A., Lehmann T.„ Meinzer H., (Eds.).

[KF93] KARP P., FEINER S.: Automated presentation planning of animation using task decomposition with heuristic reasoning. In *Graphics Interface 1993, Toronto, Canada, May 17-21* (1993), pp. 118–127.

[KTH*05] KRÜGER A., TIETJEN C., HINTZE J., PREIM B., HERTEL I., STRAUSS G.: Interactive visualization for neck dissection planning. In *IEEE/Eurographics Symposium on Visualization (EuroVis)* (2005), pp. 295–302.

[LHA01] LAW C. C., HENDERSON A., AHRENS J.: An application architecture for large data visualization: a case study. In *Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics* (2001).

[LRL*04] LANG H., RADTKE A., LIU C., FRÜHAUF N. R., PEITGEN H. O., BROELSCH C. E.: Extended left hepatectomy - modified operation planning based on three-dimensional visualization of liver anatomy. *Langenbeck's Archives of Surgery 389*, 4 (2004), 306 – 310.

[MDSA05] MEVIS-DISTANT-SERVICES-AG: http://www.mevis-distant-services.com/, 2005.

[PRS96] PREIM B., RITTER A., STROTHOTTE T.: Illustrating anatomic models: A semi-interactive approach. In *4th International Conference on Visualisation in Biomedical Computing* (1996), pp. 23–32.

[PTD05] PREIM B., TIETJEN C., DOERGE C.: Npr, focussing and emphasis in medical visualizations. In *Simulation und Visualisierung* (2005), pp. 139–152.

[RSB98] ROSSE C., SHAPIRO L. G., BRINKLEY J. F.: The digital anatomist foundational model: Principles for defining and structuring its concept domain. In *Medical Informatics Association Fall Symposium* (1998), pp. 820–824.

[SWH05] STALLING D., WESTERHOFF M., HEGE H.-C.: Amira: A highly interactive system for visual data analysis. In *The Visualization Handbook*, Hansen C. D., Johnson C. R., (Eds.). Elsevier, 2005, ch. 38, pp. 749–767.

[THIB*03] TOMANDL B. F., HASTREITER P., ISERHARDT-BAUER S., KÖSTNER N. C., SCHEMPERSHOFE M., HUK W. J., ERTL T., STRAUSS C., ROMSTOCK J.: Standardized evaluation of ct angiography with remote generation of 3d video sequences for the detection of intracranial aneurysms. *RadioGraphics 23* (2003), 12e.

[TIP05] TIETJEN C., ISENBERG T., PREIM B.: Combining silhouettes, surface,and volume rendering for surgery education and planning. In *IEEE/Eurographics Symposium on Visualization (EuroVis)* (2005), pp. 303–310.

[TMB02] TVERSKY B., MORRISON J., BETRANCOURT M.: Animation: can it facilitate? *International Journal of Human Computer Studies 57* (2002), 247–262.

[VG05] VIOLA I., GRÖLLER M. E.: Smart visibility in visualization. In *Proceedings of EG Workshop on Computational Aesthetics Computational Aesthetics in Graphics, Visualization and Imaging* (2005).

[Zel90] ZELTZER D.: *Making Them Move: Mechanics, Control, and Animation of Articulated Figures*. Morgan Kaufmann, San Mateo, CA, 1990, ch. Task-Level Graphical Simulation: Abstraction, Representation, and Control, pp. 3–33.

Appendix A: Sample low-level instructions

`'Cam'` **`move`** *`object coord zoom-factor`*
Moves the camera to a position on the surrounding sphere of an object

`'Cam'` **`rotate`** *`object axis angle`*
Rotates the cam around the center of an object and a given axis by an angle in degrees

`'Cam'` **`rotateCamZ`** *`value`*
Rotates the around their own z-axis

`'System'` **`setClipplane`** *`on|off left|right|... percent toClip (notTo-Clip)`*
Clips the scene in relation to an object. Objects not to clip can be given.

`'System'` **`setBackground`** *`R,G,B`*
Sets the background of the scenes

`'Object'` **`setTransparency`** *`value`*
Sets the transparency of an object to a value between 0...1

`'Object'` **`setColor`** *`R,G,B`*
Sets the color of an object to an R,G,B value

`'Object'` **`setVisible`** *`true|false`*
Switches an object on or off (it is not the same as `set-Transparency 1`)

`'Object'` **`setQuality`** *`value`*
Sets the quality an objects is rendered in

`'Object'` **`setSelected`** *`true|false`*
Selects or deselects an object

`'Object'` **`setStyle`** *`value`*
Sets the draw style of an object

`'Object'` **`shiftX`**|**`shiftY`**|**`shiftZ`** *`value`*
Shifts an object by a given value

`'Object'` **`volumeRendering`** *`value`*
Enables volume rendering

Appendix B: Sample high-level instructions

`sceneOverview`
Depending on the case (liver, ent, knee) this instruction gives an overview of the scene, e.g. by rotating the cam and showing all objects

`objectOverview`
Depending on the object, this instruction shows an short overview of the object, e.g. by emphasizing important parts

`emphasize`
Emphasize an object, e.g. by changing color, transparency and camera position

`setBaseColors`
Sets the default colors and transparencies for all objects

`view`
Abstracts the move-instruction

`init`
Initialize a scene (sets background, colors, cam postion)