svg-animate

Animated SVG diagrams with Ti*k*Z

Sébastien Gross

https://github.com/renard/svg-animate

2026/04/28— v1.1

Contents

1 Introduction **1**

2 Installation **2**

3 Compilation pipeline **2**
3.1 Embedding bitmap images . . . 2

4 Animation **3**
4.1 Overview . . . 3
4.2 Environment . . . 3
4.3 Commands . . . 4
4.4 Keys . . . 4
4.5 Static PDF output . . . 7

5 Example: traffic light **7**

6 Change log **9**

1 Introduction

`svg-animate` is a small LaTeX package for producing **animated SVG diagrams** using Ti*k*Z. The animation model is deliberately simple: content is organised into discrete *steps*, and only one step is fully visible at a time — exactly like a cinema film or a slide presentation.

The package wraps the low-level `tikz animations` library (available since PGF 3.1.9) with a convenient high-level interface that handles step counting, absolute timing, and option cascading automatically.

Output formats:

- **Animated SVG** — compile with `latex` + `dvisvgm`. The animation uses SMIL opacity keyframes supported by all major browsers.
- **Static PDF** — compile with `xelatex` or `pdflatex`. The package detects the output mode and adapts `\reveal` behaviour accordingly (see Section 4.5).

2 Installation

Copy the `svg-animate` directory into a location where TEX can find it. Common user-local locations (no root access required):

- `$TEXMFHOME/tex/latex/`
- `$TEXHOME/tex/latex/`
- `~/texmf/tex/latex/` (typical Linux default)
- `~/Library/texmf/tex/latex/` (typical macOS default)

Alternatively, point `TEXINPUTS` at the directory containing `svg-animate.sty` before compiling:

```
TEXINPUTS="path/to/svg-animate//:$TEXINPUTS" latex diagram.tex
```

The trailing `//` tells TEX to search subdirectories recursively.

3 Compilation pipeline

```
# Compile to animated SVG
latex      diagram.tex
latex      diagram.tex          # second pass for stable layout
dvisvgm --font-format=woff2 \
        --optimize=all      \
        --bbox=min          \
        diagram.dvi -o diagram.svg

# Compile to static PDF (for preview)
xelatex diagram.tex
```

3.1 Embedding bitmap images

If the diagram includes PNG or JPEG images (e.g. via `\includegraphics`), pass `-embed-bitmaps` to `dvisvgm` to produce a fully self-contained SVG. Each bitmap is then inlined as a base64 data URI rather than an external file reference, so the SVG can be moved or published independently of the source directory.

```
dvisvgm --font-format=woff2 --optimize=all --bbox=min \
        --embed-bitmaps diagram.dvi -o diagram.svg
```

Why a driver override is needed. The stock `graphicx` dvisvgm driver (`graphics-def/dvisvgm.def`) implements bitmap inclusion via `\special{dvisvgm:raw}`, injecting a literal `<image xlink:href="file.png"/` fragment. `dvisvgm` copies this verbatim into the output SVG, leaving an external reference that `-embed-bitmaps` does not process.

`svg-animate` therefore redefines `\Ginclude@bitmap` (in SVG mode only) to emit `\special{dvisvgm:img W` instead. This is the special type that `-embed-bitmaps` does process, converting the file to a base64 data URI.

Version requirement. Support for `-embed-bitmaps` on `dvisvgm:img` specials was introduced in **dvisvgm 3.4** (2024-07-24, shipped with TEX Live 2025). With earlier versions, `dvisvgm:img` produces an external `xlink:href` reference regardless of `-embed-bitmaps`: the override has no effect and bitmaps remain as separate files.

The package detects the output mode via `\pdfoutput` (defined by pdfTeX and LuaTeX):

- `\pdfoutput = 0` (**latex** in DVI mode) — loads the **dvisvgm** PGF driver and passes `dvisvgm` to `graphicx`.
- `\pdfoutput = 1` (**pdflatex**, **lualatex**) — default PDF drivers; no action needed.
- `\pdfoutput` undefined (**xelatex**) — default XeTeX drivers; no action needed.

