stviz-animate Technical Documentation

Last updated: 2026-01-17

Table of contents

Overview

stviz-animate is a desktop viewer and animator for spatial transcriptomics datasets. It loads `.stviz` files, renders point clouds on the GPU, and provides a timeline for animating transitions between embedding spaces. It supports categorical, continuous, and gene-based coloring, sample grid visualization, and both screenshot and loop video export.

Key runtime goals:

- Fast GPU point rendering for large datasets.
- Predictable timeline playback and export.
- Lightweight on-disk format with memory mapping.

Repository layout

- `src/` Rust application code.
  - `app.rs`: UI, state, timeline, export logic, and main app orchestration.
  - `render.rs`: WGPU pipeline and point cloud rendering.
  - `data.rs`: `.stviz` loader, memory mapping, metadata parsing.
  - `camera.rs`: 2D camera controls (pan, zoom, fit bbox).
  - `color.rs`: color utilities and palettes.
  - `main.rs`: app entrypoint, icon loading.
- `python/` conversion and export helpers.

    - `export_stviz.py`: `.h5ad` to `.stviz` exporter.
    - `mock_spatial_dataset.py`: mock dataset generator.
    - `export_video_cv2.py`: OpenCV fallback for video export.
- `assets/`: shaders, icons, and static resources.
- `scripts/`: packaging scripts for Windows, macOS, and Ubuntu.
- `docs/`: technical documentation (this file).
- `output/`: runtime output (screenshots, exports, logs).

Build and runtime

Build and run (debug):

```
cargo run
```

Release build:

```
cargo run --release
```

Dependencies:

- Rust toolchain.
- GPU drivers for accelerated rendering.
- Python 3.8+ for conversion and OpenCV fallback export (auto-venv created).

Data format (.stviz)

`.stviz` is a custom binary format designed for fast memory mapping and low overhead parsing.

File layout:

- Bytes 0..7: little-endian `u64` JSON metadata length.
- Bytes 8..(8+json_len-1): JSON metadata (`StvizMeta`).
- Padding to 16-byte alignment.
- Binary data blocks (spaces, obs, expression arrays) referenced by offsets in metadata.

Metadata (`StvizMeta`):

- `version`: format version.
- `n_points`: number of cells/points.
- `spaces`: list of 2D spaces with offsets and bounding boxes.
- `obs`: list of categorical/continuous annotation columns.
- `expr`: optional CSC gene expression metadata.

Space metadata (`SpaceMeta`):

- `name`: space identifier (e.g., `spatial`, `X_umap`).
- `dims`: dimensionality (currently 2D).

- `offset` + `len_bytes`: byte range for packed `f32` positions.
- `bbox`: `[min_x, min_y, max_x, max_y]`.

Obs metadata (`ObsMeta`):

- Categorical: `u32` labels array + category names + optional RGBA palette.
- Continuous: `f32` values array.

Expression metadata (`ExprMeta`):

- Compressed Sparse Column (CSC) matrices.
- `indptr`, `indices`, `data` arrays and `var_names` for genes.

`data.rs` maps the file into memory, validates alignment, and provides zero-copy slices.

Data conversion pipeline

`.h5ad` to `.stviz` (`python/export_stviz.py`)

- Reads `AnnData` via `anndata`.
- Extracts 2D spaces from `.obsm` (e.g., `X_umap`, `X_tsne`, `spatial`).
- Falls back to `obs` columns (`x/y`, `spatial_x/spatial_y`, etc.) if needed.
- Collects categorical obs (categories + labels + optional scanpy palette).
- Collects continuous obs (float values).
- Optionally exports gene expression as CSC for fast gene lookup.

Mock dataset generation (`python/mock_spatial_dataset.py`)

- Generates synthetic spatial and embedding spaces.
- Adds multiple categorical and continuous features.
- Used to test workflows and GPU performance.

Video export fallback (`python/export_video_cv2.py`)

- OpenCV-based MP4 encoding when ffmpeg is not available.
- Executed inside a managed venv to reduce system dependency friction.

Rendering architecture

Rendering is GPU-first using WGPU and a point-sprite shader.

Key components:

- `PointCloudGpu` manages GPU buffers, bind groups, and pipelines.
- `Uniforms` contain camera transforms, interpolations, and point size.
- `assets/pointcloud.wgsl` renders instanced quads with a circular mask.

Uniforms (core fields):

- `viewport_px`: viewport size in pixels.
- `center`, `pixels_per_unit`: camera transform.
- `t`: timeline interpolation factor.

- `point_radius_px`: point size in pixels.
- `from_center/from_scale`, `to_center/to_scale`: per-space transforms.
- `mask_mode`: toggles round vs square points.

Offscreen render (screenshots and high-quality export):

- Uses MSAA (4x) for anti-aliasing of points.
- Renders into a multisampled texture and resolves to a single-sample texture for PNG output.

