FlyingFrames

kolibril13

Oct 14, 2021

CONTENTS

Hi! FlyingFrames is a project by me (kolibril13), where I want to provide you some tutorials and code snippets that I often use when I prepare my animations for my youtube channel: https://www.youtube.com/c/kolibril It does not replace the amazing documentation at https://docs.manim.community/en/stable/ , but I hope this blog will give you some inspiration for your future projects.

CHAPTER

ONE

CONTENT

1.1 Manim in Jupyter

Working with manim in jupyter notebooks has several advantages:

- code snippets and rendered outputs are close together
- easy to iterate examples
- easy to try different varieties of one scene in multiple cells
- computation intensive code can be executed separately from the scenes
- global Mobjects can be used in multiple scenes.

1.1.1 Simple Example

First, we need to import manim

[1]:
```
from manim import *
```

```
Manim Community v0.11.0
```

Now we build up our scene

[2]:
```
%%manim -v WARNING  --progress_bar None -s -ql --disable_caching MyExample
class MyExample(Scene):
    def construct(self):
        m= ManimBanner()
        self.add(m)
```

Note, that I use the following parameters:

- `-v WARNING` means that only warnings are shown in the log
- `--progress_bar None` will not show the animation progress bar
- `-s` will only show the last frame
- `-ql` renders in low quality
- `--disable_caching` will disable the manim caching system
- `MyExample` gives the scene name

for rendering a video, just remove the -s flag. To lower the resolution, you can use -r 400,200 (pixel values in x and y direction).

[3]:
```
%%manim -v WARNING  --progress_bar None  -r 400,200  --disable_caching HelloManim
class HelloManim(Scene):
    def construct(self):
        self.camera.background_color = "#ece6e2"
        banner_large = ManimBanner(dark_theme=False).scale(0.7)
        self.play(banner_large.create())
        self.play(banner_large.expand())
```

```
<IPython.core.display.Video object>
```

We can define the parameters as a string `params` and call this string by the cell magic with `$params`

[4]:
```
params = "-v WARNING  -s -ql  --disable_caching Example"
paramsSMALL = "-v WARNING -r 400,200 -s  --disable_caching Example"
```

[5]:
```
%%manim $params
class Example(Scene):
```

(continues on next page)

(continued from previous page)

```
    def construct(self):
        m= ManimBanner()
        self.add(m)
```

1.1.2 Initializing Mobjects Outside the Class

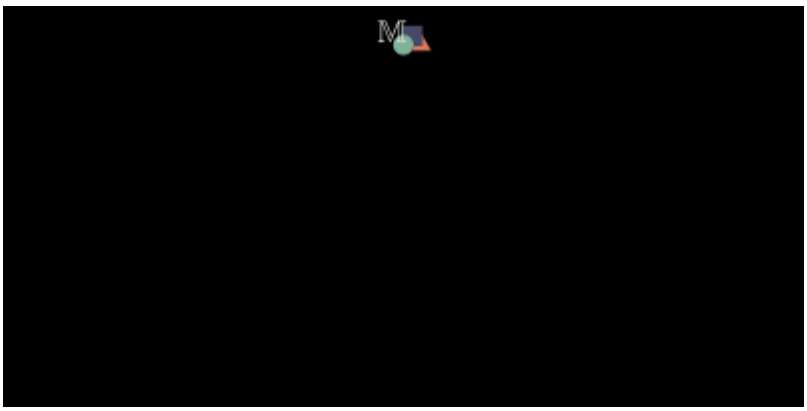
In some cases, it might be convenient to define mobjects outside the `Scene` class (e.g. for uncluttering or for speeding up the animation).

[6]:
```
m = ManimBanner()
```

[7]:
```
%%manim $paramsSMALL
class Example(Scene):
    def construct(self):
        m.scale(0.4        )
        m.shift(1.5*UP)
        self.add(m)
```

Because the mobject is manipulated in the class, the next cell might show some unexpected scaling and shifting:

[8]:
```
%%manim $paramsSMALL
class Example(Scene):
    def construct(self):
        m.scale(0.4)
        m.shift(1.5*UP)
        self.add(m)
```



To aviod this, it is better to add only a copy of these mobjects to scenes, and keep the originals untouched:

[9]:
```
m_reference = ManimBanner()
```

[10]:
```
%%manim $paramsSMALL
class Example(Scene):
    def construct(self):
        m = m_reference.copy()
        m.scale(0.4)
        m.shift(2*UP)
        self.add(m)
```

[11]:
```
%%manim $paramsSMALL
class Example(Scene):
    def construct(self):
        m = m_reference.copy()
        m.scale(0.4)
        m.shift(2*UP)
        self.add(m)
```

1.1.3 Defining Global Mobjects

When you have to build complex scenes, you might want to use parts of that scene for your next scene. That is possible with global variables, which can be accessed in any other scene.

[12]:
```
%%manim $paramsSMALL
class Example(Scene):
    def construct(self):
        stars= VGroup()
        for i in range(0,20):
            s= Star(color= random_bright_color(), fill_opacity=1).scale(0.8)
            stars.add(s)
        stars.arrange_in_grid()
        self.add(stars)
        global favoritstar
        favoritstar = stars[9]
```

[13]:
```
%%manim $paramsSMALL
class Example(Scene):
    def construct(self):
        self.add(favoritstar)
```

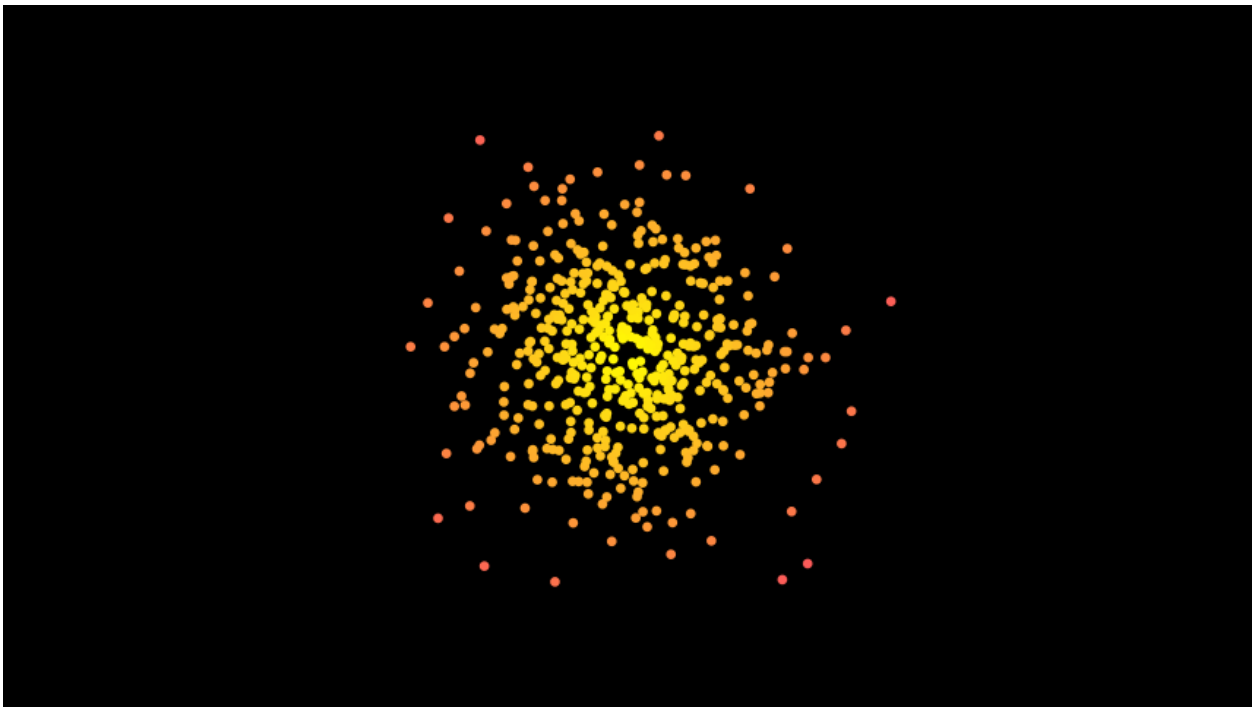
1.1.4 Pre-Execute Slow Code

In this example, calculating a random walk for 500 particles and 100000 steps takes about 4 seconds.
This step can be done before the actual scene construction, which takes about 0.2 seconds.
Making aesthetic changes to the scene will then become easier.
Note: The `%%time` command will print the execution time of the cells.

[14]:
```
%%time
np.random.seed(20)
steps = np.random.choice(a=[-1, 0, 1], size=(100000,1000))
stop = steps.cumsum(0)
end_points= stop[-1]/stop[-1].max()
end_pointsX = end_points[0:499]
end_pointsY = end_points[500:-1]
```

```
CPU times: user 2.11 s, sys: 751 ms, total: 2.86 s
Wall time: 2.86 s
```

[15]:
```
%%time
%%manim $params
class Example(Scene):
    def construct(self):
        radius= (end_pointsX*end_pointsX + end_pointsY * end_pointsY)**0.5
        dots = VGroup()
        for  x,y,r in zip(end_pointsX, end_pointsY,radius):
            c= interpolate_color(YELLOW, RED, r)
            dots.add(Dot(color=c,point=[3*x,3*y,0]).scale(0.7))
        self.add(dots)
```



```
CPU times: user 407 ms, sys: 3.82 ms, total: 411 ms
Wall time: 411 ms
```

1.1.5 Installing Plugins

plugins can be found at https://plugins.manim.community/

[16]:
```
!pip install manim-physics
```

```
Collecting manim-physics
  Downloading manim_physics-0.2.3-py3-none-any.whl (9.9 kB)
Requirement already satisfied: manim>=0.6.0 in /home/docs/checkouts/readthedocs.org/user_
↪builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim-physics) (0.
↪11.0)
Collecting pymunk<7.0.0,>=6.0.0
  Downloading pymunk-6.2.0-cp38-cp38-manylinux2010_x86_64.whl (984 kB)
     || 984 kB 4.6 MB/s
Requirement already satisfied: setuptools in /home/docs/checkouts/readthedocs.org/user_
↪builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.0->manim-
↪physics) (58.2.0)
```

(continues on next page)

(continued from previous page)

```
Requirement already satisfied: click>=7.1 in /home/docs/checkouts/readthedocs.org/user_
↪builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.0->manim-
↪physics) (8.0.3)
Requirement already satisfied: scipy in /home/docs/checkouts/readthedocs.org/user_builds/
↪flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.0->manim-
↪physics) (1.7.1)
Requirement already satisfied: pydub in /home/docs/checkouts/readthedocs.org/user_builds/
↪flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.0->manim-
↪physics) (0.25.1)
Requirement already satisfied: moderngl<6.0.0,>=5.6.3 in /home/docs/checkouts/
↪readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages␣
↪(from manim>=0.6.0->manim-physics) (5.6.4)
Requirement already satisfied: pycairo<2.0,>=1.19 in /home/docs/checkouts/readthedocs.
↪org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.
↪0->manim-physics) (1.20.1)
Requirement already satisfied: mapbox-earcut<0.13.0,>=0.12.10 in /home/docs/checkouts/
↪readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages␣
↪(from manim>=0.6.0->manim-physics) (0.12.10)
Requirement already satisfied: pygments in /home/docs/checkouts/readthedocs.org/user_
↪builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.0->manim-
↪physics) (2.10.0)
Requirement already satisfied: isosurfaces==0.1.0 in /home/docs/checkouts/readthedocs.
↪org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.
↪0->manim-physics) (0.1.0)
Requirement already satisfied: skia-pathops<0.8.0,>=0.7.0 in /home/docs/checkouts/
↪readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages␣
↪(from manim>=0.6.0->manim-physics) (0.7.1)
Requirement already satisfied: Pillow in /home/docs/checkouts/readthedocs.org/user_
↪builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.0->manim-
↪physics) (8.3.2)
Requirement already satisfied: manimpango<0.4.0,>=0.3.0 in /home/docs/checkouts/
↪readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages␣
↪(from manim>=0.6.0->manim-physics) (0.3.1)
Requirement already satisfied: tqdm in /home/docs/checkouts/readthedocs.org/user_builds/
↪flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.0->manim-
↪physics) (4.62.3)
Requirement already satisfied: numpy<2.0,>=1.9 in /home/docs/.pyenv/versions/3.8.6/lib/
↪python3.8/site-packages (from manim>=0.6.0->manim-physics) (1.19.2)
Requirement already satisfied: decorator<6.0.0,>=5.0.7 in /home/docs/checkouts/
↪readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages␣
↪(from manim>=0.6.0->manim-physics) (5.1.0)
Requirement already satisfied: networkx<3.0,>=2.5 in /home/docs/checkouts/readthedocs.
↪org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.
↪0->manim-physics) (2.6.3)
Requirement already satisfied: rich>=6.0 in /home/docs/checkouts/readthedocs.org/user_
↪builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.0->manim-
↪physics) (10.12.0)
Requirement already satisfied: requests in /home/docs/checkouts/readthedocs.org/user_
↪builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.0->manim-
↪physics) (2.26.0)
Requirement already satisfied: screeninfo<0.7.0,>=0.6.7 in /home/docs/checkouts/
↪readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages␣
↪(from manim>=0.6.0->manim-physics) (0.6.7)
```

(continues on next page)

(continued from previous page)

```
Requirement already satisfied: watchdog in /home/docs/checkouts/readthedocs.org/user_
↪builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.0->manim-
↪physics) (2.1.6)
Requirement already satisfied: moderngl-window<3.0.0,>=2.3.0 in /home/docs/checkouts/
↪readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages␣
↪(from manim>=0.6.0->manim-physics) (2.4.0)
Requirement already satisfied: cloup<0.8.0,>=0.7.0 in /home/docs/checkouts/readthedocs.
↪org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.
↪0->manim-physics) (0.7.1)
Requirement already satisfied: colour in /home/docs/checkouts/readthedocs.org/user_
↪builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.0->manim-
↪physics) (0.1.5)
Requirement already satisfied: click-default-group in /home/docs/checkouts/readthedocs.
↪org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from manim>=0.6.
↪0->manim-physics) (1.2.2)
Collecting cffi>1.14.0
  Downloading cffi-1.15.0-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (446␣
↪kB)
     || 446 kB 60.1 MB/s
Collecting pycparser
  Downloading pycparser-2.20-py2.py3-none-any.whl (112 kB)
     || 112 kB 77.5 MB/s
Requirement already satisfied: glcontext<3,>=2 in /home/docs/checkouts/readthedocs.org/
↪user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from moderngl<6.0.0,
↪>=5.6.3->manim>=0.6.0->manim-physics) (2.3.4)
Requirement already satisfied: pyrr<1,>=0.10.3 in /home/docs/checkouts/readthedocs.org/
↪user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from moderngl-window
↪<3.0.0,>=2.3.0->manim>=0.6.0->manim-physics) (0.10.3)
Requirement already satisfied: pyglet<2,>=1.5.8 in /home/docs/checkouts/readthedocs.org/
↪user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from moderngl-window
↪<3.0.0,>=2.3.0->manim>=0.6.0->manim-physics) (1.5.21)
Requirement already satisfied: commonmark<0.10.0,>=0.9.0 in /home/docs/checkouts/
↪readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages␣
↪(from rich>=6.0->manim>=0.6.0->manim-physics) (0.9.1)
Requirement already satisfied: colorama<0.5.0,>=0.4.0 in /home/docs/checkouts/
↪readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages␣
↪(from rich>=6.0->manim>=0.6.0->manim-physics) (0.4.4)
Requirement already satisfied: certifi>=2017.4.17 in /home/docs/checkouts/readthedocs.
↪org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from requests->
↪manim>=0.6.0->manim-physics) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in /home/docs/checkouts/readthedocs.org/user_
↪builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from requests->manim>=0.
↪6.0->manim-physics) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/docs/checkouts/readthedocs.
↪org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from requests->
↪manim>=0.6.0->manim-physics) (1.26.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /home/docs/checkouts/
↪readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages␣
↪(from requests->manim>=0.6.0->manim-physics) (2.0.7)
Requirement already satisfied: multipledispatch in /home/docs/checkouts/readthedocs.org/
↪user_builds/flyingframes/envs/v0.11.0/lib/python3.8/site-packages (from pyrr<1,>=0.10.
↪3->moderngl-window<3.0.0,>=2.3.0->manim>=0.6.0->manim-physics) (0.6.0)
```

(continues on next page)

(continued from previous page)

```
Requirement already satisfied: six in /home/docs/.pyenv/versions/3.8.6/lib/python3.8/
↪site-packages (from multipledispatch->pyrr<1,>=0.10.3->moderngl-window<3.0.0,>=2.3.0->
↪manim>=0.6.0->manim-physics) (1.15.0)
Installing collected packages: pycparser, cffi, pymunk, manim-physics
Successfully installed cffi-1.15.0 manim-physics-0.2.3 pycparser-2.20 pymunk-6.2.0
```

[17]:
```
%%manim -v WARNING  --progress_bar None  -qm --disable_caching Example

from manim_physics import *

class Example(SpaceScene):
    def construct(self):
        circle = Dot(radius=1).shift(1.5*LEFT+3*UP)
        rect = Square(color=YELLOW, fill_opacity=1)
        ground = Line([-4, -3.5, 0], [4, -3.5, 0])
        wall1 = Line([-4, -3.5, 0], [-4, 3.5, 0])
        wall2 = Line([4, -3.5, 0], [4, 3.5, 0])
        walls = VGroup(ground, wall1, wall2)
        self.add(walls)
        self.add(rect, circle)
        self.make_rigid_body(rect, circle)
        self.make_static_body(walls)
        self.wait(5)
```

```
<IPython.core.display.Video object>
```

1.2 Mobject Gallery

Also available on this standalone website: https://kolibril13.github.io/mobject-gallery/

1.3 Mobject Basics

After reading this chapter, you will be able to build up Mobjects on scenes, no animations included yet. There will be lots of minimal examples and only very brief explanations.

