Manim

Jianfeng He

目录

第一章　Quick Start

§1.1　Scene

1.1.1　Introduction

texttt {Scene} 是最基础和最主要的类, 几乎其它所有的类都由其构造而来

1.1.2　Method

1). Scene.plsy(): 播放动画, 如:

```
.play(ShowCreation(circle))
.play(FadeOut(circle))
.play(GrowFromCenter(square))
.play(Transform(square,triangle))
```

2). .add(): places a mobject on screen at the start of the scene

3). arrow.next_to(): 使用相对位置进行移动, 如:

```
arrow = Arrow(LEFT,UP)
arrow.next_to(circle,DOWN+LEFT)
```

4). .move_to(): move the object to a specific location on the screen, (使用绝对位置?)

5). .wait(<number>): 控制停止的秒数, 默认为 1 秒

§1.2　Mobject

1.2.1　Brace()

为物体加大括号, 可以利用 LEFT 等方向操作来控制添加的大括号的位置, 常常与组群一起使用. 如:

```
eq_group=VGroup(eq1A,eq2A)
braces=Brace(eq_group,LEFT)
eq_text = braces.get_text("A pair of equations")
```

1. .get_text()

为大括号加上文字, 似乎返回的也是一个对象, 可以进行 TextMobject 允许的各种操作, 如 Write() 等. e.g.:

```
self.play(GrowFromCenter(braces),Write(eq_text))
```

Notation:

1). This method does not draw the text on the screen, it is only used to set the location of the text relative to the braces so you will still need to add the text to the screen. test

2). 教程中关于加括号的说明的第二个例子中, 出现了诸如数组名前面加上一个星号的语句: `eq1_mob=TexMobject(*eq1_text)`, 作者对其的解释为: Notice that when creating the texmobjects that we passed the variable name of the list with an asterisk in front of it `eq1_mob=TexMobject(*eq1_text)`. The asterisk is a Python command to unpack the list and treat the argument as a comma-separated list. Thus `eq1_mob=TexMobject(*eq1_text)` is identical to:

```
eq1_mob=TexMobject("4","x","+","3","y","=","0")
```

1.2.2 TextMobject

Text rendering is based on Latex so you can use many Latex typesetting features, which means if you already known the rule of LaTeX, it will be easily to use equation. One should be noted that the backslash should be tranfered by another baskslash. A more convenient way is use Python's raw string (add `r` before the first quotation mark) e.g., those are two valid using methods:

```
eq1=TextMobject("$\\vec{X}_0 \\cdot \\vec{Y}_1 = 3$")
eq1=TextMobject(r"$\vec{X}_0 \cdot \vec{Y}_1 = 3$")
```

When mobjects of any sort are created the default position seems to be the center of the screen. Once created you can use `shift()` or `move_to()` to change the location of the mobjects.

1. `.scale(<number>)`

缩放文字

2. `.set_color()`

设置文字的颜色. 颜色可以采用默认提供的一些常数. 具体的定义参见 `constants.py`.

3. `.set_color_by_gradient(<color_1>, <color_2>, <color_3>)`

括号内的参数为一串以逗号分隔的颜色序列, 效果是生成按照该颜色序列进行的渐变色.

4. `.match_color(<obj>)`

In addition to setting the color, you can also match the color to another object. This is convenient since you can easily keep the color of two different objects as same without define extral variable.

5. `.to_edge()`, `.to_corner()`

You can align mobjects with the center of the edge of the screen by telling `to_edge()` whether you want the object to be `UP`, `DOWN`, `LEFT`, or `RIGHT`. You can also use `to_corner()`, in which case you need to combine two directions such as `UP+LEFT` to indicate the corner.

6. `.get_corner()`

Each mobject has a bounding box that indicates the outermost edges of the mobject and you can get the coordinates of the corners of this bounding box using `get_corner()` and specifying a direction. Thus `get_corner(DOWN+LEFT)` will return the location of the lower left corner of a mobject.