UI architecture and state

`StvizApp` holds all UI state:

- Dataset state (loaded dataset, path, ids).
- Timeline state (keyframes, times, ease, playback).
- Rendering state (camera, point size, shaders, buffers).
- Export state (fps, duration, quality, logs, cancellation).
- Sample grid state and filters.
- Advanced timeline state (cards, connections, selection).

UI layout:

- Left panel: conversion, dataset controls, color/filter controls, export controls.
- Central panel: viewport rendering.
- Bottom panel: timeline and playback controls.
- Separate viewport: advanced timeline canvas.

Viewport interactions:

- Pan by dragging.
- Zoom by mouse wheel.
- Reset scaling via timeline controls.

Timeline system

The timeline controls transitions between spaces and colors.

Core structures:

- `space_path`: list of space indices (keyframes).
- `color_path`: list of color keys (categorical/continuous/gene).
- `key_times`: normalized time positions for each keyframe.
- `ease_mode`: linear, smoothstep, sine, or quad.

Playback:

- `speed`: units per second.
- `playback_mode`: Once, Loop, PingPong.
- Keyboard controls for play/pause and frame stepping.

Advanced timeline

The advanced timeline is a separate viewport window with a canvas UI.

Features:

- Card-based timeline with draggable cards.
- Grid mode (snap cards to fixed grid size).
- Card inspector panel with space and color selection.
- Per-card duration controls.
- Multi-selection via marquee selection.
- Connections between cards (inputs/outputs).
- Preview mode (isolates the selected card when not playing).

Cards can include optional filters to display only subsets of cells for that segment.

Color modes and filtering

Color modes:

- Categorical: per-category palette with optional overrides.
- Continuous: gradient palette based on numeric values.
- Gene: gene expression coloring (if `expr` is present).

Filtering:

- Filters are applied immediately when categories are toggled.
- If all categories are disabled, nothing is drawn (empty draw list).
- Filters are reflected in draw indices, downsampling, and sample grid layout.

Sample grid system

The sample grid arranges spatial samples into a regular grid:

- Enabled in the left panel.
- Uses a categorical field (default `sample`) to group points.
- Optionally respects filters.

Grid layout:

- Computes per-category bounding boxes in the selected space.
- Places categories into a grid with adjustable padding.
- Supports labels, including custom label overrides.

Labels:

- Optional overlay showing group labels above each grid tile.
- Uses contrasting color relative to the background.

Export pipeline

Screenshots

- Always rendered at 4K (3840x2160).

- Uses offscreen GPU rendering with MSAA.
- Output path: `output/stviz-animate_screenshot_*.png`.
- PNG encoding is lossless.

Loop video export

Pipeline:

1. Render PNG frames into `output/loop_<timestamp>/`.
2. Encode into MP4 using ffmpeg (preferred) or OpenCV (fallback).
3. Optionally delete frames after encoding.

Quality options:

- Render resolution: Current viewport, Full HD, 4K.
- Encoding presets:
  - Standard: CRF 23, preset medium, yuv420p.
  - High: CRF 18, preset slow, yuv420p.
  - Ultra: CRF 14, preset slow, yuv420p.

Export controls:

- Cancel export stops rendering and deletes partial output.
- Export log is stored in `output/export_log_*.txt` and visible in the UI.

Output and logs

- Conversion outputs: `output/*.stviz`.
- Mock datasets: `output/mock_spatial_*.h5ad` (temporary).
- Videos: `output/*.mp4`.
- Screenshots: `output/stviz-animate_screenshot_*.png`.
- Logs: `output/convert_log_*.txt`, `output/export_log_*.txt`.

Packaging and distribution

Scripts:

- Windows: `scripts/package_windows.ps1`
- macOS: `scripts/package_mac.sh`
- Ubuntu: `scripts/package_ubuntu.sh`

Bundle notes:

- macOS bundle includes `python/` under `Contents/Resources`.
- Linux packages expect `python/` under `/usr/share/stviz-animate/python`.
- Windows bundle can ship `ffmpeg.exe` alongside the executable.

Configuration and environment

Python selection:

- Prefers active venv/conda, then system `python` / `python3`.
- Creates `.stviz_venv` for converter dependencies.

Environment variables:

- `STVIZ_FFMPEG`: override ffmpeg path.

GPU selection:

- Uses WGPU default adapter.
- CPU adapters enable fast render mode by default.

Troubleshooting

- ffmpeg missing: ensure `ffmpeg` is on PATH or bundle it.
- Python missing: install Python 3.8+ and add to PATH.
- Conversion failures: check the Conversion log for dependency errors.
- Export artifacts: verify PNG frames to isolate encoding artifacts.
- WSL GPU: performance may be limited vs native Windows/Linux.

Extension points

Common areas to modify:

- Add new export options: `start_export_loop` and `finish_export_loop` in `src/app.rs`.
- Add new color modes: `ColorMode` enum and `colors_for_key` in `src/app.rs`.
- Modify shader rendering: `assets/pointcloud.wgsl`.
- Adjust file format: `python/export_stviz.py` and `src/data.rs`.