[1]:
```
from manim import *
```

```
Manim Community v0.11.0
```

[2]:
```
#ignore this cell, only for setup
params = "-v WARNING -r 500,100 -s  --disable_caching Example"
paramsbigger = "-v WARNING -r 500,120 -s  --disable_caching Example"
```

1.3.1 Positioning

First we want to position mobjects. There are tons of options, and not everything will be covered here.

set positions

Some important methods to set positions are:
`shift` `move_to` `align_to` , `next_to` ,`to_corner` , `to_edge` , `arrange` , `arrange_in_grid`

[3]:
```
dORIGIN= Dot(color= BLUE,radius=0.5)
```

[4]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        d= Dot(color= YELLOW, radius=0.5)
        d.shift(2*RIGHT)
        self.add(dORIGIN, d)
```

[5]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        d= Dot(color= YELLOW, radius=0.5)
        d.shift(3*RIGHT+0.8*UP)
        self.add(dORIGIN, d)
```

[6]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        d= Dot(color= ORANGE, radius=0.5)
        d.next_to(dORIGIN, LEFT)
        self.add(dORIGIN, d)
```

[7]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        d= Dot(color= ORANGE, radius=0.5)
        d.next_to(dORIGIN, LEFT, buff=0)
        self.add(dORIGIN, d)
```

[8]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        d= Dot(color= ORANGE, radius=0.5)
        d.next_to(dORIGIN, LEFT, buff=4)
        self.add(dORIGIN, d)
```

[9]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        d= Dot(color= ORANGE, radius=0.5)
        d.next_to(dORIGIN, UL,buff=-0.5) # UL is UPLEFT
        self.add(dORIGIN, d)
```

[10]:
```
%%manim $params
class Example(Scene):
    def construct(self):
```

(continues on next page)

(continued from previous page)

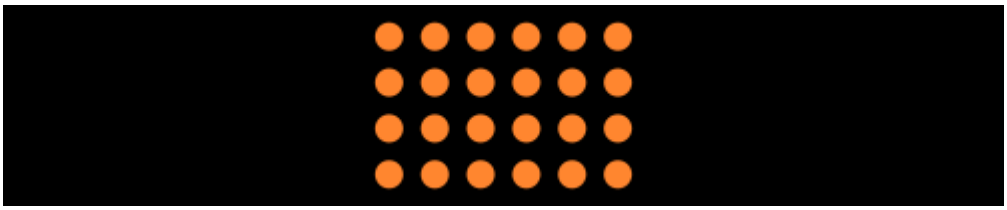
```
        d= Dot(color= ORANGE, radius=0.5)
        d.to_edge(LEFT)
        self.add(dORIGIN, d)
```

[11]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        s= Star(stroke_width=10)
        d=Dot(color= ORANGE, radius=0.5)
        d.align_to(s,DOWN)
        self.add(s,d)
```

[12]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        s= Star(stroke_width=10)
        d=Dot(color= ORANGE, radius=0.5)
        d.next_to(s,RIGHT, aligned_edge=UP) #next to and align combined
        self.add(s,d)
```

[13]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        for i in range(0,10):
            self.add(Dot(color= ORANGE, radius=0.5))
        VGroup(*self.mobjects).arrange()
```

[14]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        for i in range(0,24):
            self.add(Dot(color= ORANGE, radius=0.2))
        VGroup(*self.mobjects).arrange_in_grid(cols=6)
```



get positions

The most important methods to get positions:
`get_center` , `get_top` , `get_right` , `get_start`

[15]:
```
s= Star(stroke_width=10)
d=Dot(color= YELLOW, radius=0.2)
```

[16]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= s.get_center()
        self.add(s, d.move_to(pos))
```

[17]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= s.get_center_of_mass()
        self.add(s, d.move_to(pos))
```

[18]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= s.get_top()
        self.add(s, d.move_to(pos))
```

[19]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= s.get_right()
        self.add(s, d.move_to(pos))
```

[20]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= s.get_bottom()
        self.add(s, d.move_to(pos))
```

[21]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= s.get_left()
        self.add(s, d.move_to(pos))
```

[22]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= s.get_corner(UL)
        self.add(s, d.move_to(pos))
```

[23]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= s.get_corner(DR)
        self.add(s, d.move_to(pos))
```

[24]:
```
arc= Arc(radius=1.0, start_angle=-PI/16, angle=PI, stroke_width=10)
```

[25]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= arc.get_start()
        self.add(arc, d.move_to(pos))
```

[26]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= arc.get_end()
```

(continues on next page)

(continued from previous page)

```
        self.add(arc, d.move_to(pos))
```

[27]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= arc.get_midpoint()
        self.add(arc, d.move_to(pos))
```

[28]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= arc.point_from_proportion(0.2)
        self.add(arc, d.move_to(pos))
```

[29]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= arc.get_center()
        self.add(arc, d.move_to(pos))
```

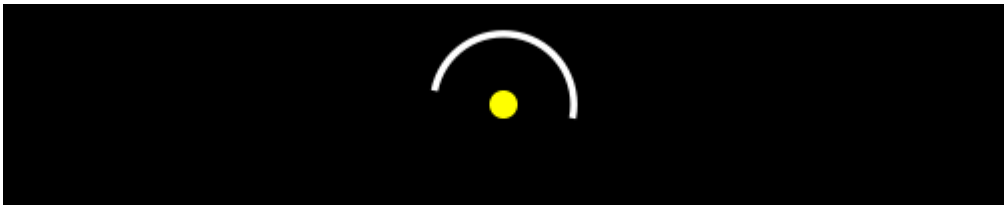
[30]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= arc.get_center_of_mass()
```

(continues on next page)

(continued from previous page)

```
        self.add(arc, d.move_to(pos))
```

[31]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        pos= arc.get_arc_center()
        self.add(arc, d.move_to(pos))
```

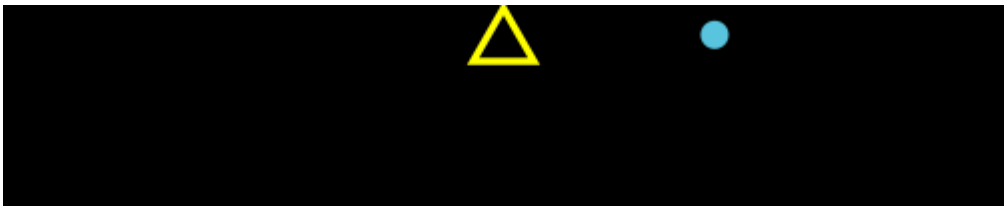


[32]:
```
%%manim $params
class Example(Scene): #Example for `get_x` , `get_y`, `set_x` and `set_y`
    def construct(self):
        d = Dot(point=[3,1,0],radius=0.2,color= BLUE)
        triangle= Triangle(color=YELLOW, stroke_width=10).scale(0.5)
        x_pos=d.get_x()
        print(x_pos)
        triangle.set_x(x_pos)
        self.add(d, triangle)
```

```
3.0
```

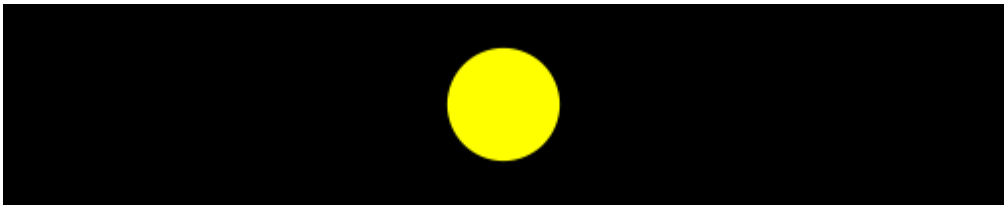
[33]:
```
%%manim $params
class Example(Scene): #Example for `get_x` , `get_y`, `set_x` and `set_y`
    def construct(self):
        d = Dot(point=[3,1,0],radius=0.2,color= BLUE)
        triangle= Triangle(color=YELLOW, stroke_width=10).scale(0.5)
        y_pos=d.get_y()
        print(y_pos)
        triangle.set_y(y_pos)
        self.add(d, triangle)
```

```
1.0
```



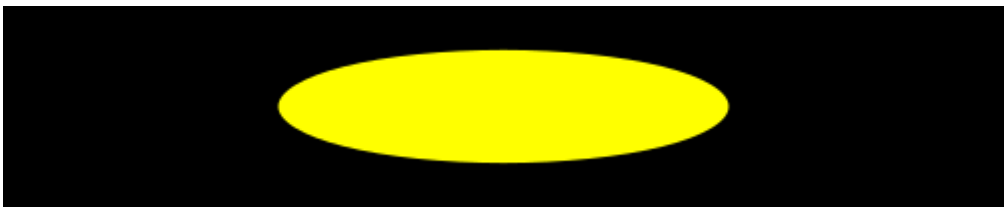
1.3.2 Scaling and Stretching

[34]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        d= Dot(color= YELLOW)
        d.scale(10)
        self.add(d)
```



[35]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        d= Dot(color= YELLOW)
        d.scale(10)
        d.stretch_in_place(4, dim = 0) # dim = 0 means vertical
        self.add(d)
```

```
[10/14/21 16:39:00] WARNING  The method Mobject.stretch_in_place has been  ]8;id=8663;
↪file:///home/docs/checkouts/readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/
↪python3.8/site-packages/manim/utils/deprecation.py\deprecation.py]8;;\:210
                             deprecated since v0.11.0 and is expected to
                             be removed after v0.12.0. Use stretch
                             instead.
```



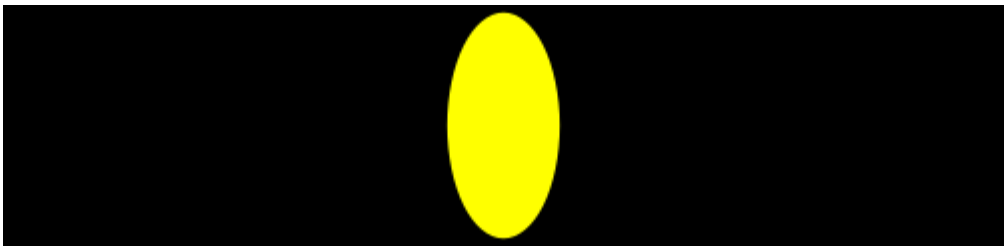
[36]:
```
%%manim $paramsbigger
class Example(Scene):
    def construct(self):
        d= Dot(color= YELLOW)
        d.scale(10)
```

(continues on next page)

(continued from previous page)

```
        d.stretch_in_place(2, dim = 1) # dim = 1 means horizontal
        self.add(d)
```

```
                    WARNING  The method Mobject.stretch_in_place has been  ]8;id=176506;
↪file:///home/docs/checkouts/readthedocs.org/user_builds/flyingframes/envs/v0.11.0/lib/
↪python3.8/site-packages/manim/utils/deprecation.py\deprecation.py]8;;\:210
                             deprecated since v0.11.0 and is expected to
                             be removed after v0.12.0. Use stretch
                             instead.
```



[37]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        d= Dot(color= YELLOW)
        d.scale(10)
        d.apply_matrix([[0.5, 0.5, 0], # shear matrix
                        [ 0 ,  1 , 0],
                        [ 0 ,  0 , 1]])
        self.add(d)
```

1.3.3 Rotating

[38]:
```
%%manim $paramsbigger
class Example(Scene):
    def construct(self):
        m= ManimBanner().scale(0.5)
        m.rotate(PI/8)
        self.add(m)
```

[39]:
```
%%manim $paramsbigger
class Example(Scene):
    def construct(self):
        m= ManimBanner().scale(0.5)
        m.rotate(-20*DEGREES)
        self.add(m)
```

[40]:
```
%%manim $paramsbigger
class Example(Scene):
    def construct(self):
        m= ManimBanner().scale(0.5)
        self.add(m.copy())
        m.rotate(about_point=2*LEFT, angle=180*DEGREES)
        self.add(m, Dot(2*LEFT,radius=0.1))
```

[41]:
```
%%manim $paramsbigger
class Example(Scene):
    def construct(self):
        m= ManimBanner().scale(0.5)
        m.rotate(axis=UP,angle=60*DEGREES)
        self.add(m)
```

Note

Python is very fertile tool, there multiple ways to accomplish a certain task, but some options are not "best practice". For the methods in the next chapters, I want to show the best practice (labeled with **BEST** and the **green check with the star**), other possible options (labeled with **YES** and the **green check**), and options that do not work (labeled with **NO** and the **red cross**)

[42]:
```
# ignore this cell, only for setup
YES = SVGMobject("good.svg").to_edge(LEFT, buff=1)
BEST = YES.copy()
BEST.add(Star(color= YELLOW, fill_opacity=1).scale(0.5).move_to(BEST).shift(0.5*DOWN+0.
↪5*RIGHT))
NO  = Cross(Square(), stroke_color = RED_D, stroke_width = 38).scale(0.9).move_to(YES)
```

1.3.4 Colors and Opacity

- Color parameters for Mobjects are `stroke_color`, `fill_color` and `color`. The parameter `color` automatically sets both `stroke_color` and `fill_color`.
  The recommended ways to set **colors** are via `c = Circle(fill_color= BLUE, fill_opacity= 1 )`, `c.set_fill(color=RED)` or `c.set_style(fill_color=GREEN)`
  Not possible are `c.fill_color=YELLOW`, `c.set(fill_color=YELLOW)` and `c.set_fill_color(YELLOW)`
- Opacity parameters for Mobjects are `fill_opacity` and `stroke_opacity` (there is **not** `opacity` here).
  The recommended ways to set **opacity** are via `c = Circle(fill_color= BLUE, fill_opacity= 0.5 )`, `c.set_fill(color=RED)` or `c.set_style(fill_color=GREEN)`
  Analog to colors, `c.fill_opacity=1`, `c.set(fill_opacity=1)` and `c.set_fill_opacity(1)` are not possible. (to keep things short, these examples are not shown).

Colors

[43]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        c = Square(fill_color= BLUE, fill_opacity= 1 )
        self.add(BEST,c)
```

[44]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        c = Square(fill_color= BLUE, fill_opacity= 1)
        c.set_fill(color=RED)
        self.add(BEST,c)
```

[45]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        c = Square(fill_color= BLUE, fill_opacity= 1)
        c.set_style(fill_color=GREEN)
        self.add(BEST,c)
```

[46]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        c = Square(fill_opacity= 1)
        c.fill_color=YELLOW
        self.add(NO,c)
```

[47]:
```
%%manim $params
class Example(Scene):
    def construct(self):
```

(continues on next page)

(continued from previous page)

```
        c = Square(fill_opacity= 1)
        c.set(fill_color=YELLOW)
        self.add(NO,c)
```

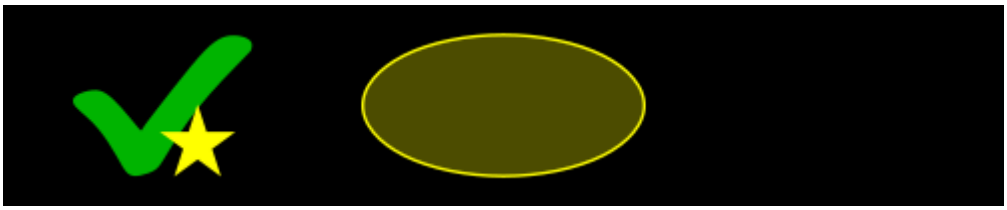
[48]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        c = Square(fill_opacity= 1)
        c.set_fill_color(YELLOW)
        self.add(NO,c)
```

```
<string>:4: DeprecationWarning: This method is not guaranteed to stay around. Please␣
↪prefer setting the attribute normally or with Mobject.set().
```

[49]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        self.camera.background_color = PURPLE
        self.add(BEST)
```

Opacitiy

[50]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        c = Ellipse(color= YELLOW, fill_opacity=0.3).scale(2)
        self.add(BEST,c)
```



[51]:
```
%%manim  $params
class Example(Scene):
    def construct(self):
        c = Ellipse(color= YELLOW).scale(2)
        c.set_fill(opacity=0.5) # be careful: here, it must be `opacity` and not `fill_
↪opacity` !
        self.add(BEST,c)
```

[52]:
```
%%manim  $params
class Example(Scene):
    def construct(self):
        c = Ellipse(color= YELLOW).scale(2)
        c.set_style(fill_opacity=0.7) # and here, it must be `fill_opacity` and not␣
↪`opacity` !
        self.add(BEST,c)
```

1.3.5 Stroke width

Strokes can be set in multiple ways:
The recommended ways are via `Circle(stroke_width=30)` , `c.set_stroke(width = 30)` or `c.set_style(stroke_width= 30)`
Also possible, but not the best solution is `c.stroke_width = 30` and `c.set(stroke_width = 30)`
Also possible, but not recommended because deprecated is `c.set_stroke_width(30)`

[53]:
```
%%manim $params
class Example(Scene):
    def construct(self):
```

(continues on next page)

(continued from previous page)

```
        c = Circle(stroke_width=30)
        self.add(BEST,c)
```

[54]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        c = Circle()
        c.set_stroke(width = 30)
        self.add(BEST,c)
```

[55]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        c = Circle()
        c.set_style(stroke_width= 30)
        self.add(BEST,c)
```

[56]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        c = Circle()
        c.stroke_width = 30
        self.add(YES,c)
```

[57]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        c = Circle()
        c.set(stroke_width = 30)
        self.add(YES,c)
```

[58]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        c = Circle()
        c.set_stroke_width(30)
        self.add(NO,c)
```

```
<string>:4: DeprecationWarning: This method is not guaranteed to stay around. Please␣
↪prefer setting the attribute normally or with Mobject.set().
```

1.3.6 Layers

There are two main ways to change the layers of Mobjects:

1. Reordering the list of submobjects that where added
2. Using the z_index

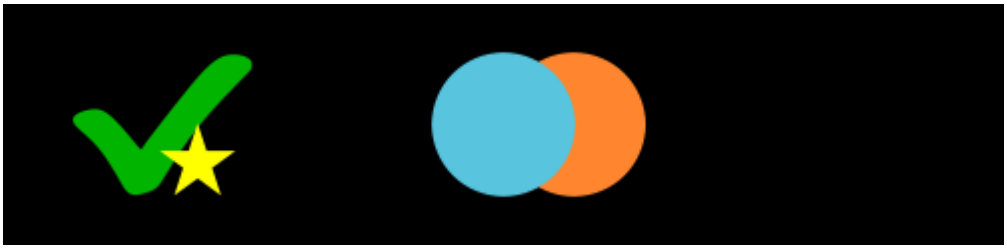
submobjects

A scene stores displayed mobjects in a list. They are displayed in the order that they are added to the scene with the syntax `self.add(circleLeft,circleRight)` First, we have a look on positioning mobjects with `self.add` and the methods `self.bring_to_back` and `self.bring_to_front`. In most cases, this is completely enough. Later, we will come to the z_index, that is seen by manim by one priority higher: Even when a mobject is added first to the mobject list, it will be displayed on top of the others, if it has a higher z_index. An example about this will be seen later.

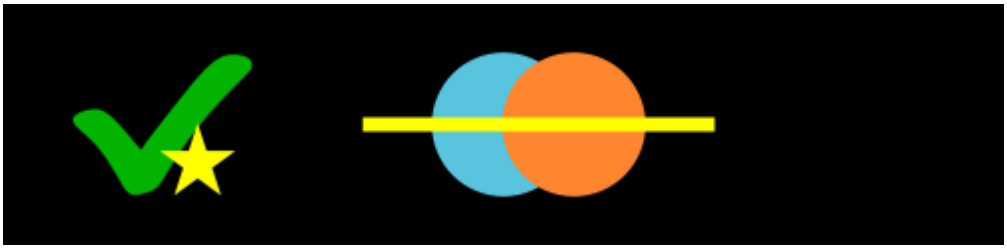
[59]:
```
circleLeft = Circle(color=BLUE, fill_opacity=1)
circleRight = Circle(color=ORANGE,fill_opacity=1).shift(RIGHT)
line = Line(2*LEFT,3*RIGHT,color=YELLOW, stroke_width=20)
```

[60]:
```
%%manim  $paramsbigger
class Example(Scene):
    def construct(self):
        self.add(circleLeft,circleRight)
        self.add(BEST)
```

[61]:
```
%%manim  $paramsbigger
class Example(Scene):
    def construct(self):
        self.add(circleRight,circleLeft)
        self.add(BEST)
```

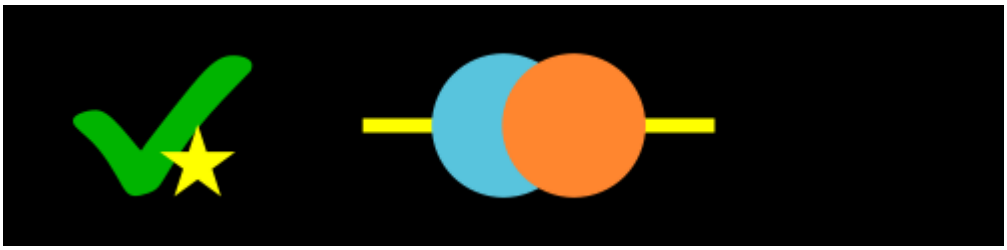


[62]:
```
%%manim  $paramsbigger
class Example(Scene):
    def construct(self):
        self.add(circleLeft,circleRight, line)
        self.add(BEST)
```



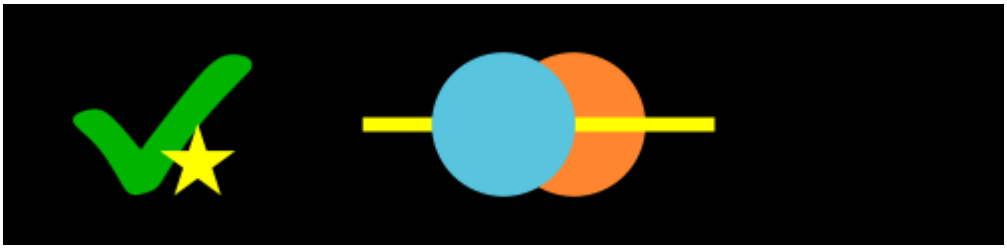
[63]:
```
%%manim  $paramsbigger
class Example(Scene):
    def construct(self):
        self.add(circleLeft,circleRight, line)
        print(self.mobjects)
        self.bring_to_back(line)
        print(self.mobjects)
        self.add(BEST)
```

```
[Circle, Circle, Line]
[Line, Circle, Circle]
```



[64]:
```
%%manim  $paramsbigger
class Example(Scene):
    def construct(self):
        self.add(circleLeft,circleRight, line)
        print(self.mobjects)
        self.bring_to_front(circleLeft)
        print(self.mobjects)
        self.add(BEST)
```

```
[Circle, Circle, Line]
[Circle, Line, Circle]
```



z_index

The default z_index is 0. Now we will see what happens, when we increase the value of the z_index.
The z_index can be changed by `triangle = Triangle(z_index=1)`,`triangle.z_index=1` , `triangle.set(z_index=1)` and `triangle.set_z_index(1)`
It can not be changed using `triangle.set_style(z_index=1)`

[65]:
```
#initilizing line,circle,square and triangle
BUFF= 0.5*DOWN
line = Line(3*LEFT,3*RIGHT,color=YELLOW, stroke_width=20)
circle = Circle(color=GREEN_D, fill_opacity=1).shift(LEFT+BUFF)
square = Square(color=BLUE_D, fill_opacity=1).shift(UP+BUFF)
triangle = Triangle(color=RED_D, fill_opacity=1).shift(RIGHT+BUFF)
```

[66]:
```
%%manim  $paramsbigger
class Example(Scene):
    def construct(self):
        self.add(line,triangle, square, circle)  # order matters
        print(self.mobjects)
```

(continues on next page)

(continued from previous page)

```
        print(f"{triangle.z_index=} , {square.z_index=} , {circle.z_index=} , {line.z_
↪index=}")
        self.add(BEST)
```

```
[Line, Triangle, Square, Circle]
triangle.z_index=0 , square.z_index=0 , circle.z_index=0 , line.z_index=0
```

