fractals.py

```
class HuchinsonContractionModel(MovingCameraScene):
    def construct (self):

        numLine = NumberLine(include_ticks= True, font_size= 3, line_to_number_buff=0.05, tick_size= 0.005, stroke_width= 0.1)
        endPts = [0, 1]
        #numLine.add_labels(endPts)
        #numLine.scale(10)
        #numLine.center()
        self.add(numLine)

        ll = Line(start = [0,0,0], end = [1,0,0], color = GREEN, stroke_width = 1)
        B = VGroup(ll)

        #self.add(Dot(point = [0,0,0], color = PINK))

        numMaps = 2
        scales = [1/3, 1/2]
        shifts = [0, 1/2]




        maxIter = 5
        fac = 10

        self.camera.frame.shift(RIGHT*1/2).scale(1/fac)


        #w1.to_edge(UP/fac+LEFT/fac, buff= 1)
        #w2.next_to(w1, DOWN/fac, buff = 1)

        colors = [GREEN, BLUE]

        def format_as_fraction(value):
            frac = Fraction(value).limit_denominator()
            if frac.denominator == 1:
                return str(frac.numerator)  # Return as integer
            return f"\\frac{{{frac.numerator}}}{{{frac.denominator}}}"

        w1 = Tex(f"$W_{1}(x) = {format_as_fraction(scales[0])} x + {format_as_fraction(shifts[0])}VSCODE_PRINT_CONTENTquot;)
        w2 = Tex(f"$W_{2}(x) = {format_as_fraction(scales[1])} x + {format_as_fraction(shifts[1])}VSCODE_PRINT_CONTENTquot;)

        w1.shift(UP*3/fac)
        self.add(w1.scale(1/fac/2))
        self.add(w2.scale(1/fac/2).next_to(w1, DOWN/fac, buff = 1/fac))

```

```
        self.add(ll)
        self.wait()
        self.remove(ll)

        for i in range(maxIter):
            BNew = VGroup()
            labs = VGroup()
            for k in range(len(B)):
                mob = B[k]
                for j in range(numMaps):

                    l = Line(start = mob.get_start()*scales[j], end = mob.get_end()*scales[j], stroke_width = 1, color = colors[j])
                    l.shift(shifts[j]*RIGHT)

                    s = l.get_start()
                    e = l.get_end()
                    start_value = numLine.p2n(s)
                    end_value = numLine.p2n(e)
                    start_label = MathTex(format_as_fraction(start_value))
                    end_label = MathTex(format_as_fraction(end_value))
                    font_size = 0.025  # Scale factor for font size
                    start_label.scale(font_size)
                    end_label.scale(font_size)

                    endPts.append(start_value)
                    endPts.append(end_value)
                    #print(start_value, end_value)

                    start_label.next_to(numLine.n2p(start_value), 0.1*DOWN)
                    end_label.next_to(numLine.n2p(end_value), 0.1*DOWN)

                    labs.add(start_label, end_label)

                    BNew.add(l)

            self.remove(B)
            B = BNew
            self.add(BNew)
            self.add(labs)
            self.wait()
```