Manim Editor

Release v0.3.8

ManimEditorProject

Jun 25, 2022

TABLE OF CONTENTS:

Warning: This documentation is a work in progress, please tread carefully.

The `Manim Editor` is a tool for post-processing animations generated via `Manim`.

> Animating technical concepts is traditionally pretty tedious since it can be difficult to make the animations precise enough to convey them accurately. `Manim` uses Python to generate animations programmatically, making it possible to specify exactly how each one should run.
>
> —The Manim Documentation

The `Manim Editor` offers two main functions:

- Reliable web presentations.
- Automated video editing (not yet implemented).

Tip: If you want to test the presentation output of the `Manim Editor` without having to install anything, take a look at the example.

If you don't know how to use `Manim`, it's documentation is where you should start. Come back once you've familiarised yourself with the core concepts.

CHAPTER

ONE

MAIN IDEA

The `Manim Editor` is using the Manim Section API. It allows the separation of a scene into multiple sections. In addition to that it optionally stores names and types for each section. `Manim` only supports the type `DefaultSectionType.NORMAL` out of the box. The `Manim` Editor defines more types, which define how a section should be played in the presentation. They are described *here*.

One or more sections build a slide. They are equivalent to slides from PowerPoint and can be shown individually. Multiple slides (possibly build out of sections from different `Manim` scenes) build one project that can be presented as a whole. A Manim Editor project is a directory that will house everything needed to present a project. More on that *here*.

Note: The Section API is a new feature of `Manim` v0.12.0; thus you can't use any older versions than that.

1.1 Installation

Just as `Manim` the `Manim Editor` is a Python package on PyPi. If you know what you're doing, simply install it using pip. But make sure that FFmpeg is accessible through the `ffmpeg` command.

First you have to install Manim and ensure that it functions on your system.

Tip: On Windows it is recommended to use the manual installation as it simplifies the installation of the `Manim Editor` later on.

Once your Python environment is functioning and populated with `Manim` and its requirements you can run this command:

```
python -m pip install manim-editor
```

Now the `Manim Editor` is accessible through the terminal via `manim_editor` or `manedit`.

If you've gotten stuck and don't know what to do, feel free to ask on the Manim Discord Server or open an Issue on GitHub. (The latter is preferred.)

1.1.1 Working with The Manim Editor

Now you should be able to use the `Manim Editor`. Now you can *get started with the Manim Editor*.

1.2 Getting Started

- First you have to *create the Manim scene*. As *already described* the `Manim Editor` uses projects, which are stored in one directory each.
- Once you have created your scenes, you can *create a Manim Editor project*.
- Finally you can *export that project as a presentation*.
- Or you could use *the CLI* if you don't want to use a web browser.

1.2.1 Creating the Manim Scene

Basics

The first thing you have to take care of is your `Manim` scene. You have to be aware of the different types each section can have. These types are defined by the `Manim Editor` and can be imported using `from manim_editor import PresentationSectionType`.

Four types (and four secondary types) are provided:

Table 1: Types defined by the `Manim Editor`

Name	Function
`NORMAL`	start, end, wait for continuation by user
`SKIP`	start, end, immediately continue to next section
`LOOP`	start, end, restart, immediately continue to next section when continued by user
`COMPLETE_LOOP`	start, end, restart, when user continues finish animation first
`SUB_NORMAL`	same as `NORMAL` but as sub section
`SUB_SKIP`	same as `SKIP` but as sub section
`SUB_LOOP`	same as `LOOP` but as sub section
`SUB_COMPLETE_LOOP`	same as `COMPLETE_LOOP` but as sub section

They are also explained in the interactive tutorial. Sub sections are sections that don't get listed in the timeline. They belong to the slide of the last full section. If you don't need sub sections you can simply ignore this feature. When there are no sub sections, sections and slides are synonymous.