[67]:
```
%%manim  $paramsbigger
class Example(Scene):
    def construct(self):
        triangle.z_index=1
        self.add(triangle, square, circle,line)  # order matters
        print(self.mobjects)
        print(f"{triangle.z_index=} , {square.z_index=} , {circle.z_index=} , {line.z_
↪index=}")
        self.add(BEST)
```

```
[Triangle, Square, Circle, Line]
triangle.z_index=1 , square.z_index=0 , circle.z_index=0 , line.z_index=0
```

[68]:
```
%%manim  $paramsbigger
class Example(Scene):
    def construct(self):
        triangle.z_index = 1
        square.z_index   = 2
        circle.z_index   = 3
        self.add(triangle, square, circle,line)  # order matters
        self.add(BEST)
        print(f"{line.z_index=}")
```

```
line.z_index=0
```

[69]:
```
%%manim  $paramsbigger
class Example(Scene):
    def construct(self):
        triangle.z_index = 3
        square.z_index   = 2
        circle.z_index   = 1
        self.add(triangle, square, circle,line)  # order matters
        self.add(BEST)
        print(f"{line.z_index=}")
```

```
line.z_index=0
```

[70]:
```
%%manim  $paramsbigger
triangle.z_index = 0
square.z_index   = 0
circle.z_index   = 0
class Example(Scene):
    def construct(self):
        triangle.set(z_index=1)
        self.add(triangle, square, circle,line)  # order matters
        print(self.mobjects)
        print(f"{triangle.z_index=} , {square.z_index=} , {circle.z_index=} , {line.z_
↪index=}")
        self.add(BEST)
```

```
[Triangle, Square, Circle, Line]
triangle.z_index=1 , square.z_index=0 , circle.z_index=0 , line.z_index=0
```

[71]:
```
%%manim  $paramsbigger
triangle.z_index = 0
square.z_index   = 0
circle.z_index   = 0
class Example(Scene):
    def construct(self):
        try:
            triangle.set_style(z_index=1)  # here we expect an error! Only for didactic␣
↪purpose, it is put into this `try` blog, so that no long error message is shown.
        except TypeError:
            print("TypeError, set_style() got an unexpected keyword argument 'z_index'.
↪")
            self.add(NO)
        self.add(triangle, square, circle,line)  # order matters
        print(f"{triangle.z_index=} , {square.z_index=} , {circle.z_index=} , {line.z_
↪index=}")
```

```
TypeError, set_style() got an unexpected keyword argument 'z_index'.
triangle.z_index=0 , square.z_index=0 , circle.z_index=0 , line.z_index=0
```

1.3.7 VGroup and Group

VGroup

It is a Group of VMobjects (“V” stands for Vector)

[72]:
```
#only for setup
def create_dots():
    blue1_ref= Dot(color= BLUE,       point=[-.3,-.5,0], radius=0.5)
    blue2_ref= Dot(color= BLUE_A,     point=[ .3,-.5,0], radius=0.5)
    yellow1_ref= Dot(color= YELLOW,   point=[-.3, .5,0], radius=0.5)
    yellow2_ref= Dot(color= YELLOW_A, point=[ .3, .5,0], radius=0.5)
    return blue1_ref, blue2_ref,yellow1_ref,yellow2_ref
```

[73]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene):
    def construct(self):
        self.add(blue1,blue2, yellow1,yellow2)
```

[74]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene):
    def construct(self):
        VGroup(yellow1,yellow2).shift(RIGHT)
        self.add(blue1,blue2, yellow1,yellow2)
```

[75]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene):
    def construct(self):
        g1=VGroup(yellow1,yellow2).shift(2*RIGHT)
        self.add(blue1,blue2, g1)
```

[76]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene):
    def construct(self):
        g1=VGroup(yellow1,yellow2).set_color(RED)
        self.add(blue1,blue2, g1)
```

[77]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene):
```

(continues on next page)

(continued from previous page)

```
    def construct(self):
        g1=VGroup(yellow1,yellow2).shift(0.5*DOWN)
        g2=VGroup(blue1,blue2)
        self.add(g1, g2)
```

[78]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene):
    def construct(self):
        g1=VGroup(yellow1,yellow2).shift(0.5*DOWN)
        g2=VGroup(blue1,blue2)
        self.add(g2,g1)
```

[79]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene): # Groups of Groups
    def construct(self):
        g1=VGroup(yellow1,yellow2).shift(0.5*DOWN)
        g2=VGroup(blue1,blue2)
        gAll = VGroup(g1, g2)
        self.add(gAll)
        print(gAll.submobjects)
        print(gAll.submobjects[0].submobjects)
        print(gAll.submobjects[1].submobjects)
```

```
[VGroup(Dot, Dot), VGroup(Dot, Dot)]
[Dot, Dot]
[Dot, Dot]
```

[80]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
```

(continues on next page)

(continued from previous page)

```
class Example(Scene): #setting VMobject attibutes
    def construct(self):
        g=VGroup(yellow1,yellow2,blue1,blue2)
        g.set_stroke(color=PURPLE_D, width=20) # <--
        self.add(g)
```

[81]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene): # add syntax
    def construct(self):
        g=VGroup(yellow1,yellow2,blue1)
        g.add(blue2) # <--
        g.set_stroke(color=GREEN, width=20)
        self.add(g)
```

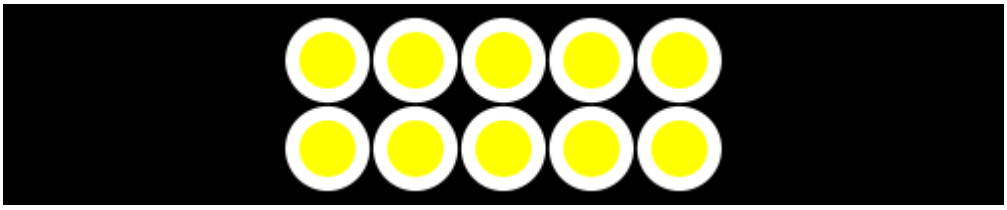
[82]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene): # += Syntax
    def construct(self):
        g=VGroup(yellow1,yellow2,blue1)
        g += blue2 # <--
        g.set_stroke(color=ORANGE, width=20)
        self.add(g)
```

[83]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene): # empty initilizing
    def construct(self):
        g=VGroup()
        for _ in range(0,10):
            g += yellow1.copy()
```

(continues on next page)

(continued from previous page)

```
        g.set_stroke(color=WHITE, width=20)
        g.arrange_in_grid(rows=2) # <-- Groups and VGroups can be arranged in grids
        g.move_to(ORIGIN)
        self.add(g)
```



Note:

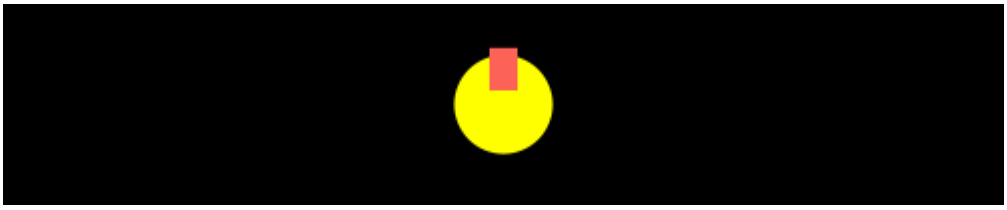
VMobject().add(…) is functionally equivalent to VGroup(…), but it is recommended to use VGroup, as

- It is better readable
- supports the += syntax

---

[84]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene):  # using VMobject instead
    def construct(self):
        g= VMobject()
        g.add(yellow1,yellow2,blue1,blue2)
        g.set_stroke(color=PURPLE_D, width=20)
        self.add(g)
```

[85]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene):  # other Mobjects can be added to any Mobjects
    def construct(self):
        d= Dot(color= YELLOW, radius=0.7)
        d.add(Line(0.2*UP, 0.8*UP, color=RED,stroke_width=40))
        self.add(d)
```



[86]:
```
%%manim $params
dot= Dot(color= YELLOW, radius=0.5)
image = ImageMobject(np.uint8([[200, 233, 111, 200],
                               [255, 100, 190, 100]])).shift(2*RIGHT)
image.height = 1
class Example(Scene):
    def construct(self):
        self.add(dot, image)
        try: # Image is not a VMobject!
            VGroup(dot,image).shift(3*RIGHT)
        except TypeError:
            print("Adding an Mobject to a VGroup is not possible!")
            self.add(NO)
```

```
Adding an Mobject to a VGroup is not possible!
```

Group

Groups Mobjects and VMobjects togehter. You can only use the methods of Mobject here. Methods of VMobject wont be supported.

[87]:
```
%%manim $params
dot= Dot(color= YELLOW, radius=0.5)
image = ImageMobject(np.uint8([[200, 233, 111, 200],
                               [255, 100, 190, 100]])).shift(2*RIGHT)
image.height = 1
class Example(Scene):
    def construct(self):
        self.add(dot, image)
        Group(dot,image).shift(3*RIGHT)
```

[88]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene): #
    def construct(self):
        g=Group(yellow1,yellow2,blue1,blue2)
        try:
            g.set_stroke(color=PURPLE_D, width=20)
```

(continues on next page)

(continued from previous page)

```
        except TypeError:
            print("TypeError!")
            self.add(NO)
        self.add(g)
```

```
TypeError!
```

Note: `z_index` is not supported, neither for VGroup nor for Group

[89]:
```
%%manim $params
blue1,blue2, yellow1,yellow2 = create_dots()
class Example(Scene):
    def construct(self):
        a=VGroup(yellow1,yellow2).shift(0.5*DOWN)
        b=VGroup(blue1,blue2)
        a.set_z_index(2)
        b.set_z_index(1)
        self.add(a,b)
        self.add(NO)
```

Congratulations!

You are now a master in setting up your Mobjects on a scene. Let's go on with the part you came to manim for in the first place: Animations!

1.4 Animations

There are a wide range of possibilities to animate your mobjects that all work a bit differently. Here is a broad overview so that you can choose the animation stategy that fits best for your project. This chapter will cover `ValueTrackers`, `Updaters`, `self.play Transformations` the `mobject.animate` syntax and `mobject.become` syntax.

[1]:
```
from manim import *
```

```
Manim Community v0.11.0
```

[2]:
```
#ignore this cell, only for setup
params= "-v WARNING  --progress_bar None  -r  500,200  --disable_caching Example"

NO  = Cross(Square(), stroke_color = RED_D, stroke_width = 38).scale(0.9).to_edge(LEFT,␣
↪buff=1)
YES = SVGMobject("good.svg").to_edge(LEFT, buff=1)
BEST = YES.copy()
BEST.add(Star(color= YELLOW, fill_opacity=1).scale(0.5).move_to(BEST).shift(0.5*DOWN+0.
↪5*RIGHT));
```

1.4.1 Simple Replacements

[3]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        dot= Dot(color= YELLOW, radius=0.5)
        self.add(dot)
        self.wait()
        dot.scale(2)
        self.wait()
        dot.scale(2)
        self.wait(2)
```

```
<IPython.core.display.Video object>
```

[4]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        dot= Dot(color= YELLOW, radius=0.5)
        square= Square(side_length=4,color= BLUE, fill_opacity=1)
        triangle= Triangle(radius=3,color= ORANGE, fill_opacity=1).shift(DOWN*0.5)
        self.add(dot)
        self.wait()
        dot.become(square)
        self.wait()
        dot.become(triangle)
        self.wait()
```

```
<IPython.core.display.Video object>
```

1.4.2 Using .animate Syntax

[5]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        dot= Dot(color= YELLOW, radius=0.5)
        self.play(dot.animate.scale(2))
```

```
<IPython.core.display.Video object>
```

[6]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        dot= Dot(color= YELLOW, radius=0.5)
        self.play(dot.animate.shift(2*RIGHT))
```

```
<IPython.core.display.Video object>
```

[7]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        dot= Dot(color= YELLOW, radius=0.5)
        self.play(dot.animate.set_color(BLUE))
```

```
<IPython.core.display.Video object>
```

[8]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        dot= Dot(color= YELLOW, radius=0.5)
        self.play(dot.animate.shift(2*RIGHT).scale(2))
```

```
<IPython.core.display.Video object>
```

[9]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        dot= Dot(color= YELLOW, radius=0.5)
        self.play(dot.animate.shift(2*RIGHT).scale(2).set_color(BLUE))
```

```
<IPython.core.display.Video object>
```

1.4.3 Updaters

They are very diverse! And they can be used with and without a “dt” parameter

[10]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        dot = Dot(color= GREEN, radius=0.7)
        self.add(dot)
        def foo(mob,dt):
            mob.shift(2*RIGHT*dt)
```

(continues on next page)

(continued from previous page)

```
        dot.add_updater(foo)
        self.wait(3)
```

```
<IPython.core.display.Video object>
```

[11]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        dot = Dot(color= GREEN, radius=0.7)
        self.add(dot)
        dot.add_updater(lambda x,dt: x.shift(2*RIGHT*dt))
        self.wait(3)
```

```
<IPython.core.display.Video object>
```

[12]:
```
%%manim $params
class Example(Scene): # when there is no dt parameter, the updater does not work
    def construct(self):
        dot = Dot(color= GREEN, radius=0.7)
        self.add(dot,NO)
        dot.add_updater(lambda x : x.shift(2*RIGHT*0.1))
        self.wait(3)
```

```
<IPython.core.display.Video object>
```

Note: Not using the "dt" parameter will make your animation framerate dependent, but this can be solved using ValueTracker, which can be seen in the next section

1.4.4 Updaters + ValueTrackers

[13]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        tracker= ValueTracker(0)
        dot = Dot(color= GREEN, radius=0.7)
        self.add(dot)
        def foo(mob):
            mob.move_to(RIGHT*tracker.get_value())
        dot.add_updater(foo)
        self.play(tracker.animate.set_value(2), rate_func= linear)
```

```
<IPython.core.display.Video object>
```

Note: now you can also use rate functions:

[14]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        tracker= ValueTracker(0)
        dot = Dot(color= GREEN, radius=0.7)
        self.add(dot)
        def foo(mob):
```

(continues on next page)

(continued from previous page)

```
            mob.move_to(RIGHT*tracker.get_value())
        dot.add_updater(foo)
        self.play(tracker.animate.set_value(2), rate_func= smooth)
```

```
<IPython.core.display.Video object>
```

[15]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        tracker= ValueTracker(0.5)
        dot = Dot(color= GREEN, radius=0.7)
        self.add(dot)
        def foo(mob):
            mob.move_to(RIGHT*tracker.get_value())
        dot.add_updater(foo)
        self.play(tracker.animate.set_value(2.2), rate_func= smooth)
        self.play(tracker.animate.increment_value(1), rate_func= smooth)
        self.play(tracker.animate.increment_value(-1), rate_func= smooth)
        self.play(tracker.animate.set_value(0.5), rate_func= linear)
```

```
<IPython.core.display.Video object>
```

[16]:
```
%%manim $params
#one can now also add additional properties to mobjects, in this case a counter.
class Example(Scene):
    def construct(self):
        tracker= ValueTracker(0)
        dot = Dot(color= GREEN, radius=0.7)
        self.add(dot)
        dot.counter=0
        def foo(mob):
            mob.move_to(RIGHT*tracker.get_value())
            if mob.counter == 20:
                mob.set_color(random_bright_color())
                mob.counter = 0
            mob.counter += 1
        dot.add_updater(foo)
        self.play(tracker.animate.set_value(2), rate_func= linear, run_time=3)
```

```
<IPython.core.display.Video object>
```

1.4.5 Transformations

[17]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        d= Dot(color= YELLOW, radius=0.5)
        d2= d.copy().shift(2*RIGHT)
        self.play(Transform(d, d2))
```

```
<IPython.core.display.Video object>
```

1.4.6 Does and Donts

Note that when you choose to work with updaters, your script might depend on the frame rate.

[18]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        print(f"{config.frame_rate = }fps")
        dotred= Dot(color= RED, radius=0.5).shift(UP)
        dotgreen = Dot(color= GREEN, radius=0.5)
        dotgreen.next_to(dotred,DOWN)
        self.add(dotgreen,dotred)
        DIR= 2*RIGHT
        dotgreen.add_updater(lambda x,dt: x.shift(DIR*dt))
        dotred.add_updater(lambda x,dt: x.shift(DIR*1/60))
        self.wait(3)
```

```
config.frame_rate = 60fps
```

```
<IPython.core.display.Video object>
```

[19]:
```
params5fps = "-v WARNING  --progress_bar None  --frame_rate=5 -r  500,200  --disable_
↪caching Example"
```

[20]:
```
%%manim $params5fps
class Example(Scene):
    def construct(self):
        print(f"{config.frame_rate = }fps")
        dotred= Dot(color= RED, radius=0.5).shift(UP)
        dotgreen = Dot(color= GREEN, radius=0.5)
        dotgreen.next_to(dotred,DOWN)
        self.add(dotgreen,dotred)
        DIR= 2*RIGHT
        dotgreen.add_updater(lambda x,dt: x.shift(DIR*dt))
        dotred.add_updater(lambda x,dt: x.shift(DIR*1/60))
        self.wait(3)
```

```
config.frame_rate = 5.0fps
```

```
<IPython.core.display.Video object>
```

Rotation animation

There are multiple ways to rotate a square, but not all will result in that animation that you might have expected.

[21]:
```
%%manim $params

class Example(Scene):
    def construct(self, **kwargs):
        s1= Square().set_color(YELLOW)
```

(continues on next page)

(continued from previous page)

```
        self.add(s1, BEST)
        self.play(Rotate(s1, angle=PI/2))
```

```
<IPython.core.display.Video object>
```

[22]:
```
%%manim $params
class Example(Scene):
    def construct(self, **kwargs):
        s2= Square().set_color(PURPLE)
        self.add(s2, NO)
        self.play(s2.animate.rotate(PI/2))
```

```
<IPython.core.display.Video object>
```

[23]:
```
%%manim $params

class Example(Scene):
    def construct(self, **kwargs):
        theta_track= ValueTracker(0)
        s3= Square().set_color(ORANGE)
        self.add(s3, YES)
        s3.previous_angle=0
        def pref(x):
            x.previous_angle=theta_track.get_value()
        s3.add_updater(lambda x: x.rotate(theta_track.get_value()-s3.previous_angle))
        s3.add_updater(pref)
        self.play(theta_track.animate.increment_value(PI/2))
```

```
<IPython.core.display.Video object>
```

[24]:
```
#not yet implemented
#class Example(Scene):
#    def construct(self, **kwargs):
        #s3b= Square().set_color(YELLOW)
        #self.add(s3b)
        #theta_track= DeltaValueTracker(0)
        #s3b.add_updater(lambda x: x.rotate(theta_track.get_delta_value()))
        #self.play(theta_track.animate.set_value(90*DEGREES))
```

[25]:
```
%%manim $params
# NOT WORKING!, BAD PRACTICE.
class Example(Scene):
    def construct(self, **kwargs):
        s4= Square().set_color(GREEN)
        self.add(s4, NO)
        theta_track= ValueTracker(0)
        s4.add_updater(lambda x: x.rotate(theta_track.get_value()))
        self.play(theta_track.animate.increment_value(PI/2))
```

```
<IPython.core.display.Video object>
```

[26]:
```
%%manim $params
class Example(Scene):
    def construct(self, **kwargs):
        s6= Square().set_color(PINK)
        self.add(s6, YES)
        s6.add_updater(lambda x, dt: x.rotate(dt*PI/2))
        self.wait(1)
```

```
<IPython.core.display.Video object>
```

Known bugs

Bug with updaters that do not have a dt

[27]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        dot = Dot(color= GREEN, radius=0.7)
        self.add(dot,NO)

        #dot.add_updater(lambda x,dt : x)

        dot.add_updater(lambda x : x.shift(2*RIGHT*1/config.frame_rate))
        self.wait(3)
```

```
<IPython.core.display.Video object>
```

[28]:
```
%%manim $params
class Example(Scene):
    def construct(self):
        dot = Dot(color= GREEN, radius=0.7)
        self.add(dot,YES)

        dot.add_updater(lambda x,dt : x) #adding this line will make the updater␣
↪continiously watch

        dot.add_updater(lambda x : x.shift(2*RIGHT*1/config.frame_rate))
        self.wait(3)
```

```
<IPython.core.display.Video object>
```

Bugs with updater in ZoomedScene

[29]:
```
%%manim $params
class Example(ZoomedScene):
    def __init__(self, **kwargs):
        ZoomedScene.__init__(
            self,
            zoom_factor=0.3,
            zoomed_display_height=4,
```

(continues on next page)

(continued from previous page)

```
            zoomed_display_width=4,
            image_frame_stroke_width=20,
            zoomed_camera_config={
                "default_frame_stroke_width": 3,
            },
            **kwargs
        )
    def construct(self):
        d= Dot()
        self.add(d)
        imgo =Square().scale(0.3).set_color(RED)
        self.add(imgo)
        #imgo.add_updater(lambda x: x) # COMMENTED OUT
        self.activate_zooming(animate=True)
        self.play(self.zoomed_camera.frame.animate.shift(0.5 * (LEFT+UP*0.8)))
        self.play(self.zoomed_camera.frame.animate.shift(0.5 * (RIGHT+DOWN*2.8)))
```

