

## GVCL - README for SCEC

Mon Mar 31  
thieboux@isi.edu

- 
1. Introduction
  2. Staging parameters
  3. Viewer parameters
  4. Operating the GUI
  5. Command prompt
- 

### 1. Introduction

The GVCL code serves as a lightweight version of the fully distributed volumetric time-series browser. Instead of running parallel dynamic filters, the data is cooked in advance during staging, and is expected to reside locally. Interactive operations are extremely limited, and do not include zooming into a sub-region for finer detail. What the operator will see is the whole spatial range of the volume at any timestep, at the sampling resolution that was selected during staging.

The source data is required to consist of a uniform grid of byte scalars, or 32 bit floating point scalar values, of some known dimensions. Following the SCEC storage model, the cooker assumes that multiple time steps are stored sequentially in a single file. Several files can be cooked in sequence to reassemble a longer time series.

The cooking process involves sorting the scalar values into a set of narrow buckets, where each bucket stores all volume spatial coordinates that correspond to each bucket range. Each bucket is stored as a separate binary file, and contains 3d point positions of all samples that were assigned to that bucket. Samples are taken stochastically, using tetrahedral interpolation to minimize spatial quantization artifacts. Once the buckets are generated, the original values for the samples are lost. The cooker will print out a histogram of the bucket distribution, to help the operator determine if the range and number of buckets is appropriate for the distribution of the data in question.

By storing each bucket separately, the browser will only read those files that are specified by the browser range and comb controls. This makes optimal use of the disk-storage bandwidth, since most useful volume-rendering transfer functions will cull out large percentages of the volume anyway. Similarly, invisible primitives will not be transported over the graphics bus, or rendered.

In the GUI, range is specified in terms of bucket indices. For example, if the raw data is mapped into 128 buckets, selecting range 64 - 84 will result in displaying buckets 64 to 84.

The 'comb' concept allows the operator to select a sparse range of values to be displayed. For example, a comb value of 5 will display every 5th bucket within the range of interest. In the example above, this will result in displaying buckets 64, 69, 74, 79, and 84.

The 'tooth' control augments the comb control by allowing the operator to adjust the thickness of a comb tooth, as a value between 0 and 1. A tooth will be at least 1 bucket wide. A comb value of 8 and a tooth value of 0.5 will result in displaying 4 sequential buckets, skipping the next 4, and so on.

Finally, the browser supports taking snapshots and movies, which are output in the portable 'PPM' format.

-----

## 2. Staging parameters

The cooker requires a parametrically structured directory tree in which to store its output buckets. The 'sh-run-build-bucket-dirs' shell script is used to build these directories automatically by the cooker. The directories are then filled with buckets according to a set of parameters that describe the time-series, and determine the mapping of raw data values into discrete buckets:

- resolution of the grid
- number of buckets
- range of scalar values to map over the whole bucket set
- point sampling rate
- datastep range

```
linux/run-sample-mbrick f 0 /tmp/SSZ3DNRVDD1 /tmp/scec_ssz_li \  
151 108 28 128 0.0001 0.01 1.5 0 499 1 0
```

Parameters:

```
raw data format: f  
byte offset:    0  
raw datafile:  /tmp/SSZ3DNRVDD1  
directory:     /tmp/scec_ssz_li  
resolution:    151 108 28  
num buckets:   128  
scalar range:  0.0001 0.01  
sampling rate: 1.5  
datastep range: 0 499  
increment:     1  
base index:    0
```

Raw data format can be 'f' for 32 bit float, or 'b' for byte. The code currently takes the absolute value of a float scalar. This can be turned off only by recompiling the code.

Byte offset specifies to offset of the first scalar value, in case

there is some header information to skip.

The point sampling rate is a linear measure of the number of samples taken per cell. A rate of 2 will generate  $2^3 = 8$  samples per cell. A rate of 1.5 will produce  $1.5^3 = 3.375$  samples per cell. Partial samples indicate a statistical average.

Increment allows the operator to skip raw datasteps, and rename them as a contiguous series. The base index allows the operator to piece together a complete series from multiple files. It specifies the index of the first datastep to be created by each operation.

---

### 3. Viewer parameters

Once the buckets are generated, the browser can be started using a similar set of parameters:

```
linux/run-viewer /tmp/scec_ssz_li 151 108 28 samp1.5 short 0 499 0 127
```

Parameters:

```
directory:    /tmp/scec_ssz_li
resolution:   151 108 28
sampling rate: 1.5
point format: short
datastep range: 0 499
bucket range: 0 127
```

By default, the comb value is set to 5. The bucket range at the commandline will specify the initial bucket range to be displayed. The operator should see buckets 0, 5, 10, 15, ... 125.

---

### 4. Operating the GUI

The graphical user interface offers a few basic controls and some feedback displays. The data volume appears in the middle of the screen, surrounded by a wireframe outline of its spatial extent. There are three colored edges:

```
red: X-axis
green: Y-axis
blue: Z-axis
```

Around the data appears a faint circle, which serves as a 'crystal-ball' interface to rotate the data using the mouse buttons. The left button rotates quickly, and the middle mouse rotates slowly.

In the upper left of the screen, a logarithmic histogram shows the point counts for each bucket, and the color mapping as it applies to the buckets. In principal, a custom colormap can be loaded by the

operator, but for this release, it is fixed to a spectrum.

Attached to the histogram is a range slider for selecting bucket range, as well as sliders for comb and tooth, as described above in section 1. A brite-ness slider controls overall brightness of the splats, and a splat selector allows the operator to choose between high-performance point splats, billboarded triangles, and billboarded gaussian splats for higher quality.

In the lower left of the screen there are controls for selecting a timestep, as well as an automatic timestep increment for animating the series.

In the upper right appears a frames-per-second indicator, which is based on the actual loop duration of the previously rendered frame.

-----

## 5. Command prompt

The viewer program checks the commandline for text based commands, used for generating snapshots and movies, as well as controlling the timestep and animation intervals.

The 'snap' command:

```
> snap <prefix>
```

will save an image of the full window to disk, with the name

```
"<prefix>.ppm"
```

The 'msnap' command:

```
> msnap <prefix> <num-frames> [<base>]
```

will save <num-frames> images of the full window to disk, with the name

```
"<prefix>.<index>.ppm"
```

where <index> is set to ( $\text{<base> + <frame-count>}$ ), <base> is 0 by default.

The 'anim' command:

```
> anim real  
> anim dt <delta>  
> anim step <delta>
```

controls how real clock time is used to animate both the animated rotation, and the automatic stepping of datasteps. Under ordinary interaction,

```
> anim real
```

will induce nearly constant apparent rotational rate, regardless of the actual rendering rate, by taking the rendering time into account.

When rendering a movie in which the object is left to spin, it is desirable to fix the virtual frame interval to match that of the final movie, which is presumed to play at a constant rate:

```
> anim dt 0.0333
```

This will force the viewer to behave as if it were rendering at exactly 30 frames per second.

Similarly, automatic time-stepping will take effect at the end of each rendered frame, regardless of how long it took to render the frame. Issuing the command:

```
> anim step 0.0333
```

will force the viewer to select frames as if it were running at exactly 30 frames per second, repeating a step, or skipping steps as needed.

To quit the program, enter 'q' at the prompt. To quit without confirmation, enter a 'k'.

-----