With this information you can create a `Manim` scene with the correct types like this:

```
from manim import *
from manim_editor import PresentationSectionType

class Test(Scene):
    def construct(self):
        self.next_section(type=PresentationSectionType.NORMAL)
        # play some animations...
        self.next_section("Names are still supported.", type=PresentationSectionType.
↪SKIP)
        # play more animations...
```

When you have defined your scene, you have to render it like this:

```
manim --save_sections example.py
```

It is essential that you use the `--save_sections` flag. Otherwise the `Manim Editor` won't find anything to work with.

This creates a `media` directory in the current working directory (CWD).

You can create as many scenes as your heart desires, they should only be created in the `media` directory. That way they can be used together for the same project. Once you're happy with your scene, you can *create the Manim Editor project*.

Minimal Example

You can run the following minimal example, to get your first presentation. It results in this presentation.

```
from manim import *
from manim_editor import PresentationSectionType

def make_elements():  # only setting up the mobjects
    dots = VGroup(Dot(), Dot(), Dot(), Dot(), Dot(), Dot(), Dot(), z_index=0)
    dots.arrange(buff=0.7).scale(2).set_color(BLUE)
    dots[0].set_color(ORANGE)
    dots[-1].set_color(ORANGE)
    moving_dot = Dot(color=ORANGE, z_index=1).scale(2.5)
    moving_dot.move_to(dots[0])
    path = VGroup()
    path.add_updater(lambda x: x.become(Line(dots[0], moving_dot, stroke_width=10, z_
↪index=1, color=ORANGE)))
    return dots, moving_dot, path


class MinimalPresentationExample(Scene):
    def construct(self):

        dots, moving_dot, path = make_elements()
        self.add(dots, moving_dot, path)

        self.next_section("A", PresentationSectionType.NORMAL)
        self.play(moving_dot.animate.move_to(dots[1]), rate_func=linear)

        self.next_section("A.1", PresentationSectionType.SUB_NORMAL)
        self.play(moving_dot.animate.move_to(dots[2]), rate_func=linear)

        self.next_section("B", PresentationSectionType.SKIP)
        self.play(moving_dot.animate.move_to(dots[3]), rate_func=linear)

        self.next_section("C", PresentationSectionType.LOOP)
        self.play(moving_dot.animate.move_to(dots[4]), rate_func=linear)

        self.next_section("D", PresentationSectionType.COMPLETE_LOOP)
        self.play(moving_dot.animate.move_to(dots[5]), rate_func=linear)
```

(continues on next page)

(continued from previous page)

```
        self.next_section("E", PresentationSectionType.NORMAL)
        self.play(moving_dot.animate.move_to(dots[6]), rate_func=linear)
```

1.2.2 Create a New Project

Once you have *installed the Manim Editor* you can start it by running *manedit*.

Tip: If the *manedit* command fails, try running *python3 -m manim_editor* instead.

The `Manim Editor` searches the current working directory (CWD) for `Manim` scenes which have been rendered using the `--save_sections` flag. Therefore you should run the `manedit` command where your `media` folder has been created.

That command launches a local web server on an open port. It presents an address which you can open in your browser. Here you can select any `Manim Editor` Projects you have already created.

When you create a new project, the editor asks you to select the scenes you want to be included in the new project. The order of the scenes can be adjusted with the priority; the smaller the priority the earlier that scene gets played.