```
<IPython.core.display.Video object>
```

[30]:
```
%%manim $params
class Example(ZoomedScene):
    def __init__(self, **kwargs):
        ZoomedScene.__init__(
            self,
            zoom_factor=0.3,
            zoomed_display_height=4,
            zoomed_display_width=4,
            image_frame_stroke_width=20,
            zoomed_camera_config={
                "default_frame_stroke_width": 3,
            },
            **kwargs
        )
    def construct(self):
        d= Dot()
        self.add(d)
        imgo =Square().scale(0.3).set_color(RED)
        self.add(imgo)
        imgo.add_updater(lambda x: x) # INCLUDED
        self.activate_zooming(animate=True)
        self.play(self.zoomed_camera.frame.animate.shift(0.5 * (LEFT+UP*0.8)))
        self.play(self.zoomed_camera.frame.animate.shift(0.5 * (RIGHT+DOWN*2.8)))
```

```
<IPython.core.display.Video object>
```

[ ]:

1.5 Resolution and Camera

[1]:
```
from manim import *
```

```
Manim Community v0.11.0
```

1.5.1 Scene Coordinates

First, let's learn a bit about how manim coordinates work.
There is the config.frame_width, config.frame_height which is unrelated to the pixelsize.
Their default values are 14.222 and 8.
These values are chosen, because it gives and width/height ratio of 16/9, which is a common screen resolution.
The coordinate center of scenes is in the center, which is at **(0,0)**.
The most left point is **(-7.1,0)**, right is **(7.1,0)**, top is **(0,4)**, and button is **(0,-4)**.

[2]:
```
config.frame_width/config.frame_height
```

[2]:
```
1.7777777777777777
```

[3]:
```
config.pixel_width/config.pixel_height
```

[3]:
```
1.7777777777777777
```

[4]:
```
16/9
```

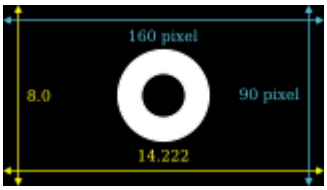
[4]:
```
1.7777777777777777
```

[5]:
```
# for setup only
def yellow_frame_annotation(framew, frameh):
    d1 = DoubleArrow(framew * LEFT / 2, framew * RIGHT / 2, buff=0).to_edge(DOWN)
    t1 = Text(str(framew)[:6]).next_to(d1, UP)
    d2 = DoubleArrow(frameh * UP / 2, frameh * DOWN / 2, buff=0).to_edge(LEFT)
    t2= Text(str(frameh)).next_to(d2, RIGHT)
    x=Group(d1,d2,t1,t2).set_color(YELLOW)
    return x

def blue_pixel_annotation(framew, frameh,pixelw, pixelh):
    d1 = DoubleArrow(framew * LEFT / 2, framew * RIGHT / 2, buff=0).to_edge(UP)
    t1 = Text(str(pixelw) + " pixel").next_to(d1, DOWN)
    d2 = DoubleArrow(frameh * UP / 2, frameh * DOWN / 2, buff=0).to_edge(RIGHT)
    t2= Text(str(pixelh) + " pixel").next_to(d2, LEFT)
    x=Group(d1,d2,t1,t2).set_color(BLUE)
    return x

annulus = Annulus(inner_radius =1,outer_radius=2,color=WHITE,  stroke_width=10)
```

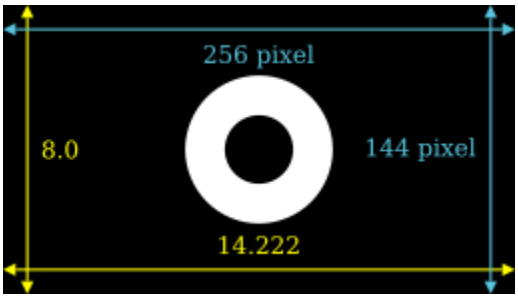
Pixel Ratio of 16/9

See a table of commen 16/9 resolutions here: https://en.wikipedia.org/wiki/16:9_aspect_ratio#Common_resolutions

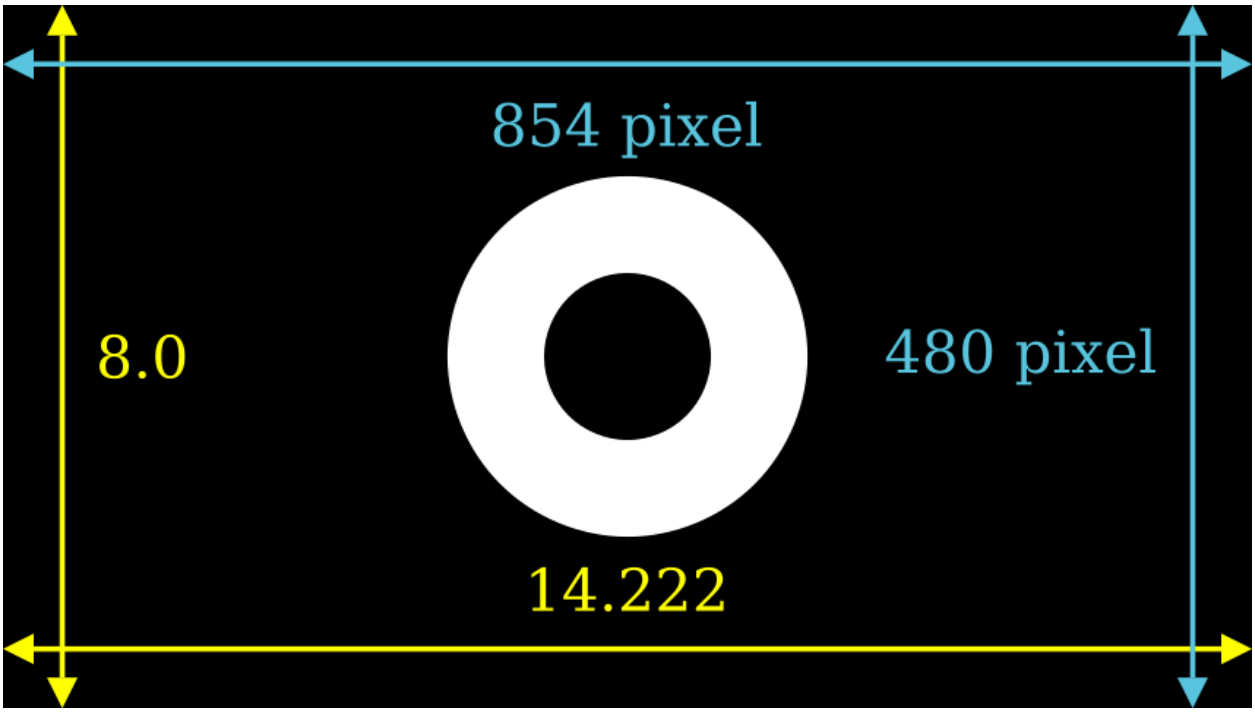
[6]:
```
%%manim -v WARNING  -s -r 160,90 --disable_caching Example
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
        self.add(frame_annotation, pixel_annotation, annulus)
```



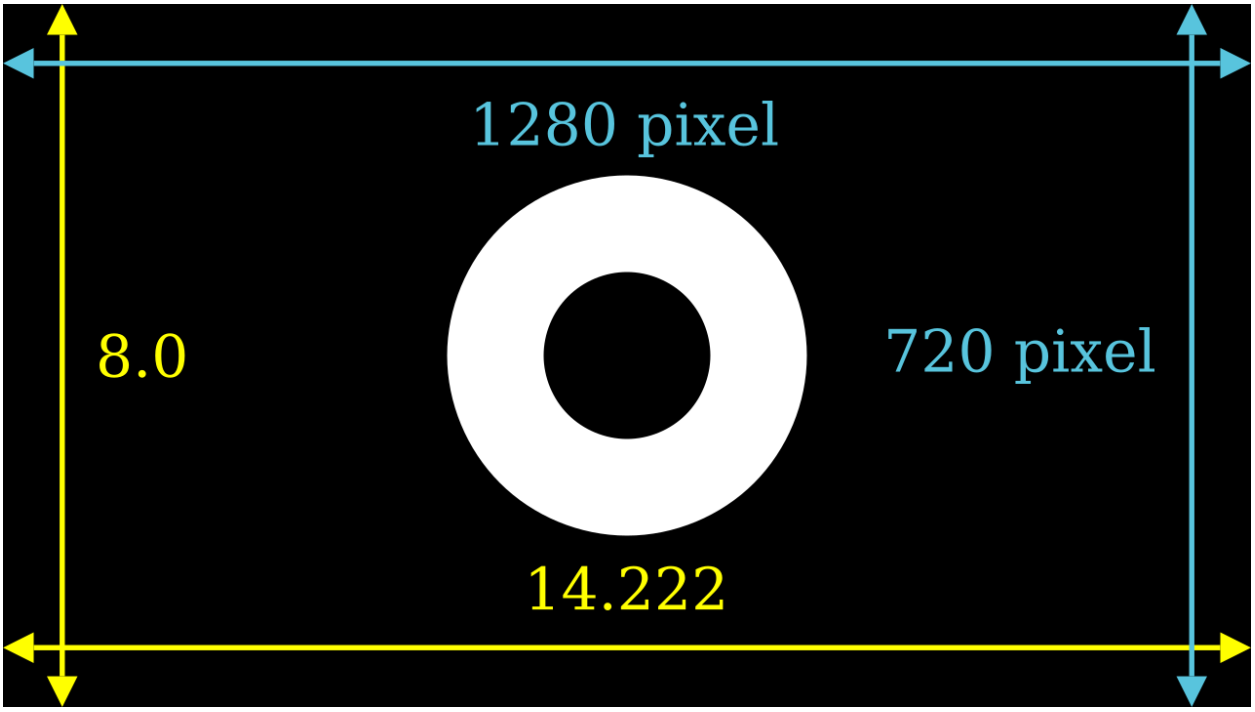
[7]:
```
%%manim -v WARNING  -s -r 256,144 --disable_caching Example
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
        self.add(frame_annotation, pixel_annotation, annulus)
```



[8]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
        self.add(frame_annotation, pixel_annotation, annulus)
```



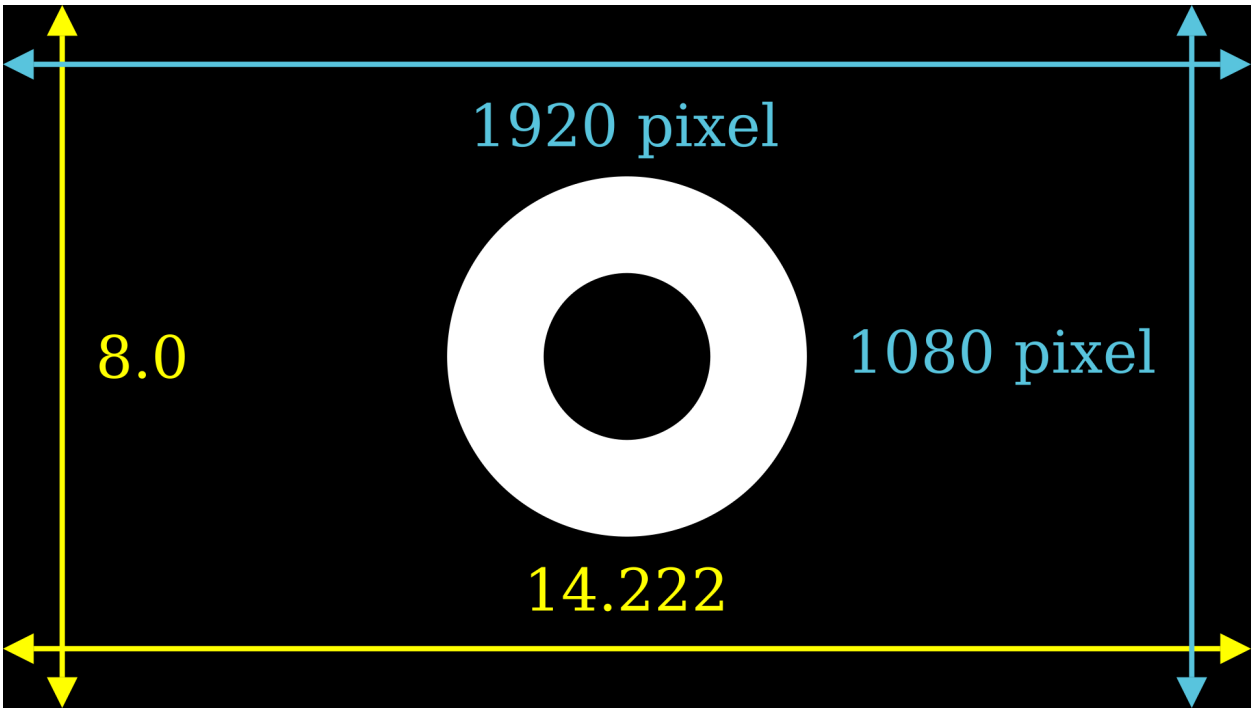
[9]:
```
%%manim -v WARNING  -s -qm --disable_caching Example
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
        self.add(frame_annotation, pixel_annotation, annulus)
```



Note

The borders of this website are narrow.
To see the changes in high resolution, open this image in a new tab.

[10]:
```
%%manim -v WARNING  -s -qh --disable_caching Example
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
        self.add(frame_annotation, pixel_annotation, annulus)
```



Pixel Ratio Unequal to 16/9

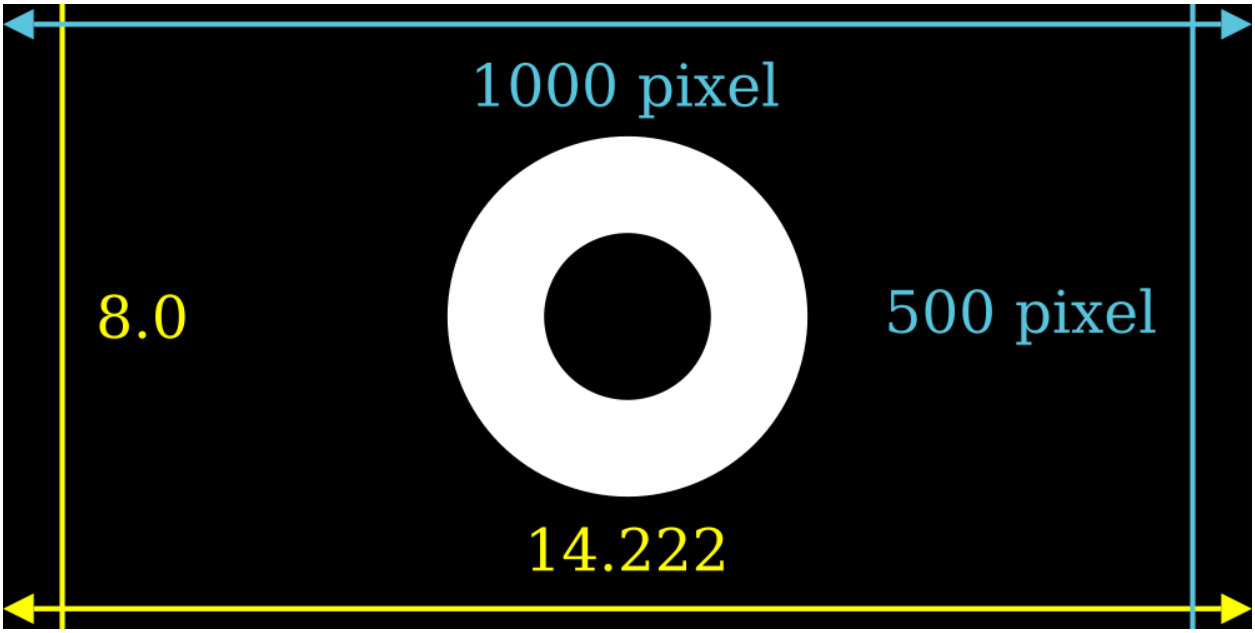
- When the pixel ratio is heigher then 16/9 frame_height cropped.
- When the pixel ratio is lower then 16/9 frame_height padded.

[11]:
```
%%manim -v WARNING  -s -r 1000,500 --disable_caching Example
#ratio of 2/1
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
```

(continues on next page)

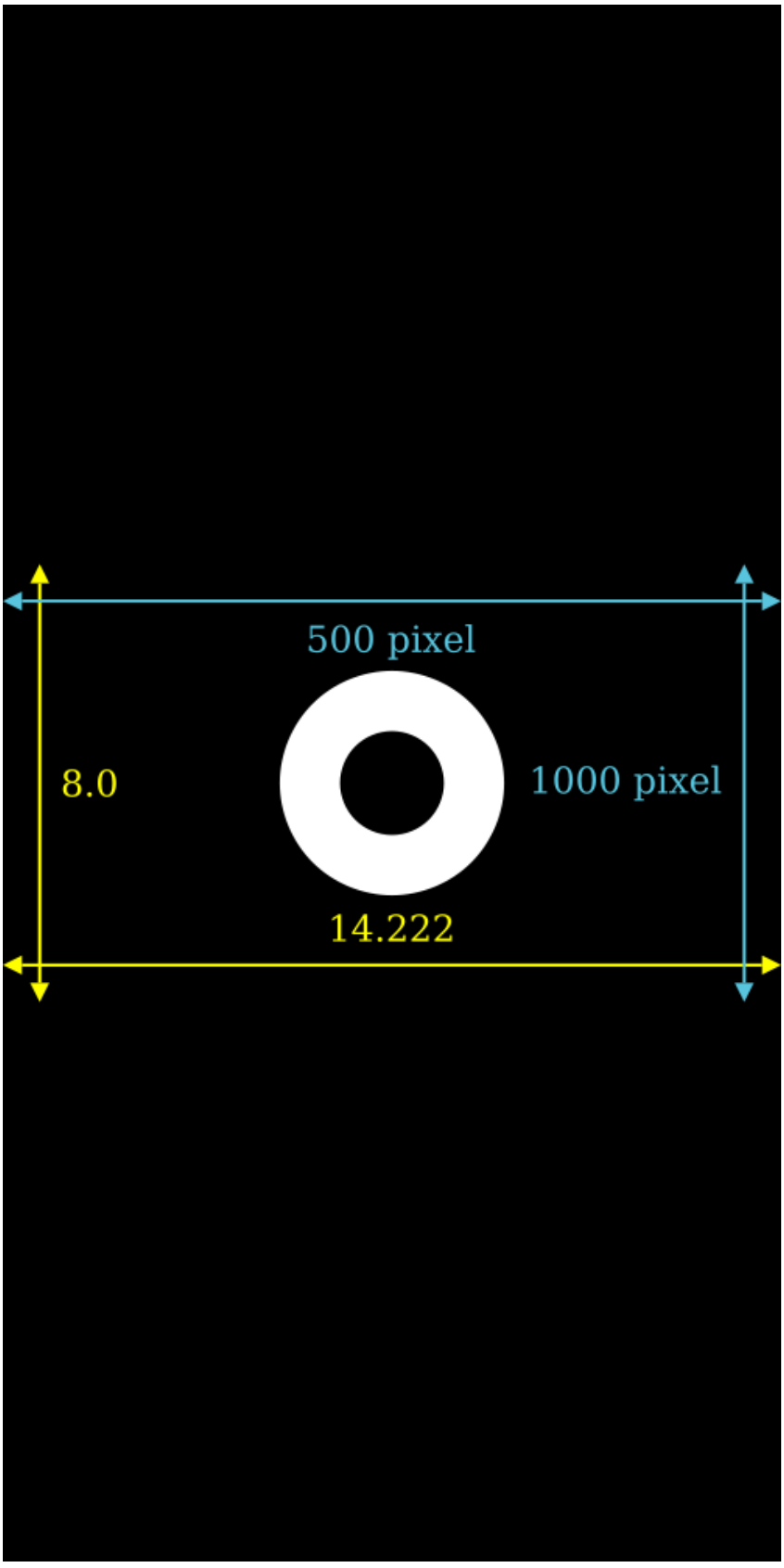
(continued from previous page)

```
        self.add(frame_annotation, pixel_annotation, annulus)
```



[12]:
```
%%manim -v WARNING  -s -r 1000,50 --disable_caching Example
#ratio of 20/1
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
        self.add(frame_annotation, pixel_annotation, annulus)
```

[13]:
```
%%manim -v WARNING  -s -r 500,1000 --disable_caching Example
#ratio of 1/2
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
        self.add(frame_annotation, pixel_annotation, annulus)
```