7. `.shift()`

变换物体的位置, 为达成缓慢上升的效果:

```
self.play(ApplyMethod(my_first_text.shift,3*UP))
```

注意到其中 ApplyMethod() 的重要性. 若没有该函数, 则上升瞬间完成. Notice the arguments of ApplyMethod() is a pointer to the method (in this case my_first_text.shift without any parentheses) followed by a comma and then the what you would normally include as the argument to the shift() method. In other words, ApplyMethod(my_first_text.shift,3*UP) will create an animation of shifting my_first_text three MUnits up.

8. .rotate()

旋转文字. 其中可调用常量 TAU, 一个 TAU 相当于 2π.

9. Write()

显示文字, 一个例子是:

```
eq1=TextMobject("$\\vec{X}_0 \\cdot \\vec{Y}_1 = 3$")
self.play(Write(eq1))
```

You can also pass a string to Write() and it will create the TextMobject for you. Write() needs to be inside of play() in order to animate it.

10. align_to()

用以进行对齐操作. 如:

```
eq1A = TextMobject("4x + 3y")
eq1B = TextMobject("=")
eq1C = TextMobject("0")
eq2A = TextMobject("5x -2y")
eq2B = TextMobject("=")
eq2C = TextMobject("3")
eq1B.next_to(eq1A,RIGHT)
eq1C.next_to(eq1B,RIGHT)
eq2A.shift(DOWN)
eq2B.shift(DOWN)
eq2C.shift(DOWN)
eq2A.align_to(eq1A,LEFT)
eq2B.align_to(eq1B,LEFT)
eq2C.align_to(eq1C,LEFT)
```

这个粒子中, 现将 eq1 进行排列, 再将 eq2 的每一个元素下移后, 与 eq1 中的元素进行对齐操作

1.2.3 TexMobject

Similar to TextMobject, the only different is you can ignore the enclosed $ symbol, which made it more convenient to type LaTeXequation. There is a good example:

```
eq2=TexMobject(r"\vec{F}_{net} = \sum_i \vec{F}_i")
```

1. set_color_by_tex()

The rule can be understanded by an example:

```
line1=TexMobject(r"\text{The vector } \vec{F}_{net} \text{ is the net }",r"\text{
    force }",r"\text{on object of mass }")
```

For instance, if we type in `line1.set_color_by_tex("F",BLUE)`, the only place a capital `F` occurs is in the force variable so the first part of this line is blue. If instead we try `line1.set_color_by_tex("e",BLUE)`, the letter e appears in several places in line1 so the entire line ends up blue.

Notation:

1). 感觉这个函数有点像是从匹配节点开始进行染色. 如果有多个匹配位点, 那么就应该会有多段被染色的地方. 具体的还需进一步实验及查看源码.

2. `set_color_by_tex_to_color_map()`

If you want to change the color of multiple elements within a list of texmobjects you can use `set_color_by_tex_to_color_map()` and a dictionary. The key for the dictionary should be the text we want colored (or a unique part of the string) and the value should be the desired color. e.g.:

```
line2=TexMobject("m", "\\text{ and acceleration }", "\\vec{a}", ". ")
line2.set_color_by_tex_to_color_map({
    "m": YELLOW,
    "{a}": RED
})
```

§1.3 Constant

1.3.1 长度常数

`UP`, `DOWN`, `LEFT`, `RIGHT`. 这些长度常数可以被当成向量来进行计算, 也就是说可以直接进行加减. (实际上它们本来也是通过 `numpy` 的数组进行定义的)

Notation :

1). The screen height is set to a default of 8 MUnits, which means a 2 MUnit shift corresponds to about a quarter of the screen height.

§1.4 VGroup

The `VGroup` class allows you to combine multiple mobjects into a single vectorized math object. The using of the class is like:

```
label2_group=VGroup(label2,label2.bg)
```

This statement combine the object `label2` and `label2.bg` then generate a new object.

1. `arrange_submobjects()`

By grouping the two lines together with `VGroup()`, we can use the `arrange_submobjects()` method to space out the two lines. The first argument is the direction you want the objects spaced out and buff is the buffer distance between the mobjects. There are several default buffer distances defined in constants.py but you can also a single number. The smallest default buffer is `SMALL_BUFF=0.1` and the largest is `LARGE_BUFF=1`. e.g.:

```
sentence=VGroup(line1,line2)
sentence.arrange_submobjects(DOWN, buff=MED_LARGE_BUFF)
```