1.2.3 Export Presenter

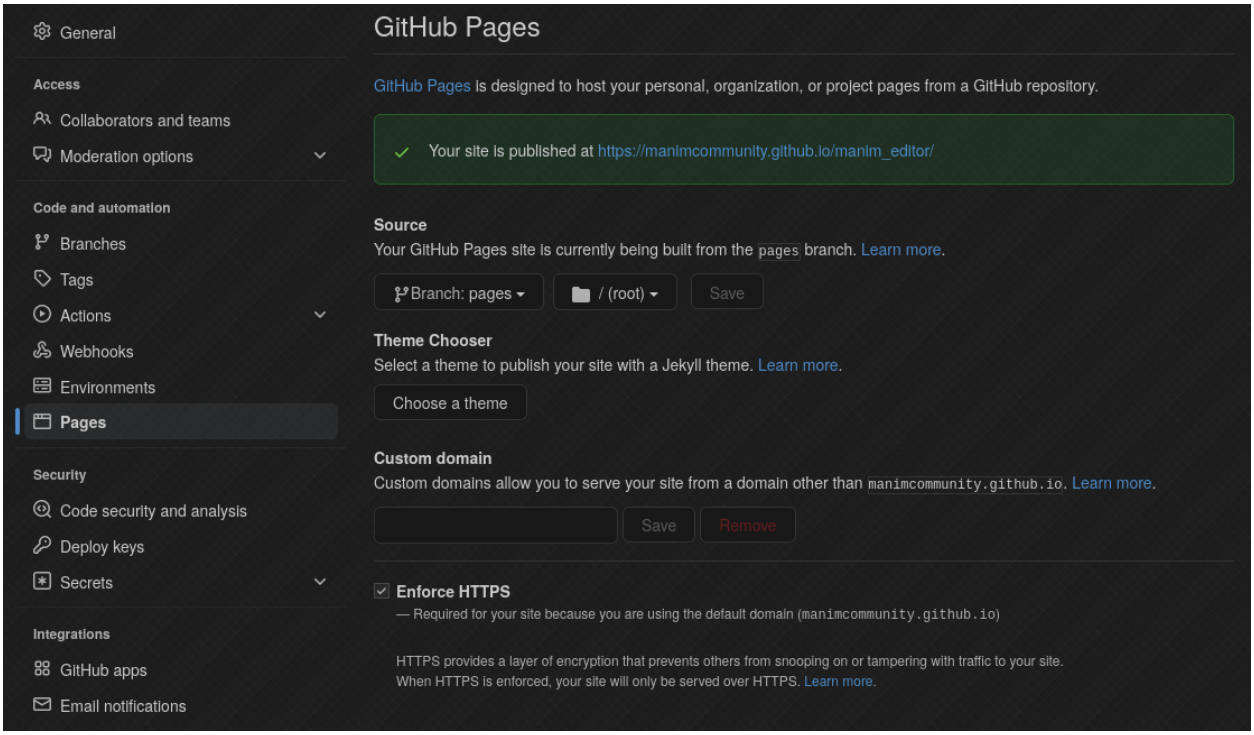
When you've created a project and you launch `manedit`, you can select the project and edit it. If everything is to your liking you can export the project as a presentation. This adds a few files required to present the project to the project directory. You can copy this directory to a flash drive or wherever you need it.

Now you don't need the `Manim Editor` to be installed anymore to be able to present the project. You only need a local web server. If you have Python installed, you're good to go. Simply run this in the projects directory:

```
python3 -m http.server
```

Instead of running a local web server you can host the project on an online web server (e.g. GitHub Pages). That way you can access it from anywhere. This is how the example has been created.

If you intend to use GitHub Pages, you have to create a GitHub repository first. All the steps you need to follow are explained in this official GitHub Pages tutorial. The tutorial diverges at one point: You don't have to create any markdown or HTML files. Instead, copy the contents of the project folder (after you've exported the presenter) into the root of the repo. Which branch you want to populate is entirely up to you. Any Python files used to create the presentation aren't needed. In the GitHub Pages settings you have to select the branch you chose (in this case `pages`) in the GitHub Pages settings. In the end you should end up with settings that look similar to these:



And the root directory directory of said branch should look like this:

```
.
├── img
│   ├── arrow_clockwise.svg
│   ├── banner.png
│   ├── favicon.png
│   ├── hourglass_split.svg
│   ├── play_btn.svg
│   └── wind.svg
├── index.html
├── project.json
├── thumb_0000.jpg
├── thumb_0001.jpg
├── thumb_0002.jpg
├── thumb_0003.jpg
├── thumb_0004.jpg
├── thumb_0005.jpg
├── video_0000.mp4
├── video_0001.mp4
├── video_0002.mp4
├── video_0003.mp4
├── video_0004.mp4
├── video_0005.mp4
└── webpack
    ├── 67475f65d9d8a1fe03a2.woff
    ├── base.css
    ├── base.js
    ├── base.js.LICENSE.txt
    ├── d0ec932c09e146590948.woff2
```

(continues on next page)

(continued from previous page)

```
    ├── edit_project.js
    └── scene_selection.js
```

If you've done all that correctly, everyone with an internet connection can access your presentation under the URL listed in the settings. Should you require multiple presentations, you can simply put them in individual subdirectories and append the subdirectory name to the url (like `https://manimcommunity.github.io/manim_editor/Tutorial` instead of `https://manimcommunity.github.io/manim_editor`).