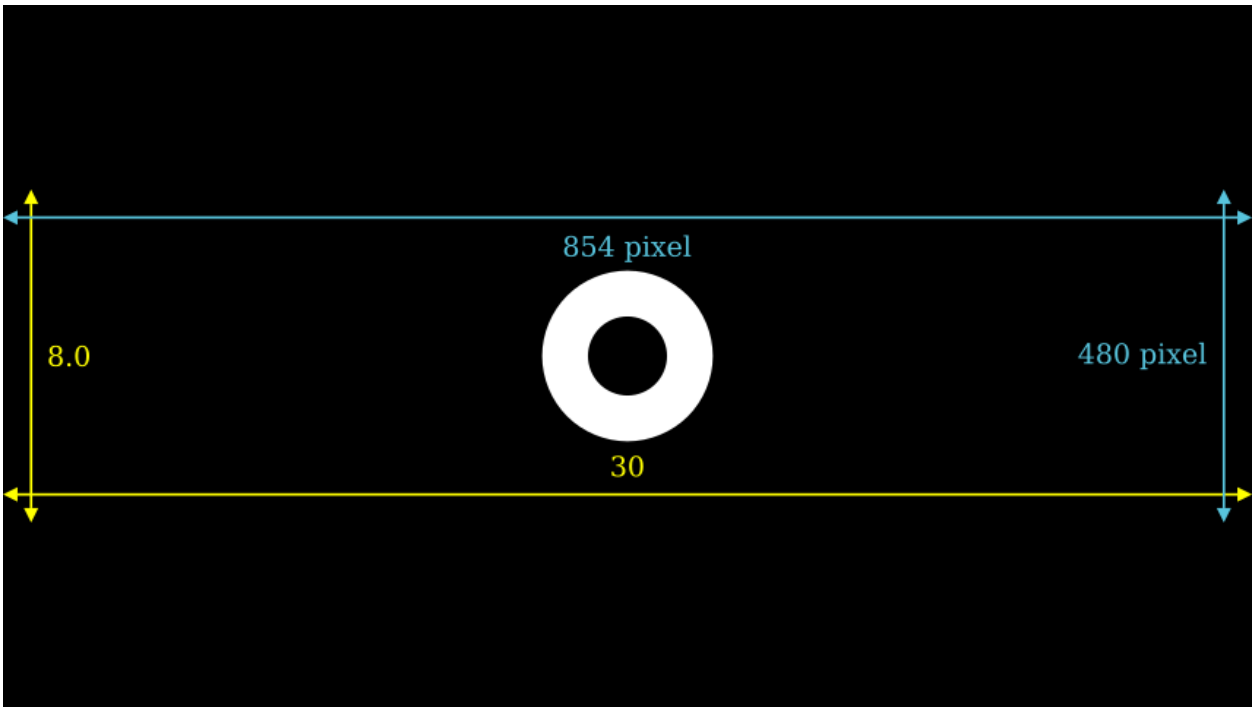

500 pixel
8.0
1000 pixel
14.222

1.5.2 Changing the frame_width

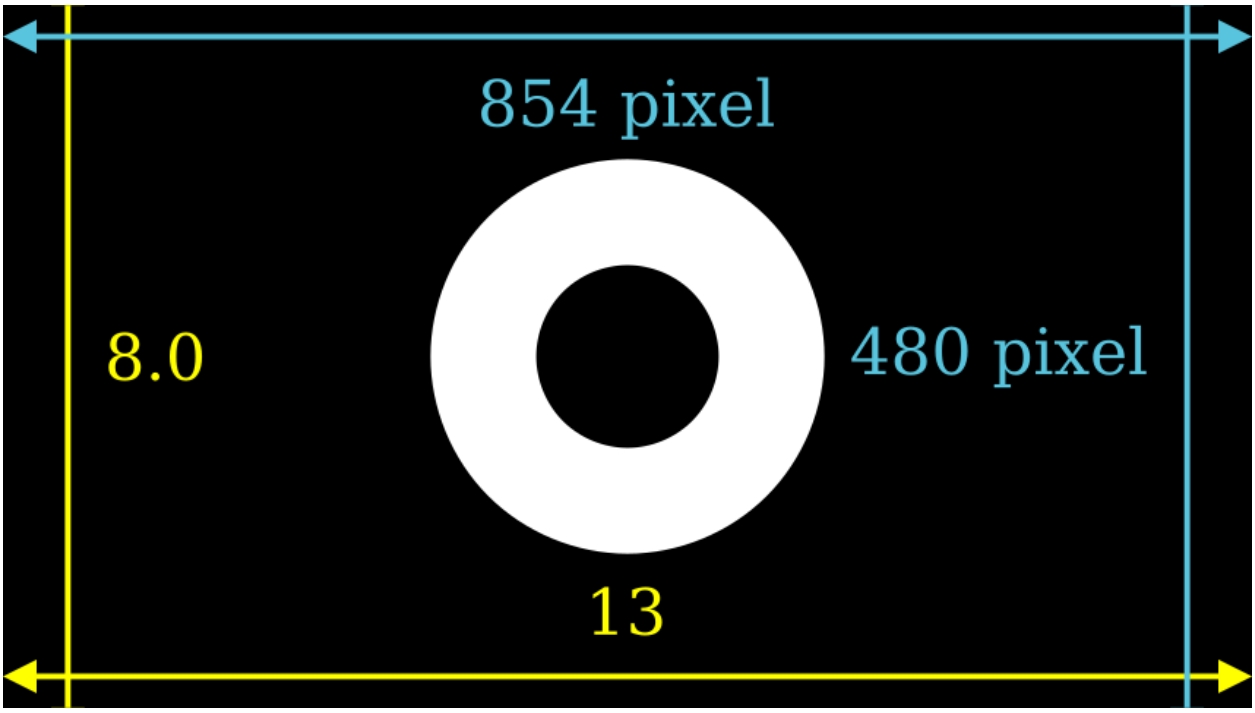
- **Increasing** the value `config.frame_width` will **zoom out** the Mobject
- **Decreasing** the value `config.frame_width` will **zoom in** the Mobject

Note: The frame_height is adjusted accordingly. Note 2: I do not recommend to change the frame width with config.frame_width, better use the `self.camera.frame.set(...)` syntax shown in the next section.

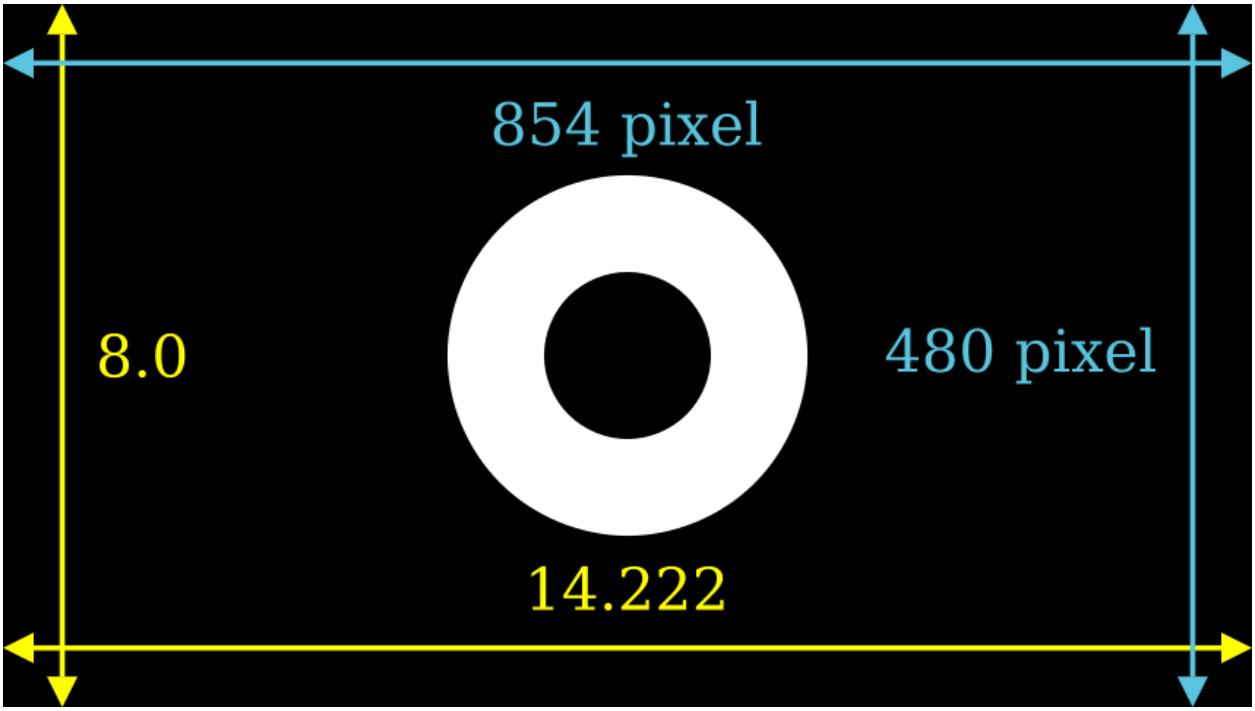
[14]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
config.frame_width =30
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
        self.add(frame_annotation, pixel_annotation, annulus)
```



[15]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
config.frame_width =13
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
        self.add(frame_annotation, pixel_annotation, annulus)
```



[16]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
config.frame_width =14.22222
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
        self.add(frame_annotation, pixel_annotation, annulus)
```



Note

Changing `config.frame_height` has no effect on the Mobjects displaied on the screen.

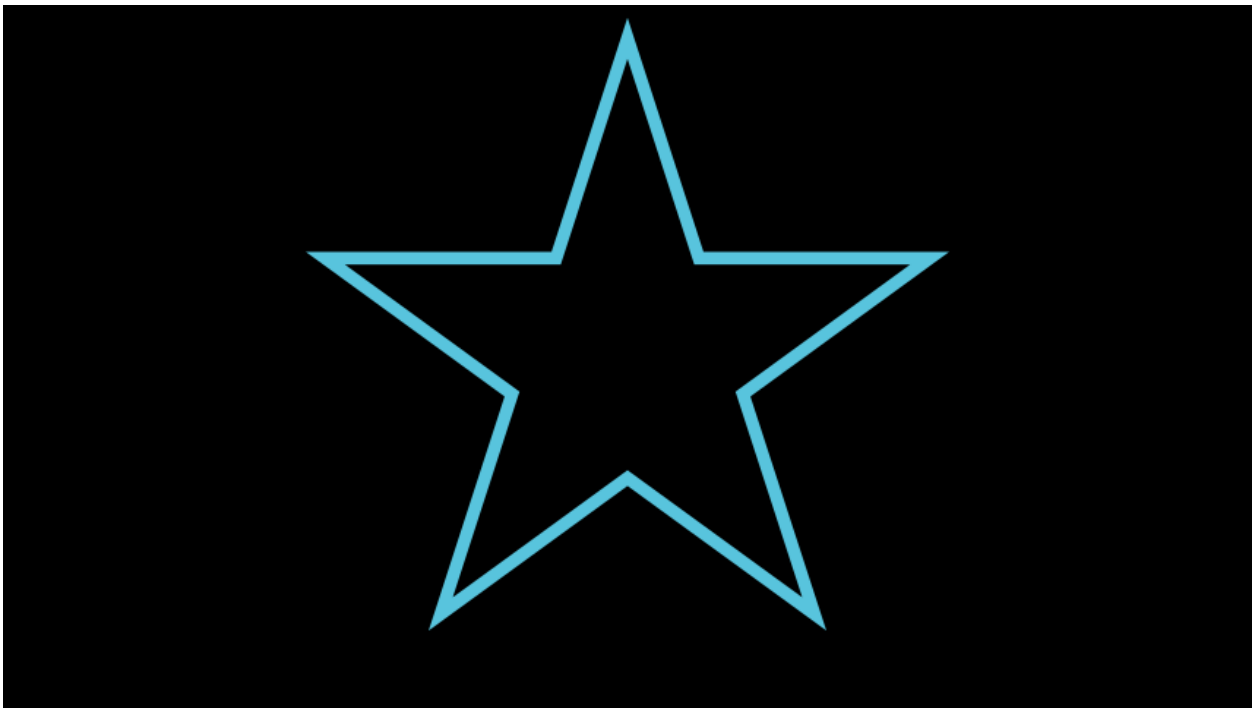
[17]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
config.frame_height =42
class Example(Scene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        pixel_annotation= blue_pixel_annotation(config.frame_width,config.frame_height,
↪config.pixel_width,config.pixel_height)
        self.add(frame_annotation, pixel_annotation, annulus)
```

[18]:
```
config.frame_height =8 # resetting the frame_height value to default
```

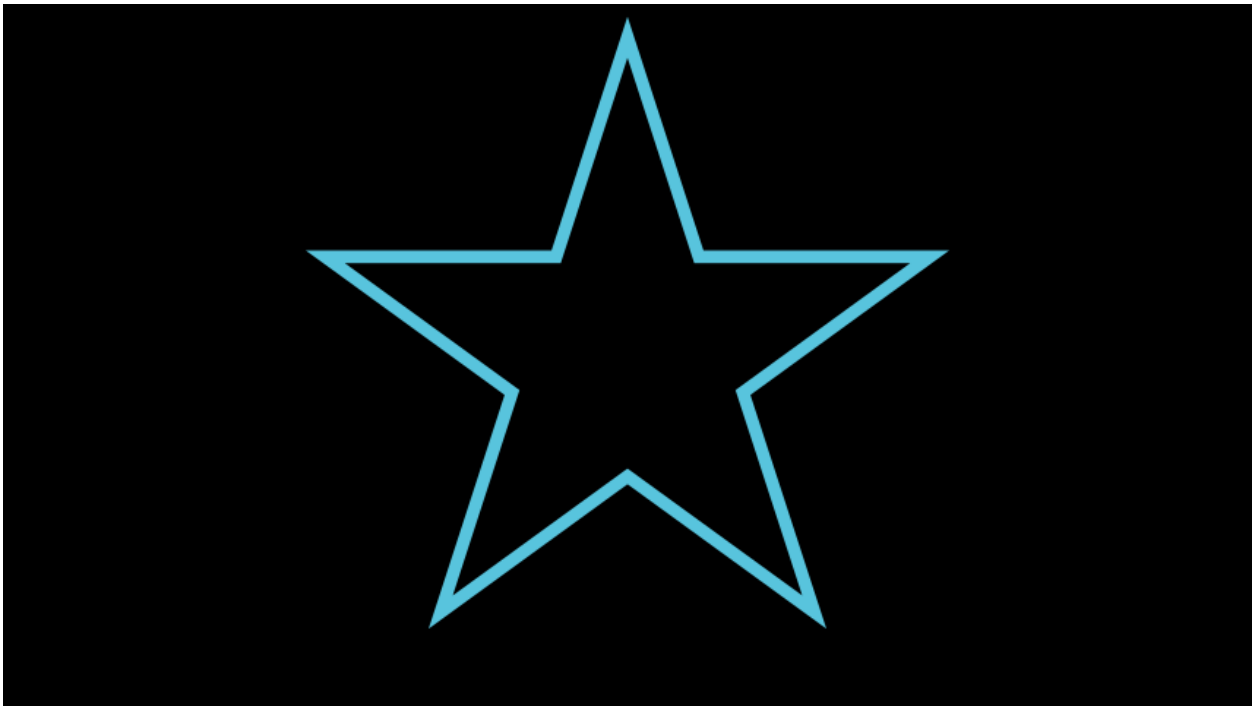
1.5.3 Camera Scene

[19]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
class Example(MovingCameraScene):
    def construct(self):
        s= Star()
        self.camera.frame.set(height=s.get_height()+4*SMALL_BUFF)
        self.add(s)
```

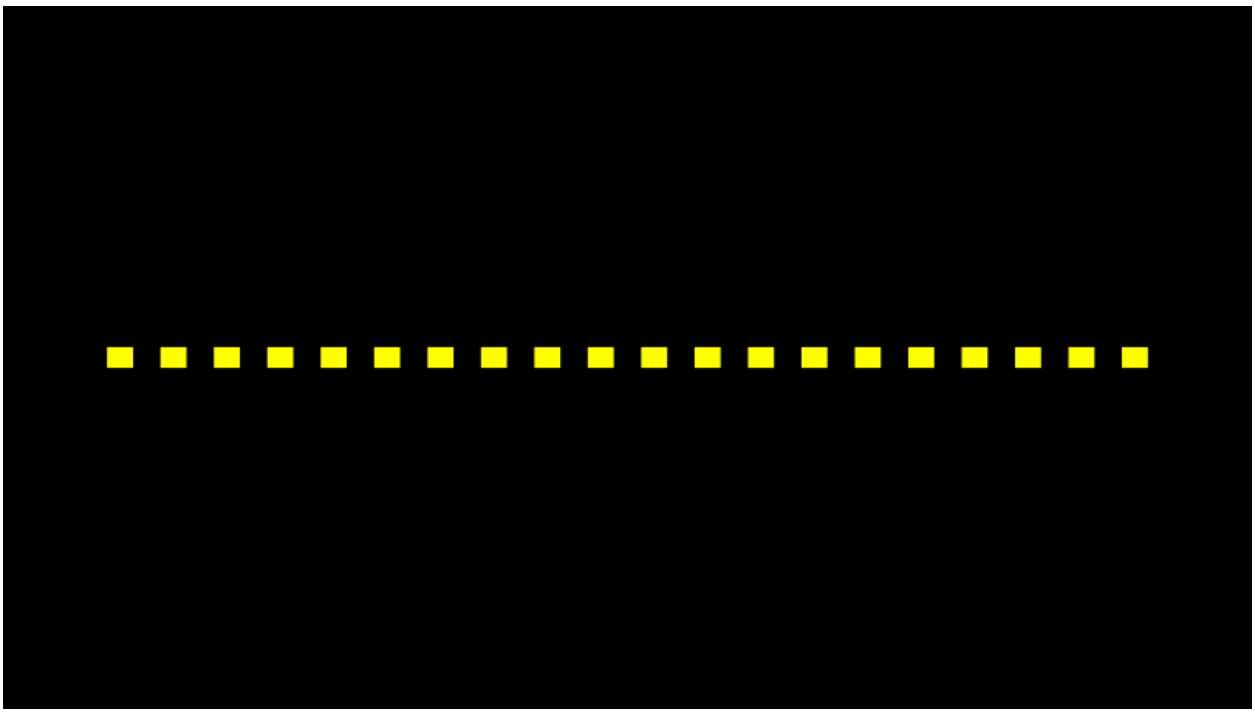
```
<string>:4: DeprecationWarning: This method is not guaranteed to stay around. Please␣
↪prefer getting the attribute normally.
```



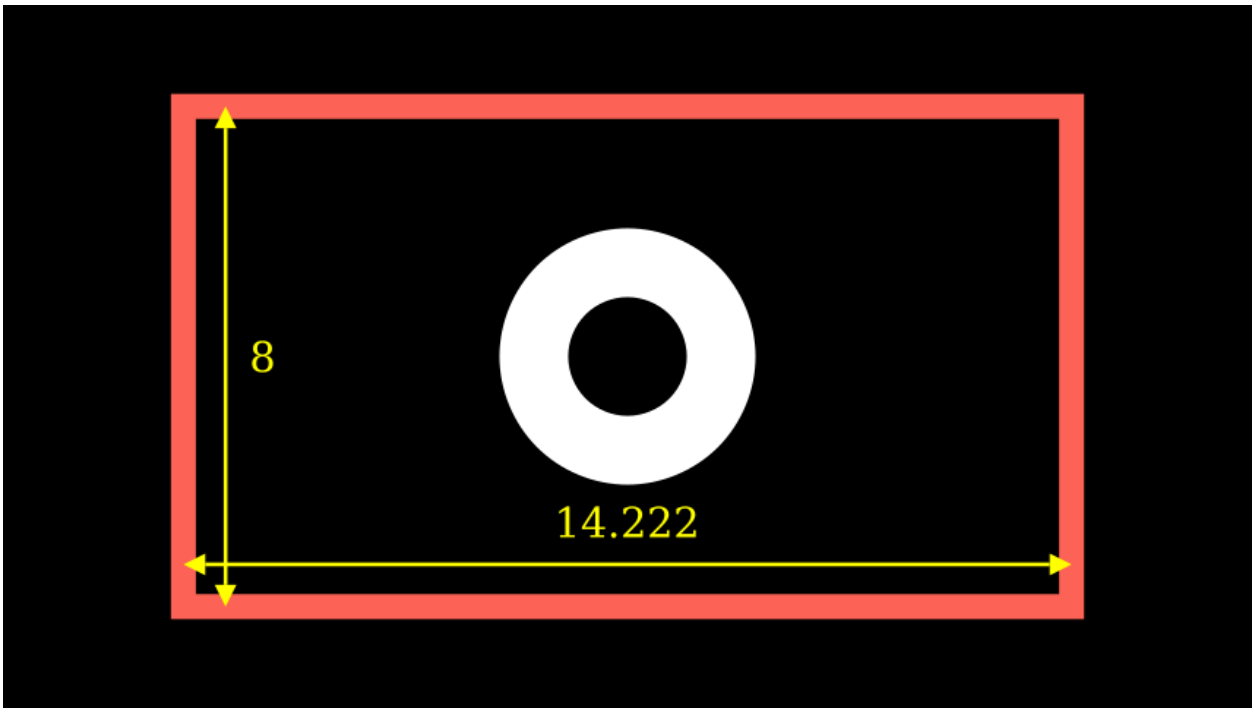
[20]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
class Example(MovingCameraScene):
    def construct(self):
        s= Star()
        self.camera.frame.height=s.height+4*SMALL_BUFF # even better
        self.add(s)
```



