Behind the scenes of the Great Ti*k*Zlings Christmas Extravaganza

samcarter, Gert Fischer

"O, wonder! How many goodly creatures are there here."
(William Shakespeare, The Tempest, V, 1)

Abstract

The Great Ti*k*Zlings Christmas Extravaganza is an annual video series that utilises LaTeX to produce animated films. This proceeding will offer a look behind the scenes of the Extravaganza and explain how we use LaTeX to create the videos.

1 The Rise of the Ti*k*Zling

In 2005 Till Tantau was about to publish his new tool to create graphic elements in LaTeX and was looking for a name. He turned to a recursive acronym and chose "Ti*k*Z ist kein Zeichenprogramm", in English: "Ti*k*Z is not a drawing program". This might be understood as a cautioning of users not to expect too much. The instant success and the usefulness of the package and its updates till today show that this probably was too modest a name. Also, as in all great creations, there were hidden dimensions the creator had not intentionally included and could not and did not foresee.

So Ti*k*Z too had its Goedelian moments. One came when samcarter was probably the first human to discover that there were animals in Ti*k*Z. In 2017 she found a duck. And in quick succession other creatures were discovered. With the help of Paulo Cereda, Ulrike Fischer, Carla Maggi and many others, samcarter listed and categorised them and in a second step supplied them with clothes and other items helping them to feel comfortable at the fringes of Ti*k*Z. The question of how to name the newly discovered was put to the TikZ community and resolved democratically. In an open ballot the members voted almost unanimously for Stefan Kottwitz's proposal. From now on there would be "Ti*k*Zlings". By 2018 the Ti*k*Zlings ecosystem had been thoroughly documented on GitHub [3] in a flexible grid allowing the addition of hitherto unknown species.

While this important work was going on, some of samcarter's collaborators decided in 2017 to honour her efforts with a special Christmas present. They took it upon them to teach some of the Ti*k*Zducks to dance and sing. As the following quotes from their e-mail exchange show, Paulo Cereda, Gert Fischer, Ulrike Fischer and Carla Maggi were in deep water at once:

- "never tried it before"
- "I tried something but it doesn't work"
- "I have almost no experience with videos"
- "when everything fails, I'll record my own screen"
- "the arara rule doesn't work"
- "it's the first time I use github"

But somehow it worked. At the end the resulting memes were merged into a video film under the title "The Great Ti*k*Zducks Christmas Extravaganza". Samcarter loved it!

From then on there was no looking back. Samcarter joined the team bringing in the expertise so sorely lacking in the first Extravaganza. Two other additions were Professore Paulinho van Duck of Sempione Park University, Milano, Italy, and Bär who came in as assistant to the producers and fashion consultant to the Ti*k*Zlings cast [1]. And the Extravaganza was indeed there to stay. The Ti*k*Zducks Extravaganza became the Ti*k*Zlings Extravaganza which since 2018 has appeared every year in December, sometimes with the additional help of non-permanent team members such as Marmot and Plergux. All editions can be viewed on the Internet [2]. Over the years the Extravaganza has gained a group of fervent supporters and it has also made its impact on TUG, where the outgoing president Boris Veytsman has been recommending it to the members from the start.

Having said as much, a view behind the scenes proves that Christmas preparations require some programming skills — or as The Bard has it:
"Though this be madness, yet there is method in't" (Hamlet, Prince of Denmark, II, 2).

2 Creating multi-page PDFs

The basic idea of creating animated films with LaTeX is to first create a multi-page PDF with incremental motion between each page, which then gets converted into a movie.

There are many possible ways of using LaTeX to create a multi-page PDF. For many of our scenes, we use a combination of beamer and Ti*k*Z.

The overlay mechanism of beamer is used to repeat the animation on all pages, and Ti*k*Z makes it easy to position individual elements on a page.

The following code block shows a shortened example of how this setup can be used to let a penguin move across a page:

```
\documentclass{beamer}
\usepackage{tikz}
\usetikzlibrary{tikzlings}
\begin{document}

\begin{frame}
```

doi.org/10.47397/tb/44-2/tb137samcarter-tikzlings

```
  \begin{tikzpicture}[remember picture,overlay]
    \foreach \macro in {1,...,5}{
        \path<+>[fill=white]
           (2*\macro,-0.7*\macro) pic{penguin};
    }
  \end{tikzpicture}
\end{frame}
\end{document}
```

This code produces a PDF with five pages. In Figure 1 the five pages are stacked on top of each other with different levels of transparency to visualise the movement of the penguin.

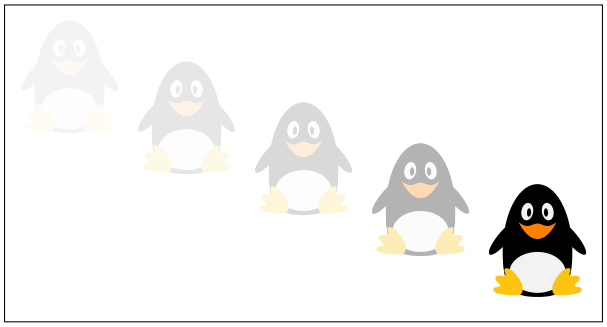

Figure 1: Visualisation of a penguin moving across a page. The different opacity levels denote the different pages stacked on top of each other. (Grayscaled for print.)