A minimal document looks like:

```
\documentclass{standalone}
\usepackage{svg-animate}

\begin{document}
\begin{tikzpicture}
  % ... your diagram ...
\end{tikzpicture}
\end{document}
```

4 Animation

4.1 Overview

All animation content is placed inside an **animate** environment. The body is divided into *steps* by **\animstep** separators. Each **\reveal** call wraps a TikZ element and controls its opacity: the element is active (fully visible) during its step and inactive (hidden or dimmed) during all other steps.

```
\begin{animate}[duration=1]
  \reveal[inactive opacity=0.2]{\node (A) at (0,0) {Step 1};}
  \animstep
  \reveal{\node (B) at (0,0) {Step 2};}
  \animstep[duration=2]
  \reveal{\node (C) at (0,0) {Step 3 (lasts 2 s)};}
\end{animate}
```

Options cascade from the environment down through each step to individual elements: inner options override outer ones.

```
\begin{animate}[duration=1, inactive opacity=0]
\animstep[inactive opacity=0.3]
\reveal[active opacity=0.8]{...}
\end{animate}
```

4.2 Environment

\begin{animate}[⟨*options*⟩]
  ⟨*environment content*⟩
\end{animate}

Collects all content up to **\end{animate}** and processes it in two passes: first to sum all step durations (without executing any TikZ code), then to render each step with correct absolute timing.

[⟨*options*⟩] accepts any **/anim/.cd** key and applies them as defaults for all steps in this environment.

4.3 Commands

`\reveal`[⟨*options*⟩]{⟨*content*⟩}

Wraps ⟨*content*⟩ in an opacity animation. The element is shown at `active opacity` during its step and at `inactive opacity` at all other times.
Set `inactive opacity` to a value above 0 to *dim* the element instead of hiding it — useful for nodes that should remain faintly visible in the background.
`duration` in [⟨*options*⟩] is accepted but silently ignored (it has no meaning at the per-element level).
PDF mode behaviour depends on whether `\noanimate` is present in the enclosing `animate` body:

- No `\noanimate`: ⟨*content*⟩ is rendered at full opacity (all steps stacked — useful for a quick overview).
- `\noanimate` present: ⟨*content*⟩ is suppressed; use the `noanimate` key on `\reveal` to override for individual elements.

`\noanimate`{⟨*content*⟩}

PDF mode: renders ⟨*content*⟩ as a plain static element, with no animation wrapper. When `\noanimate` is present inside an `animate` body, all `\reveal` elements in that body are automatically suppressed, leaving ⟨*content*⟩ as the sole visible output — a single clean frame.
SVG mode: completely ignored; the animated steps play normally.
`\noanimate` must appear inside `animate`. Multiple calls are allowed; each one renders independently in PDF.

```
\begin{animate}[inactive opacity=0.08]
  \noanimate{\fill[red!85!black] (0,2.2) circle[radius=0.85];}
  \reveal{\fill[green!65!black]  (0,-2.2) circle[radius=0.85];}
  \animstep
  \reveal{\fill[red!85!black]   (0, 2.2) circle[radius=0.85];}
\end{animate}
```

`\animstep`[⟨*options*⟩]

Step separator inside `animate`. The token is never executed; it is consumed as a delimiter when the body is split into per-step segments.
[⟨*options*⟩] accepts any `/anim/.cd` key and applies them to all elements of the *following* step, overriding environment-level defaults.

4.4 Keys

All keys live under the `/anim/.cd` path and can be set at any level of the cascade.

`/anim/duration`=⟨*seconds*⟩ (initial: `2`)

Duration of this step in seconds. Can be set globally, per-environment, or per-step via `\animstep`[⟨*duration*=⟨*N*⟩⟩]. Has no effect when passed to `\reveal` (silently ignored).

`/anim/active opacity`=⟨*value*⟩ (initial: `1`)

Opacity of a `\reveal` element while its step is active. `1` = fully opaque (default), `0` = fully transparent.

`/anim/inactive opacity`=⟨*value*⟩ (initial: `0`)

Opacity of a `\reveal` element while its step is *inactive*. `0` = fully hidden (default). Set to a small positive value (e.g. `0.15`) to keep elements faintly visible rather than completely disappearing.

`/anim/loop=true|false` (initial: `true`)

Controls whether the animation repeats after reaching the last step.

- `true` (default) the animation loops indefinitely.
- `false` the animation plays once and freezes on the last frame.

Can be set globally or per-environment.

```
%% One-shot animation  plays through all steps once, then stops.
\begin{animate}[loop=false]
  \reveal{\node {Step 1};}
  \animstep
  \reveal{\node {Step 2};}
  \animstep
  \reveal{\node {Step 3};}
\end{animate}
```

Interaction with `\noanimate`: When the animation ends (`loop=false`), the SVG browser removes all `<animate>` effects and reverts each element to its base (unanimated) opacity — which for `\reveal` elements is 1, i.e. fully visible. If a `\noanimate` fallback is present in the body, it therefore becomes visible once the animation has finished, giving the diagram a natural static resting state.

```
\begin{animate}[inactive opacity=0.08, loop=false]
  %% Static resting state shown after the animation ends.
  \noanimate{\fill[red!85!black] (0,2.2) circle[radius=0.85];}
  \reveal{\fill[green!65!black]  (0,-2.2) circle[radius=0.85];}
  \animstep
  \reveal{\fill[orange!90!black] (0, 0.0) circle[radius=0.85];}
  \animstep
  \reveal{\fill[red!85!black]    (0, 2.2) circle[radius=0.85];}
\end{animate}
```