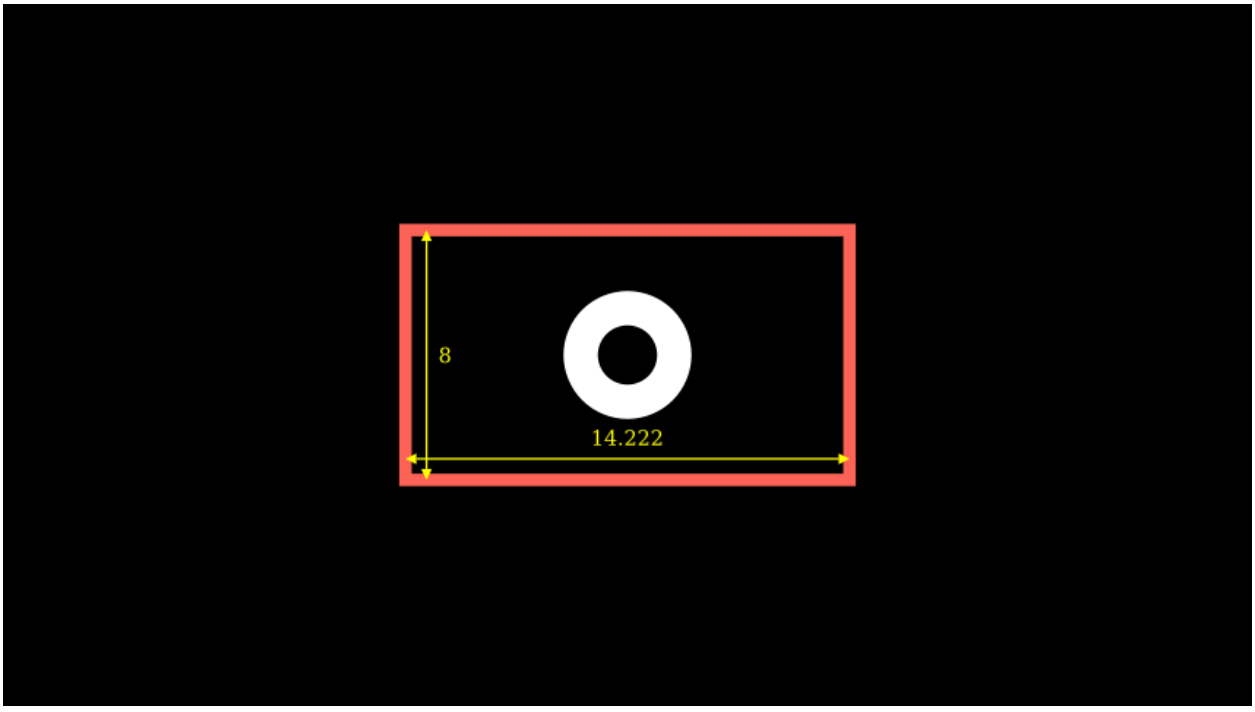
[21]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
class Example(MovingCameraScene):
    def construct(self):
        mob = DashedLine(color= YELLOW)
        self.camera.frame.width=mob.width +4*SMALL_BUFF
        self.add(mob)
```



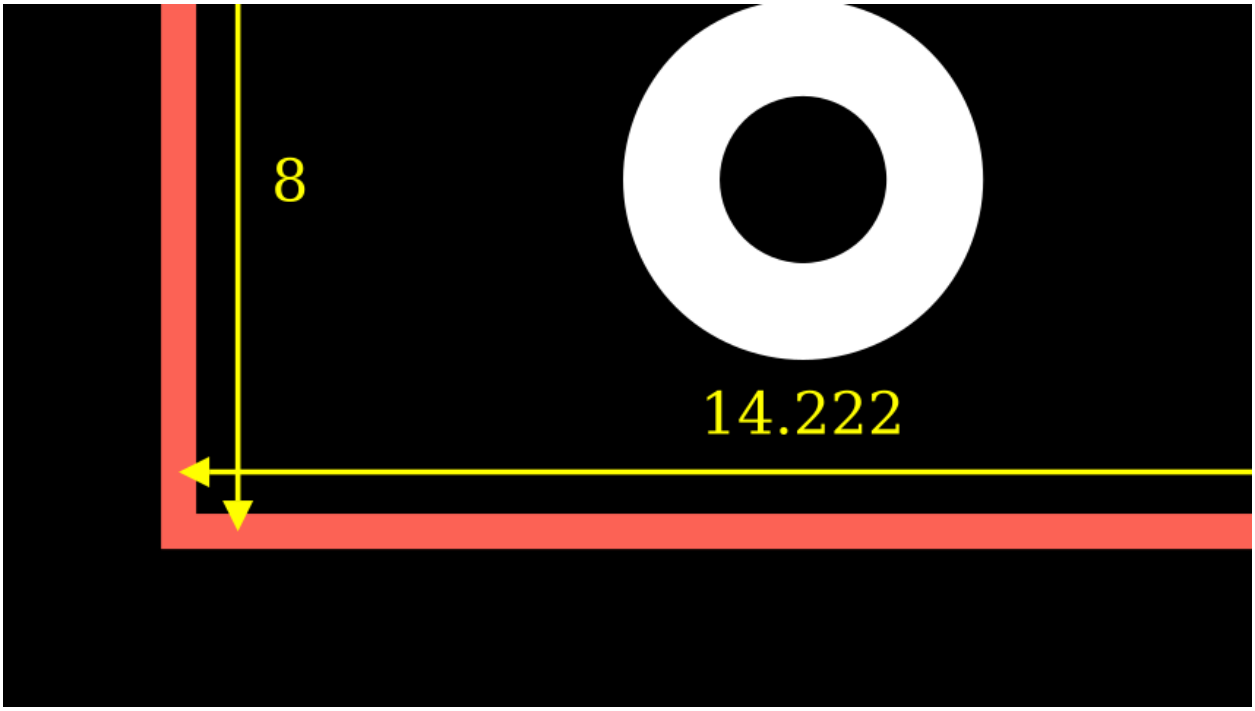
[22]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
class Example(MovingCameraScene):
    def construct(self):
        self.camera.frame.set(width=20)
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        self.add(FullScreenRectangle(color=RED, stroke_width=40))
        self.add(frame_annotation, annulus)
```



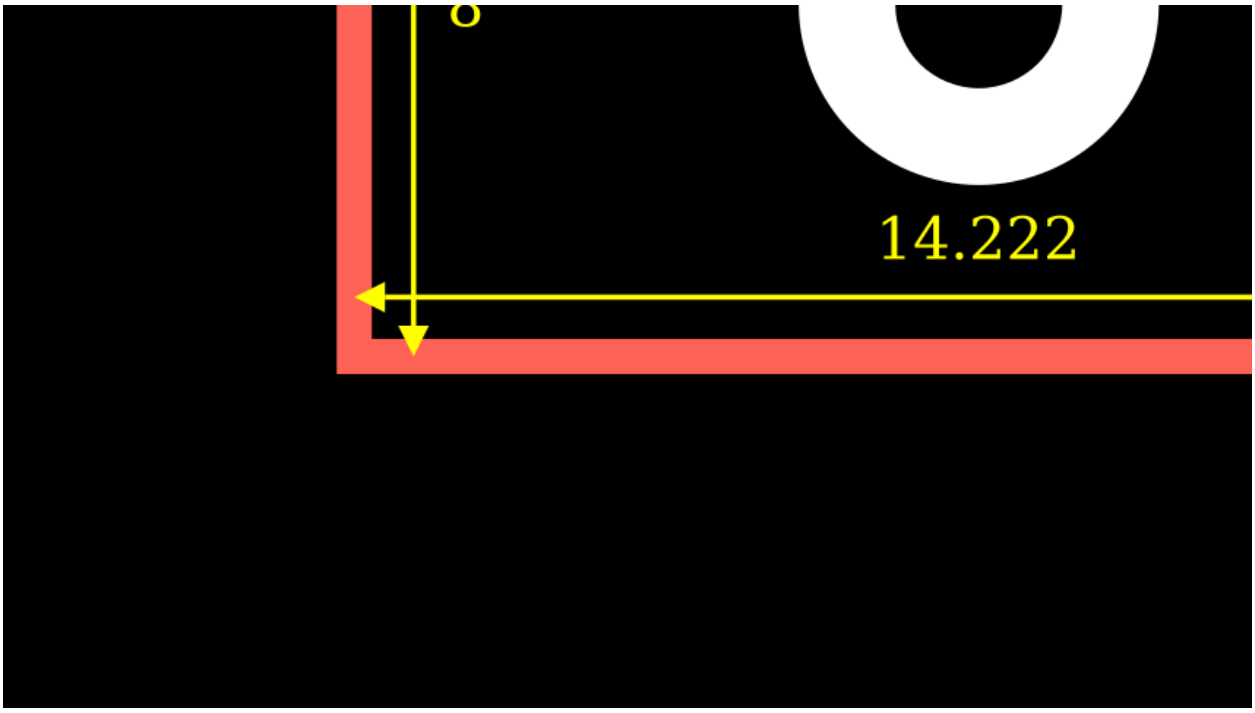
[23]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
class Example(MovingCameraScene):
    def construct(self):
        self.camera.frame.set(width=40)
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        self.add(FullScreenRectangle(color=RED, stroke_width=40))
        self.add(frame_annotation, annulus)
```



[24]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
class Example(MovingCameraScene):
    def construct(self):
        self.camera.frame.shift(2*DOWN+2*LEFT)
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        self.add(FullScreenRectangle(color=RED, stroke_width=40))
        self.add(frame_annotation, annulus)
```



[25]:
```
%%manim -v WARNING  -s -ql --disable_caching Example
class Example(MovingCameraScene):
    def construct(self):
        self.camera.frame.shift(4*DOWN+4*LEFT)
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        self.add(FullScreenRectangle(color=RED, stroke_width=40))
        self.add(frame_annotation, annulus)
```



[26]:
```
%%manim -v WARNING   -ql --disable_caching Example
class Example(MovingCameraScene):
    def construct(self):
        frame_annotation= yellow_frame_annotation(config.frame_width,config.frame_height)
        self.add(FullScreenRectangle(color=RED, stroke_width=40))
        self.add(frame_annotation, annulus)

        self.play(self.camera.frame.animate.shift(UP+2*LEFT).set(width=20))
        self.play(self.camera.frame.animate.shift(2*DOWN+4*RIGHT))

        self.play(self.camera.frame.animate.move_to(ORIGIN).set(width=14.222))
```

```
<IPython.core.display.Video object>
```

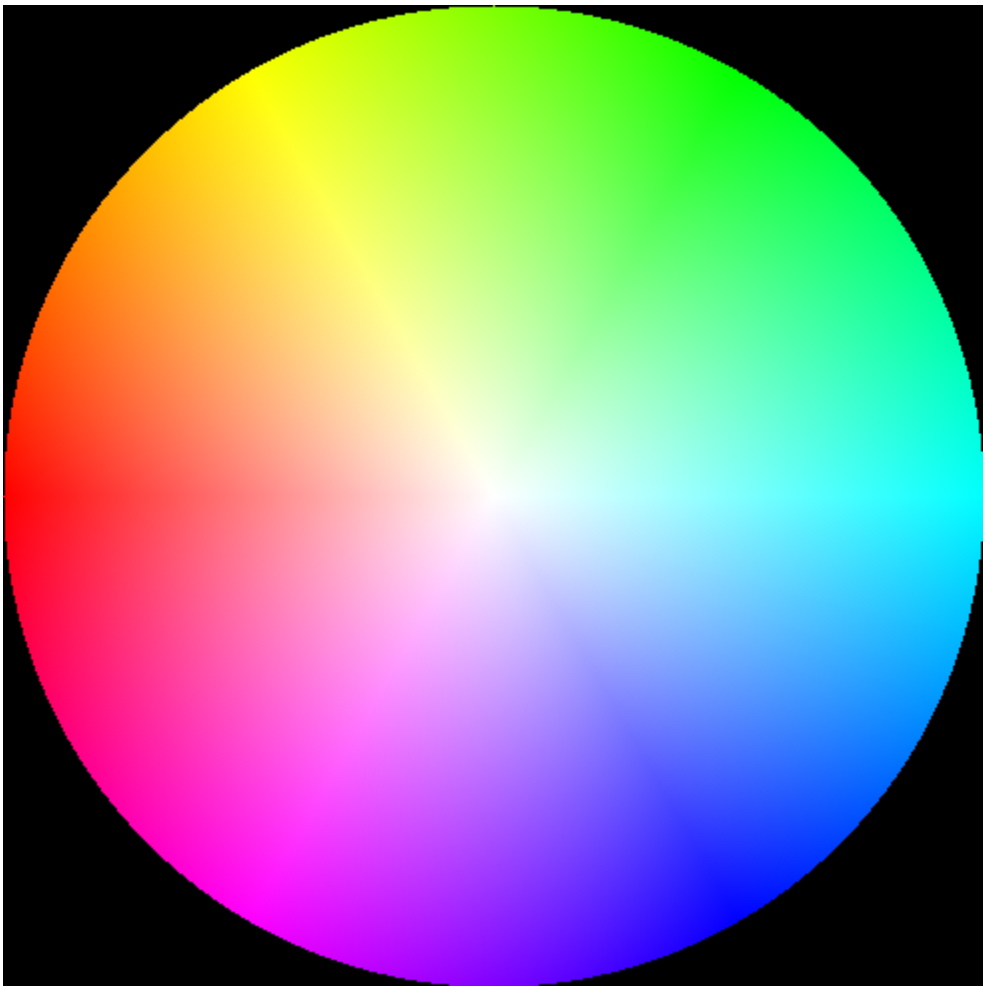
1.6 Color Wheel Tutorial

In this notebook, you will learn how to create a color picker with a moving wheel in manim (scroll to the end to see the result)

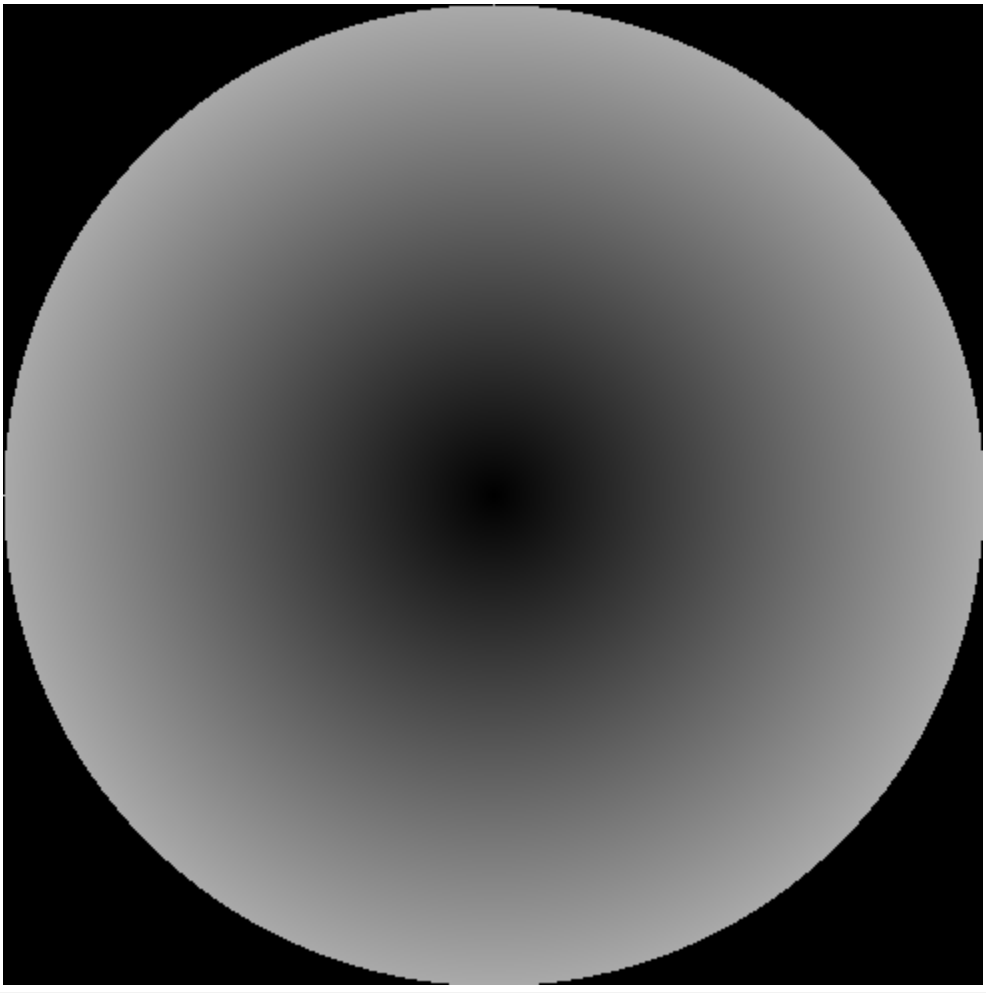
[1]:
```
from manim import *
from PIL import Image
import colorsys
import math
#from manim.utils.color import Colors
from colorutils import hsv_to_hex,hex_to_hsv
```

```
Manim Community v0.11.0
```

[2]:
```
size=490
im = Image.new("RGB", (size,size))
radius = min(im.size)/2.0
cx, cy = im.size[0]/2, im.size[1]/2
pix = im.load()

for x in range(im.width):
    for y in range(im.height):
        rx = x - cx
        ry = y - cy
        s = (rx ** 2.0 + ry ** 2.0) ** 0.5 / radius
        if s <= 1.0:
            h = ((math.atan2(ry, rx) / math.pi) + 1.0) / 2.0
            rgb = colorsys.hsv_to_rgb(h, s, 1)
            pix[x,y] = tuple([int(round(c*255.0)) for c in rgb])
hsv_hue_sat = im
display(hsv_hue_sat)
```



[3]:
```
im = Image.new("RGB", (size,size))
radius = min(im.size)/2.0
cx, cy = im.size[0]/2, im.size[1]/2
pix = im.load()

for x in range(im.width):
    for y in range(im.height):
        rx = x - cx
        ry = y - cy
        s = (rx ** 2.0 + ry ** 2.0) ** 0.5 / radius
        if s <= 1.0:
            h = ((math.atan2(ry, rx) / math.pi) + 1.0) / 2.0
            rgb = colorsys.hsv_to_rgb(0, s, 1 )
            rgb = [np.mean(rgb)]*3
            pix[x,y] = tuple([int(255-round(c*255.0)) for c in rgb])
hsv_value = im
display(hsv_value)
```



[4]:
```
class ColorWheels(Group):
    def __init__(self, **kwargs):
        Group.__init__(self, **kwargs)
```

(continues on next page)

(continued from previous page)

```
        im_hue = ImageMobject(hsv_hue_sat).set_z_index(-5)
        im_val = ImageMobject(hsv_value).set_z_index(-5)
    #    im_hue = Circle(radius=1.5).set_style(fill_color=WHITE, fill_opacity=1).set_z_
↪index(-5)
    #    im_val = Circle(radius=1.5).set_style(fill_color=WHITE, fill_opacity=1).set_z_
↪index(-5)
        self.radius = im_hue.height/2
        self.add(im_hue, im_val)
        Group(*self.submobjects).arrange(DOWN, SMALL_BUFF*1.3).to_edge(RIGHT)
        t1= Text("Hue and Saturation").scale(0.3)
        t1.next_to(im_hue, UP, buff=SMALL_BUFF).rotate(35*DEGREES, about_point=im_hue.
↪get_center())
        self.add(t1)
        t2= Text("Value").scale(0.3)
        t2.next_to(im_val, UP, buff=SMALL_BUFF).rotate(35*DEGREES, about_point=im_val.
↪get_center())
        self.add(t2)
        global CENTER_HUE , CENTER_VAL
        CENTER_HUE = im_hue.get_center()
        CENTER_VAL = im_val.get_center()
```

[5]:
```
class HueValSlider(Group):
    def __init__(self, wheels, h, s, v,**kwargs):
        hue_tracker= ValueTracker(h)
        sat_tracker= ValueTracker(s)
        val_tracker= ValueTracker(v)
        self.hue_tracker= hue_tracker
        self.sat_tracker= sat_tracker
        self.val_tracker= val_tracker

        Group.__init__(self, **kwargs)
        hue_dot = Dot(CENTER_HUE+LEFT).set_color(BLACK).scale(0.8).set_z_index(1)
        hue_line = Line(CENTER_HUE, hue_dot.get_center()).set_color(BLACK).set_
↪stroke(width=2)
        self.hue_line =hue_line
        hue_circ= Circle().set_color(BLACK).scale(0.08).move_to(hue_dot.get_center())
        hue_dot.add_updater(lambda x: x.move_to(CENTER_HUE+wheels.radius*sat_tracker.get_
↪value()* np.array([-np.cos(hue_tracker.get_value()*DEGREES),np.sin(hue_tracker.get_
↪value()*DEGREES),0])))
        hue_dot.add_updater(lambda x: x.set_color(hsv_to_hex((hue_tracker.get_value()
↪%360, sat_tracker.get_value(),val_tracker.get_value()))))
        hue_line.add_updater(lambda x: x.put_start_and_end_on(CENTER_HUE, hue_dot.get_
↪center()))

        hue_circ.add_updater(lambda x: x.move_to(hue_dot.get_center()))
        self.add(hue_dot, hue_circ, hue_line)

        val_dot = Dot(CENTER_VAL+LEFT).set_color(BLACK).scale(0.8).set_z_index(1)
        val_line = Line(CENTER_VAL, val_dot.get_center()).set_color(BLACK).set_
↪stroke(width=2)
        val_circ= Circle().set_color(BLACK).scale(0.08).move_to(val_dot.get_center())
        val_dot.add_updater(lambda x: x.move_to(CENTER_VAL+wheels.radius*val_tracker.get_
↪value()* np.array([-np.cos(hue_tracker.get_value()*DEGREES),np.sin(hue_tracker.get_
↪value()*DEGREES),0])))
```

