Manim

Python and ffmeg required

Download community version

We write animations in classes, every class has a construct function which contains the animation, you can think class as canvas and construct method has the animation

from manim import \*

class example(Scene):

    def construct(self):

        blue\_circle = Circle(color=BLUE, fill\_opacity=0.5) //just created not

        green\_square = Square(color=GREEN, fill\_opacity=0.5)

        green\_square.next\_to(blue\_circle,RIGHT) //positioning

        self.add(blue\_circle,green\_square) //this add them to screen

to run it in terminal type **manim –qm –p filename.py classname**

qm is medium quality and –p is the preview flag

* scene: animation canvas
* mobjects: circle square manim objects
* mobject have properties color fill\_opacity
* global constants colors (BLUE,GREEN…) directions (UP, DOWN, LEFT…)
* docs.manim.community has the reference manual for all constants and mobjects

Plotting a Function:

from manim import \*

def g1(x):

    return (x\*\*3)/10

class example(Scene):

    def construct(self):

        ax = Axes(x\_range=(-10,10), y\_range=(-10,10))

        curve = ax.plot(g1, color=BLUE)

        area = ax.get\_area(curve, x\_range=[-5,5])

        self.add(ax,curve,area)

we can also use lamda functions

animations:

scence.add adds mobjects immediately

scene.wait pauses then adds the mobjects

scene.play plays animation

animations can

* add mobjects (create, fadein)
* change mobjects (transform)
* emphasize mobjects (indicate, circumscribe)
* remove mobjects (uncreate, fadeout)
* def construct(self):
* ax = Axes(x\_range=(-10,10), y\_range=(-10,10))
* curve = ax.plot(g1, color=BLUE)
* area = ax.get\_area(curve, x\_range=[-5,5])
* # self.play(Create(ax)) #first play this
* # self.play(Create(curve)) #then play this
* self.play(Create(ax, run\_time=5), Create(curve,run\_time=2)) #play at same time
* self.play(FadeIn(area), run\_time=2)
* self.wait(1) #wait before animating

another example:

    def construct(self):

        blue\_square = Square(color=BLUE, fill\_opacity=0.5)

        circle2 = Circle(color=RED, fill\_opacity=0.5)

        self.play(DrawBorderThenFill(blue\_square))

        self.play(Transform(blue\_square, circle2))

        # self.play(FadeOut(circle2)) #it doest fade out

        # this happens because after transform the blue\_square becomes circle2 or the first argument becomes the second argument, so we have to fade out blue\_square or the first argument

        self.play(Indicate(blue\_square)) #emphasize

        self.play(FadeOut(blue\_square))

        #or we can use ReplacementTransform so that the first argument doesnt become the second argument