`/anim/blink=`⟨*seconds*⟩ (no default)

Makes the element blink during its active step: the opacity alternates between `blink on opacity` and `blink off opacity` at the given period. The element is shown at `inactive opacity` during all other steps.
The ⟨*seconds*⟩ value is the *full* blink period: the element is visible for ⟨*seconds*⟩/2 seconds, then hidden for ⟨*seconds*⟩/2 seconds, then repeats.
If the step duration is not an exact multiple of ⟨*seconds*⟩/2, the last partial half-period is included and cut at the step boundary.
Note: `blink` and `step` are mutually exclusive. When both are specified, `blink` wins, `step` is ignored, and a `PackageWarning` is issued.

```
\begin{animate}[duration=2, inactive opacity=0.15]
  \reveal{\node at (0,1) {Step 1  static};}
  \animstep
  \reveal[blink=0.4]{\node[fill=yellow] at (0,0) {Step 2  blinks};}
  \animstep
  \reveal{\node at (0,-1) {Step 3  static};}
\end{animate}
```

/anim/blink on opacity=⟨*value*⟩ (initial: 1)

Opacity used during the *"on"* (visible) half-periods of a blink. Defaults to 1 (fully opaque). Independent of **active opacity**.

/anim/blink off opacity=⟨*value*⟩ (initial: 0)

Opacity used during the *"off"* half-periods of the blink oscillation *within the active step*. Defaults to 0 (fully hidden).
Between steps the element uses **inactive opacity** as usual **blink off opacity** has no effect outside the active step. To hide a blink element between steps, set **inactive opacity=0** explicitly; the **static** key on the **animate** environment ensures this does not suppress the element in the static display:

```
%% Hidden between steps; blinks between 1 (on) and 0.25 (off) during step 2.
\begin{animate}[duration=2]
  \reveal{\node at (0,1) {Step 1};}
  \animstep
  \reveal[blink=0.4, inactive opacity=0, blink off opacity=0.25]{%
    \node[fill=yellow] at (0,0) {Step 2  dimmed blink};}
  \animstep
  \reveal{\node at (0,-1) {Step 3};}
\end{animate}
```

/anim/static(no value) (default: not set)

Renders all **\reveal** elements at their natural (full) opacity without emitting any SMIL keyframe animation. Equivalent to drawing all steps simultaneously, which is useful for printed handouts and static overviews.
This key supersedes the older idiom of setting **active opacity=1, inactive opacity=1, blink on opacity=1, blink off opacity=1** at the environment level, which failed whenever an element carried an explicit element-level override.

```
\begin{animate}[static]
  \stepOne \animstep \stepTwo \animstep \stepThree
\end{animate}
```

/anim/step=⟨*spec*⟩ (no default)

By default **\reveal** makes an element visible only during the *current* step (the step in which it appears in the source). The **step** key overrides this: the element becomes visible during every step listed in ⟨*spec*⟩, regardless of its position in the source.

Syntax of ⟨*spec*⟩:

- Single step **step=2**
- Range **step=2-5** (steps 2, 3, 4, 5)
- List **step={1,3}** (braces required when listing multiple values)
- Mix **step={1,3-5}** (braces required)

Consecutive steps in the specification are automatically merged into a single animation window (no redundant keyframe at the shared boundary).

```
\begin{animate}[inactive opacity=0.08]
  \reveal[step={1,3}]{\node[red]  at (4,1) {visible in steps 1 and 3};}
  \reveal[step=2-4]{  \node[blue] at (4,0) {visible in steps 2 through 4};}
  \reveal[step={1,3-5}]{\node    at (4,-1) {steps 1, 3, 4, 5};}
\end{animate}
```

`/anim/noanimate` (no value) (default: not set)

When passed to `\reveal`, forces that element to render in PDF mode even when `\noanimate` is present in the enclosing `animate` body. Has no effect in SVG mode. Useful for elements that should appear in both the animated SVG *and* the static PDF fallback — for example a transition light that belongs to multiple steps and should also be visible in the PDF snapshot.

```
\begin{animate}[inactive opacity=0.08]
  \noanimate{\fill[red!85!black] (0,2.2) circle[radius=0.85];}
  \reveal{\fill[green!65!black]  (0,-2.2) circle[radius=0.85];}
  \animstep
  %% amber is both animated and shown in the PDF static frame
  \reveal[noanimate]{\fill[orange!90!black] (0,0.0) circle[radius=0.85];}
\end{animate}
```

4.5 Static PDF output

The package exposes the boolean flag `\if@anim@svgmode`, which is `true` only when compiling with `latex` in DVI mode (the SVG pipeline). `\reveal` and `\noanimate` both query this flag; the logic is summarised below.