(continues on next page)

(continued from previous page)

```
        val_dot.add_updater(lambda x: x.set_color(hsv_to_hex((hue_tracker.get_value()
↪%360, sat_tracker.get_value(),val_tracker.get_value()))))
        val_line.add_updater(lambda x: x.put_start_and_end_on(CENTER_VAL, val_dot.get_
↪center()))
        val_circ.add_updater(lambda x: x.move_to(val_dot.get_center()))
        self.add(val_dot, val_circ, val_line)
```

[6]:
```
%%manim  -v WARNING -qm  --disable_caching Idea3

class Idea3(Scene):
    def construct(self):
        wheels = ColorWheels()
        self.add(wheels)
        t1= Dot().scale(4)
        t2= Dot().scale(4)

        t3 = Dot().scale(4)
        gr = VGroup(t1,t2,t3).arrange(DOWN)
        self.add(gr)

        t1.add_updater(lambda x: x.set_color(hsv_to_hex((huevals1.hue_tracker.get_value()
↪%360, huevals1.sat_tracker.get_value(),1))))
        t2.add_updater(lambda x: x.set_color(hsv_to_hex((huevals2.hue_tracker.get_value()
↪%360, huevals2.sat_tracker.get_value(),1))))
        t3.add_updater(lambda x: x.set_color(hsv_to_hex((huevals3.hue_tracker.get_value()
↪%360, huevals3.sat_tracker.get_value(),1))))


        huevals1=HueValSlider(wheels,0,1,1)
        huevals2=HueValSlider(wheels,120,1,1)
        huevals3=HueValSlider(wheels,240,1,1)



        self.add(huevals1)
        self.add(huevals2)
        self.add(huevals3)
        hues_all_tracker = ValueTracker(0)

        self.add(hues_all_tracker)
        self.add(huevals1.hue_tracker)
        self.add(huevals2.hue_tracker)
        self.add(huevals3.hue_tracker)

        huevals1.hue_tracker.add_updater(lambda mobject, dt: mobject.increment_
↪value(dt*30))
        huevals2.hue_tracker.add_updater(lambda mobject, dt: mobject.increment_
↪value(dt*30))
        huevals3.hue_tracker.add_updater(lambda mobject, dt: mobject.increment_
↪value(dt*30))

        self.wait(3)
```

(continues on next page)

(continued from previous page)

```
        self.play(
            huevals1.sat_tracker.animate.increment_value(-0.2),
            huevals2.sat_tracker.animate.increment_value(-0.2),
            huevals3.sat_tracker.animate.increment_value(-0.2),
        )
        self.wait(1)
        self.play(
            huevals1.val_tracker.animate.increment_value(-0.2),
            huevals2.val_tracker.animate.increment_value(-0.2),
            huevals3.val_tracker.animate.increment_value(-0.2),
        )
        self.wait(1)
```

```
<IPython.core.display.Video object>
```

1.7 Scene Building With Plots

One can distingish between two kinds of plots:

- Quantitative scientific plots with numbers and dimensions to describe data (e.g. made with matplotlib).
- Qualitative explanatory plots that convey a message in the clearest way possible (e.g. made with manim)

In this tutorial, I will take you on a journey from choosing a topic to making a scientific plot to then transforming it into an explanatory plot with manim:

1.7.1 Formulating a Quantitative Concept

First, we do research on our topic of choice and then look up the formulas that we need. Alternatively, one can search for existing implementations.
I chose the carnot process where I want to see how the pressure **pressure p** is altering, when **volume V** or **temperature T** are changing.

In order to see how this works, the only think we need to know is that the Carnot Cycle obeys these formulas:

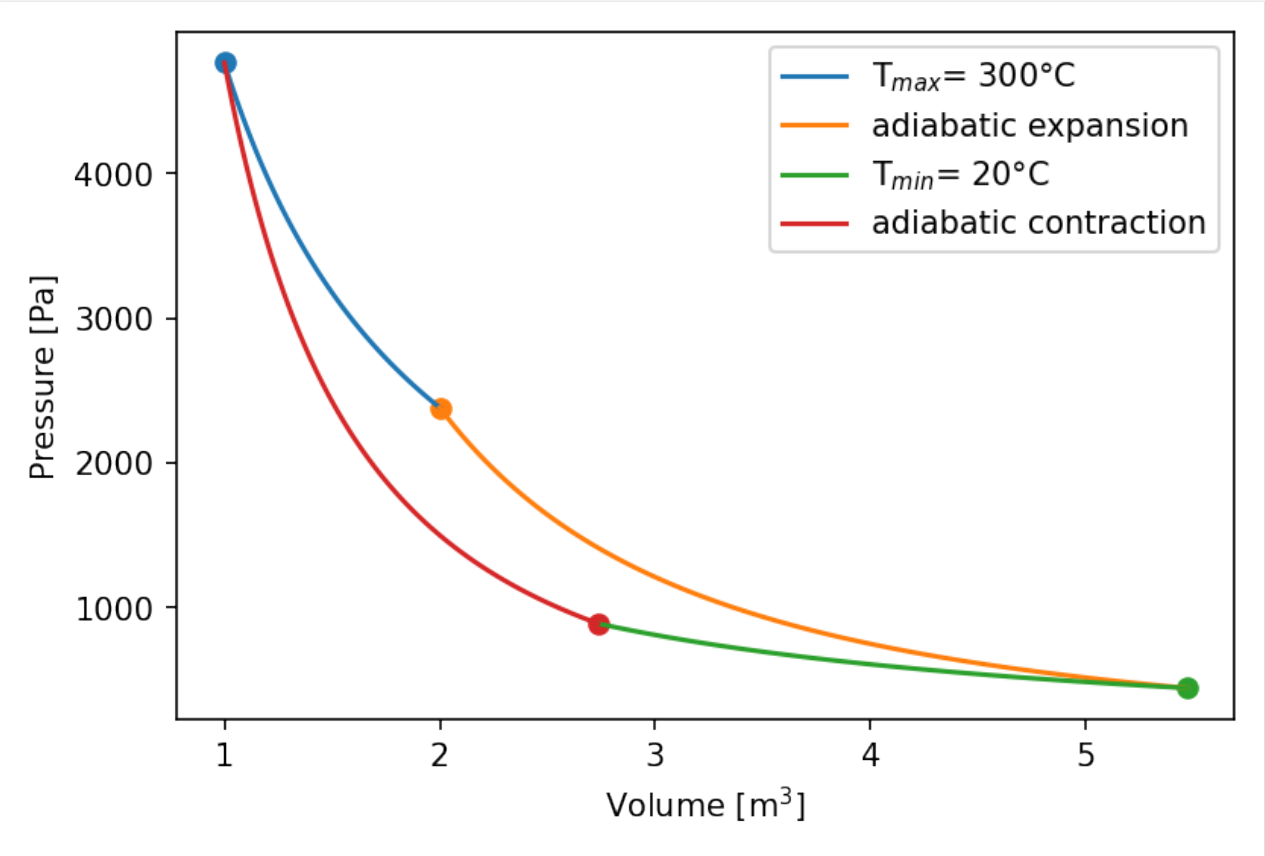
- $pV = RT $ ideal gas equation
- $pV = const $ upper and lower curve (also called “isotherm”)
- $pV^k = const $ with $ k = 5/3$ for the left and right curve (also called “adiabatic”)

You don’t have to understand these formulas in detail in order to understand this tutorial.
As we need reference points in the diagram, we first define some default values.
For temperatures, we choose $ 20 °$C and 300C. For volumes we choose $v_1 = 1\,\mathrm{m}^3$, and $v_2 = 2\,\mathrm{m}^3$.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.constants import zero_Celsius
plt.rcParams['figure.dpi'] = 150

Tmax = zero_Celsius +300
Tmin = zero_Celsius +20
R = 8.314
kappa = 5/3
V1= 1
V2= 2
```

Now, let's have a look on the plot via matplotlib. As of now, implementing and debugging formulas is important, design is not.

[2]:
```
p1 = R*Tmax/V1 # ideal gas equation
p2 = p1*V1/V2

V3 = (Tmax/Tmin * V2**(kappa-1))**(1/(kappa-1))
p3 = p2* V2**kappa / V3**kappa

V4 = (Tmax/Tmin * V1**(kappa-1))**(1/(kappa-1))
p4 = p3*V3/V4

V12 = np.linspace(V1,V2,100)
V23 = np.linspace(V2,V3,100)
V34 = np.linspace(V3,V4,100)
V41 = np.linspace(V4,V1,100)

def p_isotherm(V,T):
    return (R*T)/V

def p_adiabatisch(V,p_start,v_start):
    return (p_start*v_start**kappa)/V**kappa

plt.plot(V12, p_isotherm(V12,Tmax),label = "T$_{max}$" +f"= {Tmax-zero_Celsius:.0f}°C")
plt.plot(V23, p_adiabatisch(V23, p2,V2),label = f"adiabatic expansion")
plt.plot(V34, p_isotherm(V34,Tmin),label = "T$_{min}$" +f"= {Tmin-zero_Celsius:.0f}°C")
plt.plot(V41, p_adiabatisch(V41, p4,V4),label = f"adiabatic contraction")

plt.legend()
plt.scatter(V1,p1)
plt.scatter(V2,p2)
plt.scatter(V3,p3)
plt.scatter(V4,p4)

plt.ylabel("Pressure [Pa]")
plt.xlabel("Volume [m$^3$]")
plt.ticklabel_format(axis="x", style="sci", scilimits=(0,5))
```



Good! Now comes the second part:
Building the explanatory plot!

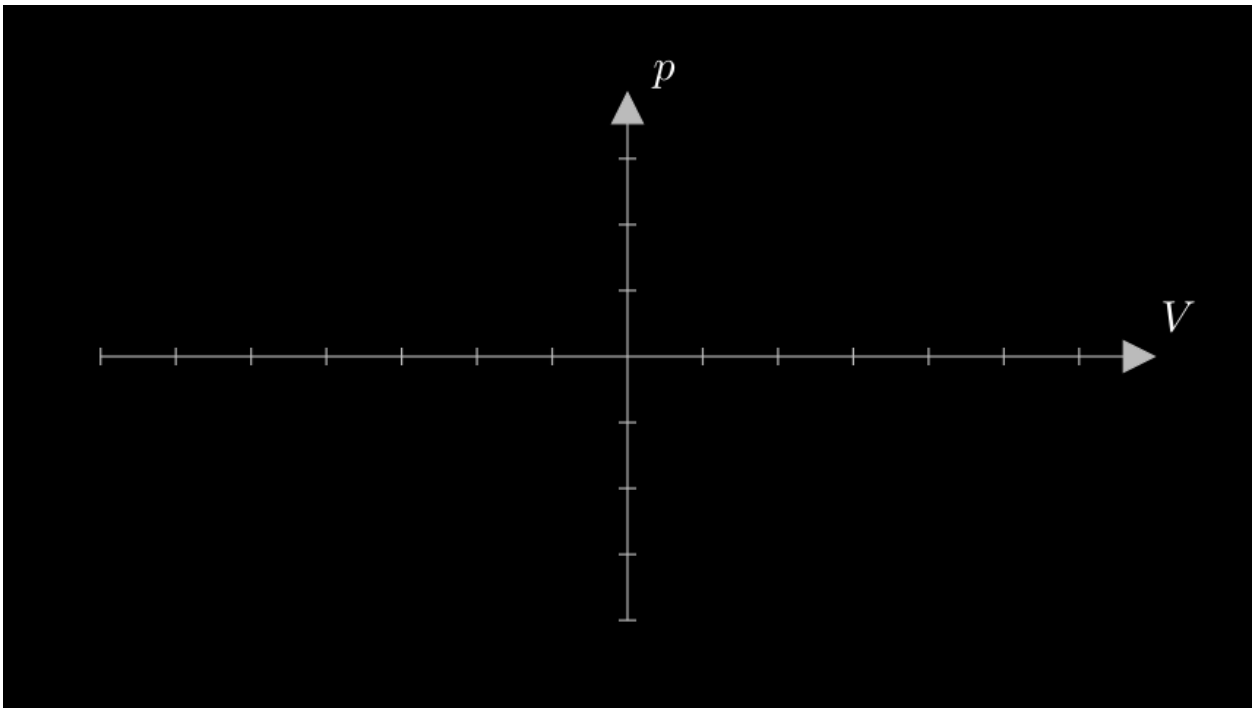
1.7.2 Extending to The Qualitiative Concept

Now we have a good basis to convert this idea into a visually appealing and explanatory graph that will make it easy for everyone to understand complex problems.

[3]:
```
from manim import *
param   = "-v WARNING -s -ql --disable_caching --progress_bar None Example"
```

```
Manim Community v0.11.0
```

[4]:
```
%%manim $param
class Example(Scene):
    def construct(self):
        my_ax = Axes()
        labels = my_ax.get_axis_labels(x_label="V", y_label="p")
        self.add(my_ax,labels)
```



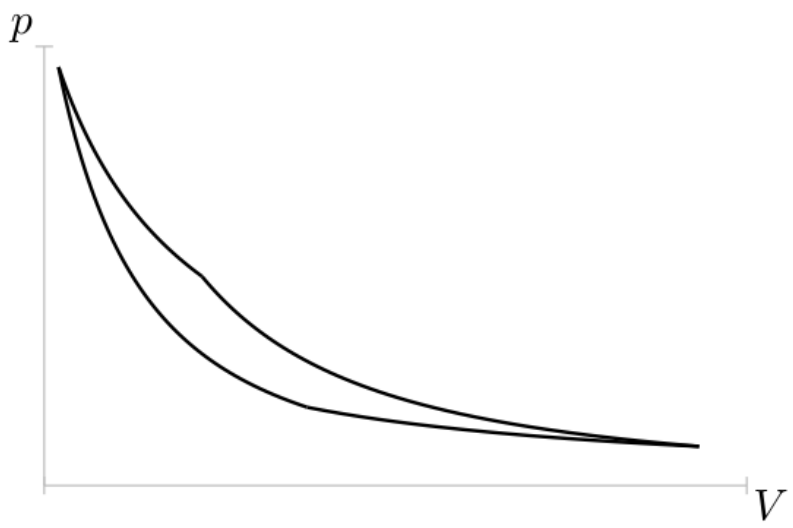
[5]:
```
# making some styling here
Axes.set_default(color=GREY, tips= False)
MathTex.set_default(color=BLACK)
config.background_color=WHITE
```

[6]:
```
%%manim $param
ax = Axes(x_range=[0.9, 5.8, 4.9], y_range=[0, 5000, 5000],x_length=8, y_length=5)
labels = ax.get_axis_labels(x_label="V", y_label="p")
labels[0].shift(.6*DOWN)
labels[1].shift(.6*LEFT)
isotherm12_graph = ax.get_graph(
    lambda x: p_isotherm(x, Tmax), x_range=[V1, V2,0.01], color=BLACK
)
adiabatisch23_graph = ax.get_graph(
    lambda x: p_adiabatisch(x, p2, V2) , x_range=[V2, V3,0.01], color=BLACK
)
isotherm34_graph = ax.get_graph(
    lambda x: p_isotherm(x, Tmin), x_range=[V3, V4,0.01], color=BLACK
)
adiabatisch41_graph = ax.get_graph(
    lambda x: p_adiabatisch(x, p4, V4), x_range=[V4, V1,0.01], color=BLACK
)
lines = VGroup(
    isotherm12_graph, adiabatisch23_graph, isotherm34_graph, adiabatisch41_graph
)

ax.add(labels)

class Example(Scene):
```

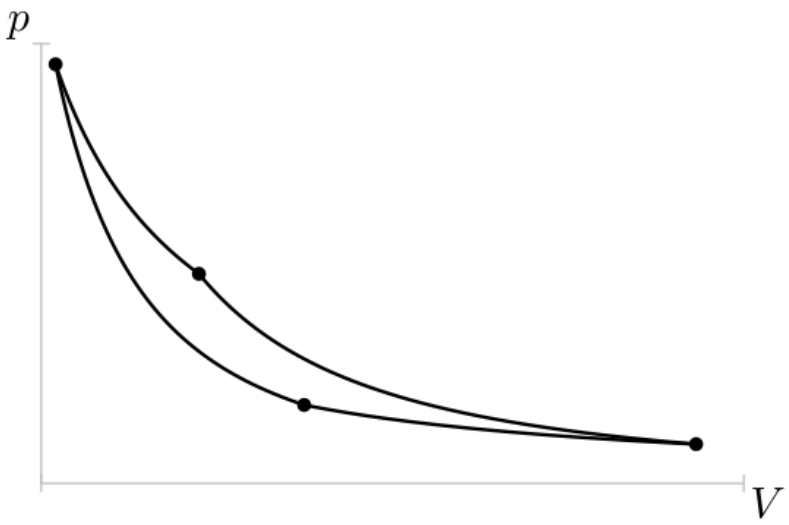
(continues on next page)

(continued from previous page)

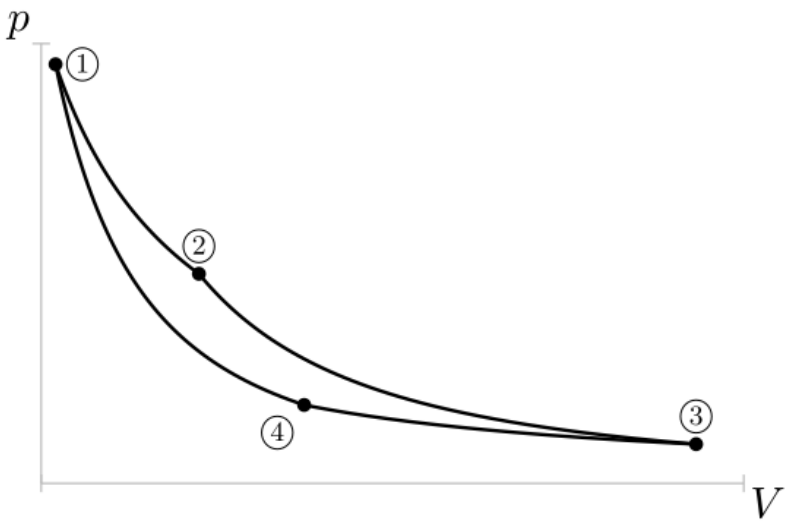
```
    def construct(self):
        self.add(ax,lines)
```



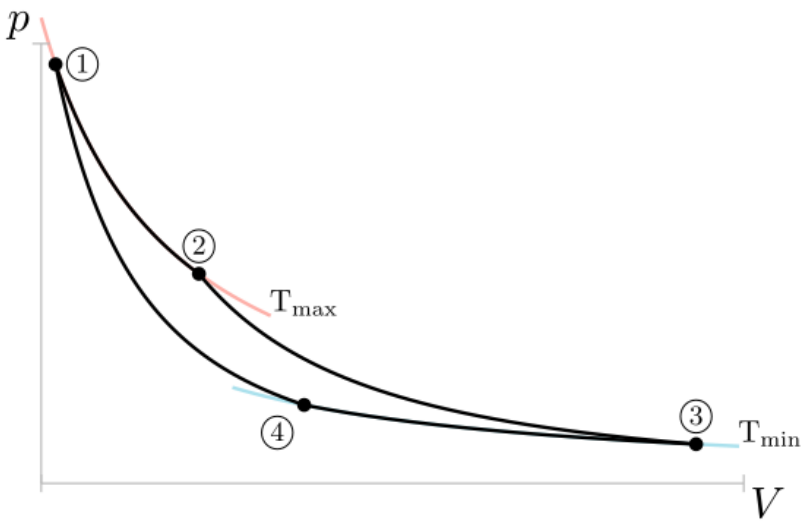
[7]:
```
%%manim $param