3 Converting PDFs into videos

Converting a PDF into a video consists of multiple steps. The first step is converting each page of the PDF into a raster image. We are using PNG as image format as many of the protagonists in our videos consist of simple geometric shapes with clear edges and PNG is a format which will preserve these edges and won't introduce image artefacts.

There are many tools available to convert PDFs into PNGs. Currently, our tool of choice is `pdftoppm`, which is part of the Poppler[1] package:

```
pdftoppm -png -r 240 Example.pdf Example
```

Compared to the widely used `convert` command from ImageMagick, we found that `pdftoppm` is faster in converting the PDF and lighter on CPU usage.

The next step is to use FFmpeg[2] to assemble the individual images into a video and combine it with suitable music:

```
ffmpeg \
  -ss 00:00:00 -i Example-%03d.png \
  -ss 00:00:10 -i Music.m4a \
  -shortest \
  Example_raw.mp4
```

We sometimes encountered problems with videos not working in some multimedia players, so we now preventively run the resulting video through HandBrake:[3]

```
HandBrakeCLI --crop 0:0:0:0 \
  -i Example_raw.mp4 -o Example.mp4
```

This leaves us with one video clip for each scene and each intermission, which need to be combined into a single video. To make the resulting video more pleasant to watch, transitions between the individual videos have to be inserted and the volume level of some of the clips needs to be adjusted to fit in with the others.

In the past, we used video editing applications like iMovie (macOS) to combine the videos. They usually required some level of manual interaction, e.g. dragging videos into the correct order, clipping them if necessary, etc. This turned out to be error-prone and could also be frustrating if there were last-minute changes to one of the clips which required re-rendering the whole video.

Our current solution is to script the whole process using `moviepy`,[4] a python library for video editing. An abbreviated example of our merge script is shown in the following:

```
################################################
# stitching together the videos
# and adding transitions
################################################
from moviepy.editor import *
import os

# duration of transitions between the videos
padding = 1.5

video_clips = [
VideoFileClip("../intermissions/title.mp4"),
VideoFileClip("../intermissions/example.mp4"),
VideoFileClip("../example-scene/Example.mp4")
↪ .volumex(1.1),
VideoFileClip("../intermissions/credits.mp4"),
]

################################################
# merge title and first intermission to get
# continious audio
################################################
video_fx_list = []
idx = 0
for video in video_clips[0:2]:
    video_fx_list.append(video.set_start(idx)
```

[1] poppler.freedesktop.org/
[2] ffmpeg.org/
[3] handbrake.fr/
[4] zulko.github.io/moviepy/

```
    ↪ .crossfadein(padding)
    ↪ .crossfadeout(padding)
    ↪ .audio_fadein(padding)
    ↪ .audio_fadeout(padding))
    idx += video.duration - padding

merged_video =
↪ CompositeVideoClip(video_fx_list)
duration = merged_video.end

audioclip = AudioFileClip("../intermissions/
↪ JingleBells.m4a").subclip(2,duration+2)
merged_video =
↪ merged_video.set_audio(audioclip)

###############################################
# adding rest of the scenes
###############################################
video_fx_list = []
idx = 0

# adding the merged video first
video_fx_list.append(merged_video
↪ .set_start(idx).crossfadein(padding)
↪ .crossfadeout(padding).audio_fadein(padding)
↪ .audio_fadeout(padding))
idx += merged_video.duration - padding

# rest of the videos
for video in video_clips[2:]:
    video_fx_list.append(video.set_start(idx)
    ↪ .crossfadein(padding)
    ↪ .crossfadeout(padding)
    ↪ .audio_fadein(padding)
    ↪ .audio_fadeout(padding))
    idx += video.duration - padding

final_video = CompositeVideoClip(video_fx_list)
final_video.write_videofile
↪ ("Extravaganza_raw.mp4")
```

The full version of this `moviepy` script is available from `github.com/TikZlings/Extravaganza2022/blob/main/videos/merge_videos.py`.

The final step is another pass through HandBrake. This will ensure that the video is properly encoded and all potential issues are fixed. It also reduces the file size quite a bit.

4 Watch the videos

The finished video gets published on Vimeo. If you would like to watch some of the previous videos, we collected the links to all our previous videos at `github.com/TikZlings`. This site also contains links to the source code of all the previous extravaganzas in case you are curious about how a particular clip was made.

References

[1] U. Fischer. *The Bearwear package: Shirts to dress TikZbears.* `ctan.org/pkg/bearwear`

[2] Ti*k*Zlings organisation, 2023. `github.com/TikZlings`

[3] samcarter. Ti*k*Zlings, 2023. `github.com/samcarter/tikzlings`

◇ samcarter

◇ Gert Fischer
Bonn, Germany