	SVG	PDF
`\reveal{...}` (no `\noanimate` in body)	animated	rendered
`\reveal{...}` (`\noanimate` present)	animated	suppressed
`\reveal[noanimate]{...}`	animated	rendered
`\noanimate{...}`	ignored	rendered

The `animate` environment scans its collected body for the token `\noanimate` before executing any TikZ code, so the suppression decision is made once per environment instance and does not require two separate document structures.

5 Example: traffic light

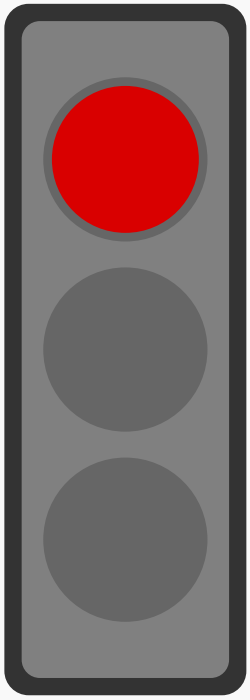
The diagram animates a traffic light through four steps. The housing and lens covers are static (always visible at full opacity). The lights are revealed one at a time via `\reveal`; when inactive their opacity drops to `0.08`, simulating an unlit bulb seen through dark glass.

The sequence demonstrates two features: per-step duration (step 2 lasts only `0.5 s`) and multiple `\reveal` calls in a single step (step 2 shows green and amber simultaneously as a transition):

Step	Lights	Duration	Meaning
1	green	1.5 s	go
2	green + amber	0.5 s	prepare to stop
3	amber	1.0 s	caution
4	red	1.5 s	stop

The box below shows the complete `svg-animate-example.tex` source on the left and the compiled PDF on the right. The `\noanimate` call selects the red light as the static PDF frame; open `svg-animate-example.svg` in a browser for the full animation.

```
\documentclass{standalone}
\usepackage{svg-animate}

\begin{document}
\begin{tikzpicture}

  \tikzset{/anim/duration=1.5}

  %% Housing (static  always visible)
  \fill[black!80, rounded corners=8pt]  (-1.4, -4.0)
    rectangle (1.4, 4.0);
  \fill[black!50, rounded corners=6pt]  (-1.2, -3.8)
    rectangle (1.2, 3.8);

  %% Lens covers (static dark glass behind each bulb)
  \foreach \y in {2.2, 0, -2.2} {
    \fill[black!60] (0, \y) circle[radius=0.95];
  }


  %% The three lights, one per animation step.
  %% When inactive, opacity=0.08 simulates an unlit bulb
    seen through dark glass.
  \begin{animate}[inactive opacity=0.08]
    %% Static PDF fallback: show only the red light (one
    clean frame).
    %% Ignored in SVG mode; in PDF mode \reveal draws
    nothing so only this appears.
    \noanimate{\fill[red!85!black] (0, 2.2)
    circle[radius=0.85];}

    %% Step 1 -- green (1.5 s)
    \reveal{\fill[green!65!black]  (0, -2.2)
    circle[radius=0.85];}

    \animstep[duration=0.5]
    %% Step 2 -- green+amber transition (0.5 s)
    \reveal{\fill[green!65!black]  (0, -2.2)
    circle[radius=0.85];}
    \reveal[blink=0.2]{\fill[orange!90!black] (0,  0.0)
    circle[radius=0.85];}

    \animstep[duration=1.0]
    %% Step 3 -- amber (1.0 s)
    \reveal{\fill[orange!90!black] (0,  0.0)
    circle[radius=0.85];}

    \animstep
    %% Step 4 -- red (1.5 s)
    \reveal{\fill[red!85!black]    (0,  2.2)
    circle[radius=0.85];}

  \end{animate}

\end{tikzpicture}
\end{document}
```



6 Change log

`v1.1` (2026/04/28)

- Added installation instructions (user-local `texmf` paths and `TEXINPUTS`).
- Rewrote the `\Ginclude@bitmap` override to detect the `dvisvgm` version at compile time: the override is now only applied with `dvisvgm` ≥ 3.4, which is the version that added `-embed-bitmaps` support for `dvisvgm:img` specials.
- Renamed `example.tex` and `test.tex` to `svg-animate-example.tex` and `svg-animate-test.tex` to avoid filename collisions on shared TEX search paths (CTAN requirement).

`v1.0` (2026/03/18)

- Initial CTAN release.