Dot.set_default(color=BLACK)
dots = VGroup()
dots += Dot().move_to(isotherm12_graph.get_start())
dots += Dot().move_to(isotherm12_graph.get_end())
dots += Dot().move_to(isotherm34_graph.get_end())
dots += Dot().move_to(isotherm34_graph.get_start())

class Example(Scene):
    def construct(self):
        self.add(ax,lines,dots)
```



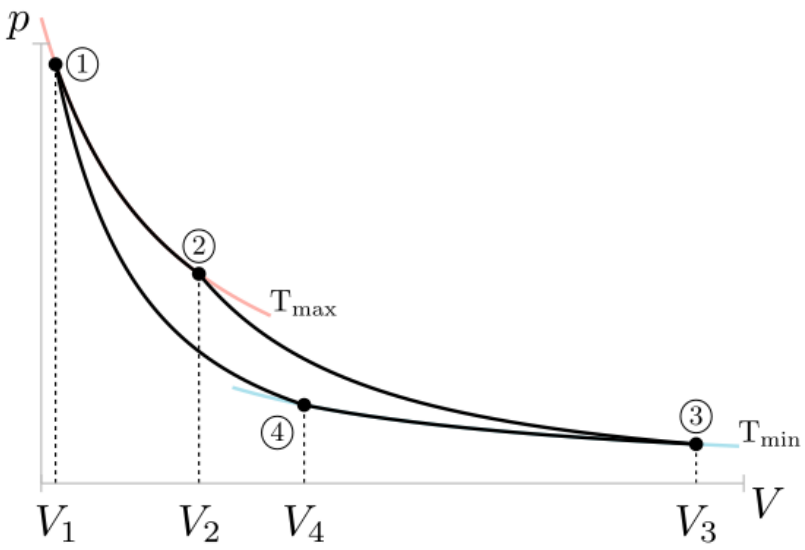
[8]:
```
%%manim $param

nums= VGroup()
nums+= MathTex(r"{\large \textcircled{\small 1}} ").scale(0.7).next_to(dots[0],RIGHT,
↪buff=0.4*SMALL_BUFF)
nums+= MathTex(r"{\large \textcircled{\small 2}} ").scale(0.7).next_to(dots[1],UP,␣
↪buff=0.4 * SMALL_BUFF)
nums+= MathTex(r"{\large \textcircled{\small 3}} ").scale(0.7).next_to(dots[2],UP,buff=0.
↪4*SMALL_BUFF)
nums+= MathTex(r"{\large \textcircled{\small 4}} ").scale(0.7).next_to(dots[3],DL ,
↪buff=0.4*SMALL_BUFF)

class Example(Scene):
    def construct(self):
        self.add(ax,lines, dots,nums)
```



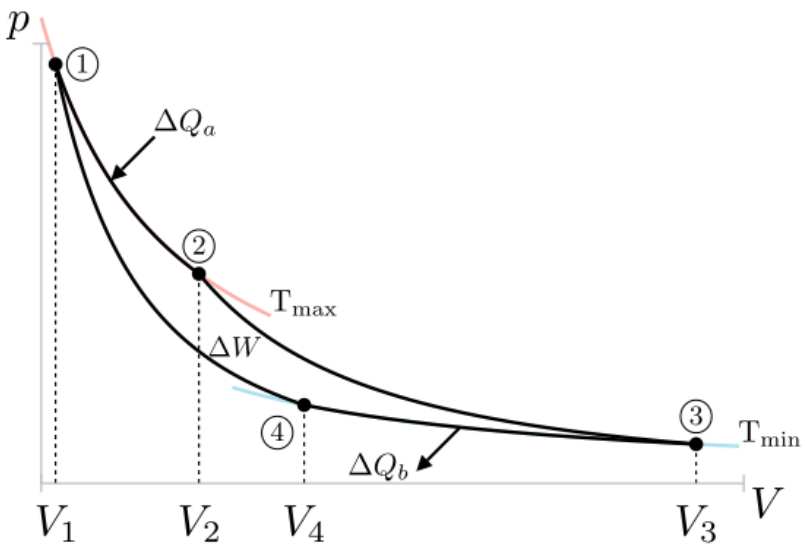
[9]:
```
%%manim $param

background_strokes = VGroup()
background_strokes += ax.get_graph(lambda x: p_isotherm(x, Tmax),x_range=[V1 - 0.1, V2 +␣
↪0.5, 0.01], color=RED, stroke_opacity=0.5)
background_strokes += ax.get_graph(lambda x: p_isotherm(x, Tmin), x_range=[V3 + 0.3,V4 -␣
↪0.5,0.01], color=BLUE, stroke_opacity=0.5)
background_strokes.set_z_index(-1);
label = VGroup()
label += MathTex(r"\text{T}_{\text{min}}").scale(0.7).next_to(background_strokes[1],
↪RIGHT,aligned_edge=DOWN, buff=0)
label += MathTex(r"\text{T}_{\text{max}}").scale(0.7).next_to(background_strokes[0],
↪RIGHT,aligned_edge=DOWN, buff=0)
background_strokes += label

class Example(Scene):
    def construct(self):
        self.add(ax,lines, dots,nums,background_strokes)
```



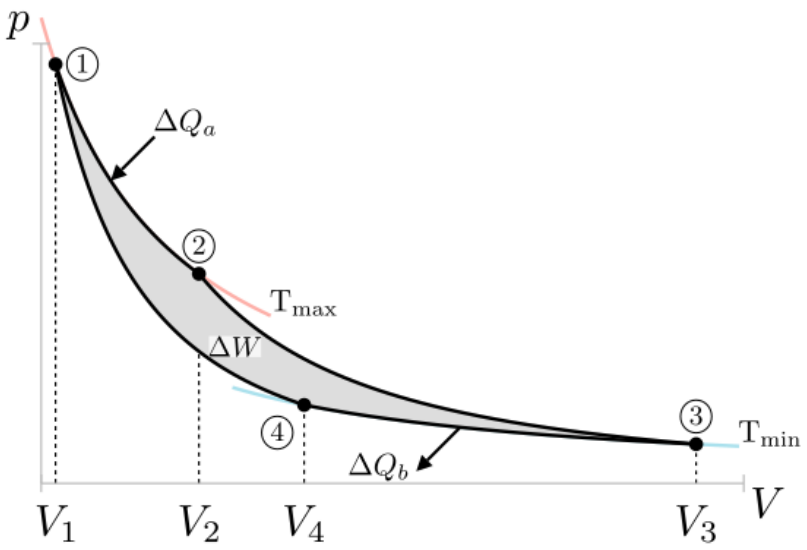
[10]:
```
%%manim $param

downstrokes = VGroup()
downstrokes += ax.get_vertical_line(ax.i2gp(V1, isotherm12_graph), color=BLACK).set_z_
↪index(-2)
downstrokes += ax.get_vertical_line(ax.i2gp(V2, isotherm12_graph), color=BLACK).set_z_
↪index(-2)
downstrokes += ax.get_vertical_line(ax.i2gp(V3, isotherm34_graph), color=BLACK).set_z_
↪index(-2)
downstrokes += ax.get_vertical_line(ax.i2gp(V4, isotherm34_graph), color=BLACK).set_z_
↪index(-2)
down_labels= VGroup()
down_labels += MathTex("{ V }_{ 1 }").next_to(downstrokes[0], DOWN)
down_labels += MathTex("{ V }_{ 2 }").next_to(downstrokes[1], DOWN)
down_labels += MathTex("{ V }_{ 3 }").next_to(downstrokes[2], DOWN)
down_labels += MathTex("{ V }_{ 4 }").next_to(downstrokes[3], DOWN)

class Example(Scene):
    def construct(self):
        self.add(ax,lines, dots,nums,background_strokes, downstrokes,down_labels)
```



[11]:
```
%%manim $param

heat_annotation = VGroup()
deltaW = MathTex(r"\Delta W").next_to(dots[3], UL).scale(0.65).shift(0.15 * UP)
bg = deltaW.add_background_rectangle(color=WHITE)
heat_annotation += deltaW

point = isotherm12_graph.point_from_proportion(0.5)
arrow = Arrow(point + UR * 0.5, point, buff=0).set_color(BLACK)
deltaQa = MathTex(r"\Delta Q_a").scale(0.7).next_to(arrow, UR, buff=0)
heat_annotation += arrow
heat_annotation += deltaQa

point = isotherm34_graph.point_from_proportion(0.4)
arrow = Arrow(point, point + DL * 0.5, buff=0).set_color(BLACK)
deltaQb = MathTex(r"\Delta Q_b").scale(0.7).next_to(arrow, LEFT, buff=0.1).shift(0.2 *␣
↪DOWN)
heat_annotation += arrow
heat_annotation += deltaQb

class Example(Scene):
    def construct(self):
        self.add(ax,lines, dots,nums,background_strokes, downstrokes,down_labels,heat_
↪annotation)
```



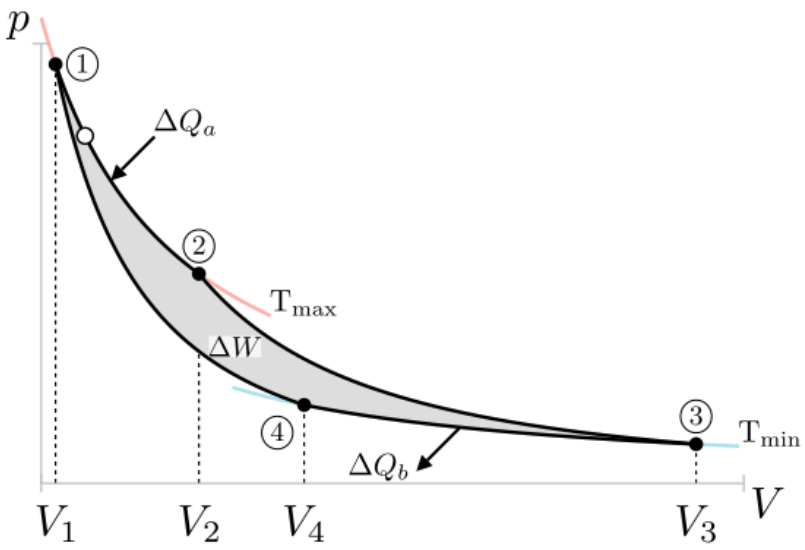
[12]:
```
%%manim $param

c1 = Cutout(lines[0].copy().reverse_points(),lines[3]).set_opacity(1).set_color(GREEN)
c2 = Cutout(lines[1],lines[2])
bg_grey = Union(c1,c2, color=GREY_A).set_opacity(1)
bg_grey.z_index=-1

class Example(Scene):
    def construct(self):
        #self.add(c1,c2)
        self.add(ax,lines, dots,nums,background_strokes)
        self.add(downstrokes,down_labels,heat_annotation,bg_grey)
```



[13]:
```
carnot_graph= VGroup(ax,lines, dots,nums,background_strokes,downstrokes,down_labels,heat_
↪annotation,bg_grey)
```

And here is the final plot:

[14]:
```
%%manim $param

sourunding_dot = Dot().scale(1.3).set_fill(color=BLACK).set_z_index(-1)
innerdot = Dot().set_color(WHITE).scale(1)
moving_dot = VGroup(sourunding_dot, innerdot)
moving_dot.move_to(isotherm12_graph.point_from_proportion(0.3))

class Example(Scene):
    def construct(self):
        self.add(carnot_graph)
        self.add(moving_dot)
```

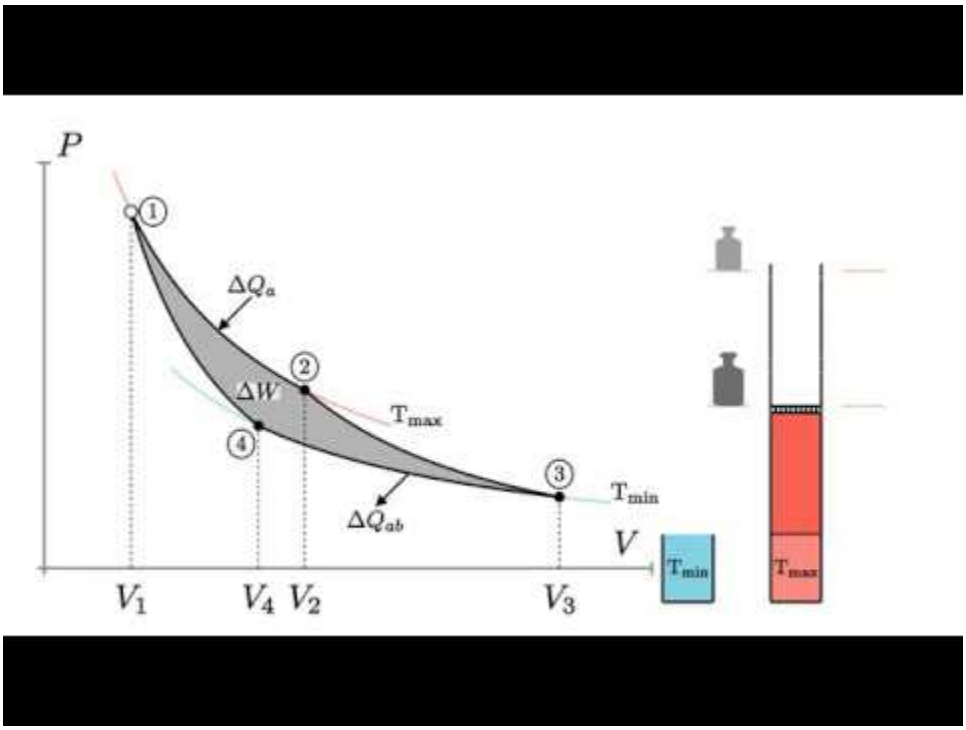


1.7.3 Outlook

After having this foundation of an explanory plot, one can go on and animtate it as can be seen here.
(A tutorial for this animation will follow!)

[15]:
```
from IPython.display import YouTubeVideo
YouTubeVideo('_8RkZaiXP0E', width=800, height=600)
```

[15]:

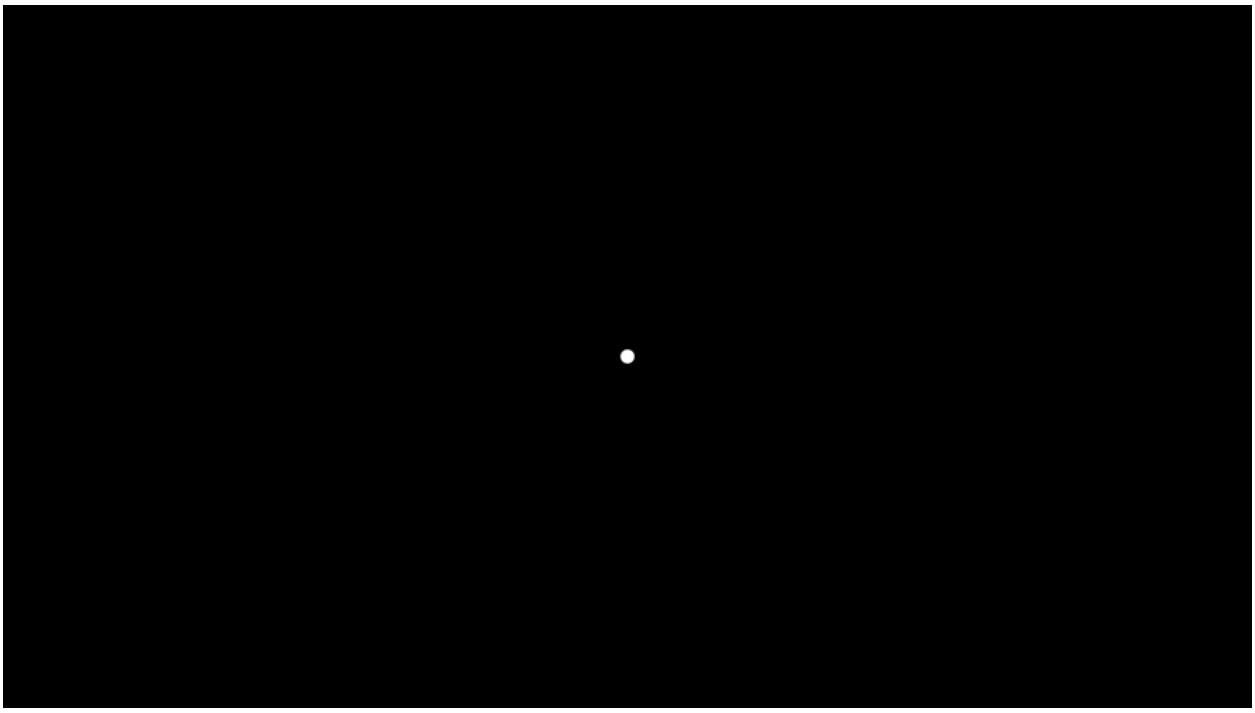


1.8 (TL;DR CheatSheet)

[1]:
```
from manim import *

param   = "-v WARNING -s -ql --disable_caching --progress_bar None Example"
paramH  = "-v WARNING -s -qh --disable_caching --progress_bar None Example"
paramp  = "-v WARNING    -ql --disable_caching --progress_bar None Example"
parampH = "-v WARNING    -qh --disable_caching --progress_bar None Example"
```

```
Manim Community v0.11.0
```

[2]:
```
%%manim $param
class Example(Scene):
    def construct(self):
        self.add(Dot())
```



[3]:
```
!manim render --help
```

```
Manim Community v0.11.0

Usage: manim render [OPTIONS] FILE [SCENE_NAMES]...

  Render SCENE(S) from the input FILE.

  FILE is the file path of the script.

  SCENES is an optional list of scenes in the file.

Global options:
  -c, --config_file TEXT          Specify the configuration file to use for
                                  render settings.
  --custom_folders                Use the folders defined in the
                                  [custom_folders] section of the config file
                                  to define the output folder structure.
  --disable_caching               Disable the use of the cache (still
                                  generates cache files).
  --flush_cache                   Remove cached partial movie files.
  --tex_template TEXT             Specify a custom TeX template file.
  -v, --verbosity [DEBUG|INFO|WARNING|ERROR|CRITICAL]
                                  Verbosity of CLI output. Changes ffmpeg log
                                  level unless 5+.
  --notify_outdated_version / --silent
                                  Display warnings for outdated installation.
  --enable_gui                    Enable GUI interaction.
  --gui_location TEXT             Starting location for the GUI.
  --fullscreen                    Expand the window to its maximum possible
```

(continues on next page)

(continued from previous page)

```
                                size.
  --enable_wireframe            Enable wireframe debugging mode in opengl.
  --force_window                Force window to open when using the opengl
                                renderer, intended for debugging as it may
                                impact performance

Output options:
  -o, --output_file TEXT        Specify the filename(s) of the rendered
                                scene(s).
  -0, --zero_pad INTEGER RANGE  Zero padding for PNG file names.  [0<=x<=9]
  --write_to_movie              Write to a file.
  --media_dir PATH              Path to store rendered videos and latex.
  --log_dir PATH                Path to store render logs.
  --log_to_file                 Log terminal output to file.

Render Options:
  -n, --from_animation_number TEXT
                                Start rendering from n_0 until n_1. If n_1
                                is left unspecified, renders all scenes
                                after n_0.
  -a, --write_all               Render all scenes in the input file.
  --format [png|gif|mp4|webm|mov]
  -s, --save_last_frame
  -q, --quality [l|m|h|p|k]     Render quality at the follow resolution
                                framerates, respectively: 854x480 30FPS,
                                1280x720 30FPS, 1920x1080 60FPS, 2560x1440
                                60FPS, 3840x2160 60FPS
  -r, --resolution TEXT         Resolution in (W,H) for when 16:9 aspect
                                ratio isn't possible.
  --fps, --frame_rate FLOAT     Render at this frame rate.
  --renderer [cairo|opengl|webgl]
                                Select a renderer for your Scene.
  --use_opengl_renderer         Render scenes using OpenGL (Deprecated).
  --use_webgl_renderer          Render scenes using the WebGL frontend
                                (Deprecated).
  --webgl_renderer_path PATH    The path to the WebGL frontend.
  -g, --save_pngs               Save each frame as png (Deprecated).
  -i, --save_as_gif             Save as a gif (Deprecated).
  -s, --save_last_frame         Save last frame as png (Deprecated).
  -t, --transparent             Render scenes with alpha channel.
  --use_projection_fill_shaders Use shaders for OpenGLVMobject fill which
                                are compatible with transformation matrices.
  --use_projection_stroke_shaders
                                Use shaders for OpenGLVMobject stroke which
                                are compatible with transformation matrices.

Ease of access options:
  --progress_bar [display|leave|none]
                                Display progress bars and/or keep them
                                displayed.
  -p, --preview                 Preview the Scene's animation. OpenGL does a
                                live preview in a popup window. Cairo opens
```

(continues on next page)

(continued from previous page)

```
                                the rendered video file in the system
                                default media player.
  -f, --show_in_file_browser    Show the output file in the file browser.
  --jupyter                     Using jupyter notebook magic.

Other options:
  --help                        Show this message and exit.

  Made with <3 by Manim Community developers.
```

1.9 Changelog

1.9.1 0.9.0

- added copybutton
- updated manim version
- Shortened headings

1.9.2 0.10.0

- changed plugin to manim_physics
- Added copy-paste gallery in chapter 2 as a link to https://kolibril13.github.io/mobject-gallery/

1.9.3 0.11.0

- Removing “Last Edited”
- Update Version
- added new Banner
- Removing numbers in chaptername
- adding chapter “Scene Building with plots”
- removing chapter “additional tools”
- Reordering chapters

Repository on GitHub: https://github.com/kolibril13/flyingframes