Supported Browsers

These browsers are officially supported. Others may work as well but haven't been tested yet.

- Firefox
- Chrome
- Edge

If you confirm another functioning browser, feel free to open an issue and tell the devs.

Presenter Explanation

The presenter is separated in three parts: timeline, video player, controls and informative tables.

In the timeline you can find the name, type (which is also displayed in the tables beneath the controls) and the thumbnail of each slide. You can click on an element and it will take you to that part in the presentation. In addition to that it shows the amount of time spent playing each slide. This value will only update once a different section is being shown or the section gets restarted. That way you get accurate information on how much time you spent on each section.

Instead of using the keyboard, you can utilise the controls on the right. They offer basic media functionality like play last section, play next section, pause, restart section and enter fullscreen. Pressing the "last section" button (or using ArrowLeft on the keyboard) doesn't necessarily go back one section. If the current section has already been playing for a while, the current section will be restarted instead. You can always use the Control key (or Command on a Mac) to forcefully go to the next or last section.

If you are hosting the presenter on a remote webserver, you should consider using the "Cache Videos" button. It requests all videos and thus offers the browser the opportunity of caching them, speeding up future requests.

Note: The timeline shows slides while the controls jump from section to section. That way you can create a lot of individual sections that don't clutter the timeline.

The player settings offer some fine-grained options, most of which are for debugging purposes. Only the loader switch is of any interest: The presenter of the `Manim Editor` employs two redundant video loaders, the buffer and the fallback loader. You should always use the buffer loader. But if for any reason it doesn't work, you can use the fallback loader instead.

Warning: Be aware that the fallback loader has a detrimental effect on loading times between sections. Only use it when there is no other way!

1.2.4 Command Line Interface

Instead of using a web browser to create a project and export the presenter, you can use the CLI.

```
manedit --quick_present_export path/to/example.json
```

This command creates a new project with the name `example`, populates it with videos and thumbnails and exports the presenter. That way you immediately have the final presenter, which you can load onto a webserver.

It loads all sections referenced in the JSON file provided. These files get created by `Manim` when using the `--save_sections` flag. If you need multiple scenes to become one project, you can use the `--quick_present_export` flag multiple times:

```
manedit --quick_present_export path/to/a.json --quick_present_export path/to/b.json
```

Note: You can also define the name of the to be created project with the `--project_name` flag.

1.3 Internals

1.3.1 Build from Source

- clone repo: `git clone https://github.com/ManimEditorProject/manim_editor && cd manim_editor`
- install poetry dependencies: `poetry install`
- enter poetry shell: `poetry shell`
- install Pre-Commit: `poetry run pre-commit install`
- install npm modules: `npm ci`
- compile web files: `npm run build_debug` or `npm run build_release`
- start editor in debug mode: `manedit --debug`

1.3.2 Files to be Updated when Bumping Version

- `pyproject.toml`
- `package.json`
- `manim_editor/config.py`
- `docs/source/conf.py`

1.3.3 A Brief History of the Manim Editor

This project started in September 2021 as the Manim Web Presenter, which has been inspired by the Manim Presentation Repo. Back then `Manim` didn't have any sections, therefore this functionality had to be implemented using wrappers around the `Scene` class and subclasses. This was an ugly solution, which turned obsolete with the implementation of the Section API. Since more and more features were requested, it made sense to re-design the project from scratch and give it another name: the `Manim Editor`. The `Manim Editor` repository got created in October and was moved under the ManimCommunity namespace in November.

These people directly participated in developing the `Manim Editor` (in order of first contribution):

- Christopher Besch
- Markus Jørstad Svendsen
- Jan-Hendrik Müller