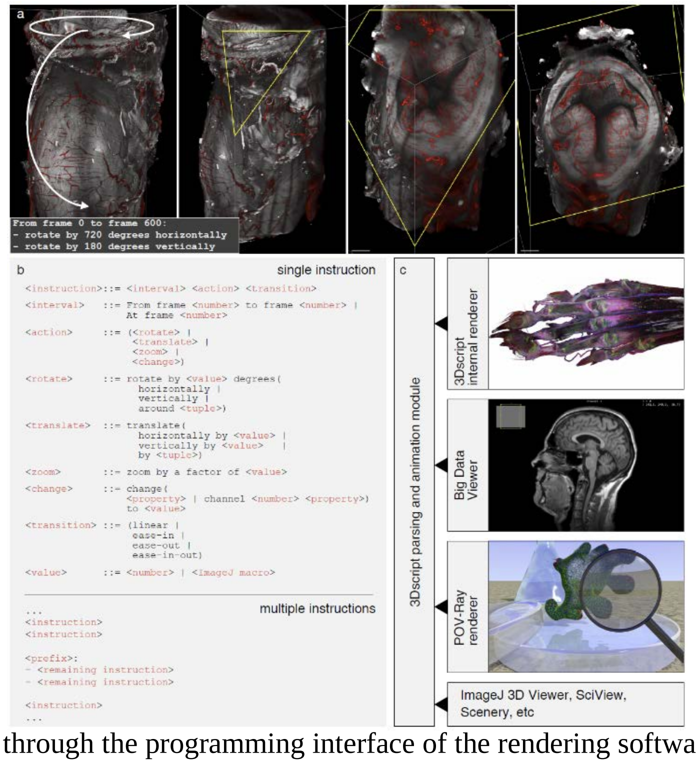
Benjamin Schmid, Philipp Tripal, Tina Fraaß, Ralf Palmisano
Optical Imaging Centre Erlangen
Friedrich-Alexander University
Cauerstr. 3, 91058 Erlangen, Germany
E-mail: ralf.palmisano@fau.de

3Dscript: Animating 3D/4D microscopy data using a natural language based syntax

Keywords: Rendering Software Framework, 3D - 4D Animation, Microscopy Data, ImageJ, Big Data Viewer, POV-Ray Renderer, Natural Language Based Syntax, 3Dscript

We introduce a software framework that assembles 3D animations of imaging data using a textual description with a syntax based on natural English language. In contrast to the established key frame based approach, smooth and complex motion sequences are intuitively implemented by concatenating multiple instructions. A dedicated autocompletion-enabled editor supports the composition of the animation text. The proposed syntax is extensible and we demonstrate its integration into existing 3D visualization software.State-of-the-art 3D rendering software for microscopy data assemble 3D animations based on key frames (Imaris, Arivis 4D Viewer, Volocity, FluoRender). The user typically stores different rendering settings and scene transformations in a number of key frames along a timeline, and the rendering engine creates a smooth animation by interpolating between them. Key frames save the spatial transformation of an object as a state, e.g. three angles for a rotation, and not as a transition (e.g. an angle and a rotation axis). Therefore, at least four key frames are required to define a full 360-degree rotation with constant speed unambiguously. Significantly more are required if the motion is non-linear, which is essential for achieving smooth and pleasant motions. Software packages therefore typically provide a shortcut to insert key frames for rotations around common axes. More complex motions such as combined rotations around multiple axes, however, are difficult to achieve.


a
b
single instruction
multiple instructions
c
3Dscript parsing and animation module
3Dscript internal renderer
Big Data Viewer
POV-Ray renderer
ImageJ 3D Viewer, SciView, Scenery, etc

```
From frame 0 to frame 600:
- rotate by 720 degrees horizontally
- rotate by 180 degrees vertically
```

```
<instruction>::= <interval> <action> <transition>
<interval>   ::= From frame <number> to frame <number> |
                 At frame <number>
<action>     ::= (<rotate> |
                  <translate> |
                  <zoom> |
                  <change>)
<rotate>     ::= rotate by <value> degrees(
                   horizontally |
                   vertically |
                   around <tuple>)
<translate>  ::= translate(
                   horizontally by <value> |
                   vertically by <value>   |
                   by <tuple>)
<zoom>       ::= zoom by a factor of <value>
<change>     ::= change(
                 <property> | channel <number> <property>)
                 to <value>
<transition> ::= (linear |
                  ease-in |
                  ease-out |
                  ease-in-out)
<value>      ::= <number> | <ImageJ macro>
```

```
...
<instruction>
<instruction>

<prefix>:
- <remaining instruction>
- <remaining instruction>

<instruction>
...
```

Some 3D rendering packages record animations directly while the user transforms the object interactively, e.g. with the mouse pointer. Despite exceeding intuitiveness, smooth animations of high quality can hardly be achieved, and concatenated transformations like rotations around multiple axes are almost impossible.

3D animations can also be created using scripting to directly render a sequence of frames by manipulating the 3D scene through the programming interface of the rendering software. Theoretically, any kind of motion can be created in this way, but usage is severely limited due to the requirement of programming skills.

Here, we propose to describe animations as textual instructions. The syntax is based on natural (English) language instead of a traditional programming language, to maximize user experience and to provide easy access for users of distinct fields. Each instruction is an English sentence. Complex motion sequences are implemented by combining multiple instructions which are applied in the order they appear in the text. The syntax comprises phrases for standard non-linear accelerations, resulting in smooth and natural animations. Additionally, self-defined macro functions can be used to define motion and other parameters as functions over time. We provide an implementation as a Fiji/ImageJ plugin including an hardware-accelerated 3D renderer and a dedicated editor featuring auto-completion and recording. Although tailored to the included rendering engine, both the syntax and our animation framework can be easily adapted to be used by other rendering software (ImageJ 3D viewer & Povray).