Índice

Lección 1 del curso - Vídeo 2

Marcos Bujosa

8 de mayo de 2024

0.1. Definición una subclase con algunas definiciones generales

```
class MiEscenaConVoz(VoiceoverScene):
    def pausa_muy_corta(self, n=0.3):
        self.wait(n)

    def pausa_corta(self, n=0.5):
        self.wait(n)

    def pausa(self, n=1):
        self.wait(n)

    def pausa_media(self, n=1.5):
        self.wait(n)

    def pausa_larga(self, n=3):
        self.wait(n)

    def pausa_muy_larga(self, n=5):
        self.wait(n)

    <<Créditos en distintas partes de la pantalla>>
```

```
def creditos(self, variante=1):
    def analisis_opcion_elegida(tipo):
        'Análisis de las opciones de eliminación elegidas'
        lista = [100,20,10,4,2,1]
        opcion = set()
        for t in lista:
            if (tipo - (tipo % t)) in lista:
                opcion.add(tipo - (tipo % t))
                tipo = tipo % t
        return opcion

    copyright = Tex(r"\textcopyright{\;} 2024\; Marcos Bujosa  ")
    if 1 in analisis_opcion_elegida(variante):
        stampDcha   = VGroup(copyright.copy()).rotate( PI/2).scale(0.5).to_edge(RIGHT, buff=0.1).set_color(GRAY_D)
        self.add(stampDcha)
    if 2 in analisis_opcion_elegida(variante):
        stampIzda   = VGroup(copyright.copy()).rotate(-PI/2).scale(0.5).to_edge(LEFT, buff=0.1).set_color(GRAY_D)
        self.add(stampIzda)
    if 4 in analisis_opcion_elegida(variante):
        stampBottom= VGroup(copyright.copy()).rotate(    0).scale(0.5).to_edge(DOWN, buff=0.1).set_color(GRAY_D)
        self.add(stampBottom)
    if 10 in analisis_opcion_elegida(variante):
        stampTop    = VGroup(copyright.copy()).rotate(    0).scale(0.5).to_edge(  UP, buff=0.1).set_color(GRAY_D)
        self.add(stampTop)
```

```
class VectorR2():
    def __init__(self, lista, rpr='c', color=GRAY_B):
        """Inicializa un Vector con una lista"""
        coords = lista + [0] if len(lista)<3 else lista
        self.color  = color
        self.coords = tuple(coords)
        self.Vector = nc.Vector(self.coords[:2], rpr)
        self.tex    = MathTex(self.Vector.latex(), color=self.color).scale(0.8)

    def dot(self, ejes, radio=0.08):
        return Dot(ejes.c2p(*self.coords), radius=radio, color=self.color)

    def v_line(self, ejes):
        return ejes.get_vertical_line(ejes.c2p(*self.coords), color=self.color)

    def h_line(self, ejes):
        return ejes.get_horizontal_line(ejes.c2p(*self.coords), color=self.color)

    def arrow(self, ejes):
        return ejes.get_vector(self.Vector.lista, stroke_color = self.color, stroke_width=4)
```

```
class VectorR3():
    def __init__(self, lista, rpr='c', color=GRAY_B):
        """Inicializa un Vector con una lista"""
        coords = lista + [0] if len(lista)<3 else lista
        self.color  = color
        self.coords = tuple(coords)
        self.Vector = nc.Vector(self.coords, rpr)
        self.tex    = MathTex(self.Vector.latex(), color=self.color).scale(0.8)

    def dot(self, ejes, radio=0.08):
        return Dot3D(ejes.c2p(*self.coords), radius=radio, color=self.color)

    def v_line(self, ejes):
        return ejes.get_vertical_line(ejes.c2p(*self.coords), color=self.color)

    def h_line(self, ejes):
        return ejes.get_horizontal_line(ejes.c2p(*self.coords), color=self.color)

    def arrow(self, ejes):
        return Arrow3D(
            start=np.array([0, 0, 0]),
            end=np.array(ejes.c2p(*self.coords)),
            resolution=8,
            color = self.color )
```

1. Suma de Vectores de $\mathbb{R}^n$ - V02

1.1. Español

1.1.1. Escena 1 - Suma de vectores de $\mathbb{R}^n$

```
<<Carga de la librería Manim y NacAL con AzureService>>
<<Definición de mi escena con voz>>
<<Definición de VectorR2>>
<<Definición de VectorR3>>

import itertools
def get_sub_indexes(tex):
    ni = VGroup()
    colors = itertools.cycle([RED,TEAL,GREEN,BLUE,PURPLE])
    for i in range(len(tex)):
        n = Text(f"{i}",color=next(colors)).scale(0.7)
        n.next_to(tex[i],DOWN,buff=0.01)
        ni.add(n)
    return ni

class L01_V02_E01_SumaDeVectores(MiEscenaConVoz):
    def construct(self):
        self.set_speech_service( AzureService(voice="es-ES-AlvaroNeural" ) )
        #self.set_speech_service(GTTSService(lang="es", tld="com"))

        myTemplate = TexTemplate()
        myTemplate.add_to_preamble(r"""\usepackage{nacal} """)

        self.creditos()

        # Portada
        titulo = Title(r"Suma de vectores de \R[n]",
                     tex_template = myTemplate,
                     font_size=70).set_color(BLUE)
        self.play(Write(titulo))
        self.pausa_media()
        self.play(FadeOut(titulo))

        # Definición de vector suma
        operacionSuma = Tex(r"Suma de vectores de \R[n]",
                         tex_template = myTemplate, font_size=70
                         ).to_edge(UP).set_color(BLUE)

        operacionDescripcion = Tex("La suma se define componente a componente.",
                         tex_template = myTemplate,
                         ).move_to([0,2.5,0]).to_edge(LEFT)
        # Ejemplos
        EjR3 = Tex(r"\textbf{Ejemplo en \R[3]:}",
                 tex_template = myTemplate,
                 font_size=50).set_color(GREEN).next_to(operacionDescripcion, DOWN, aligned_edge=LEFT)

        EjR4 = Tex(r"\textbf{Ejemplo en \R[4]:}",
                 tex_template = myTemplate,
                 font_size=50).set_color(GREEN).next_to(operacionDescripcion, DOWN, aligned_edge=LEFT)

        a    = nc.Vector( [0, 3, 6])
        b    = nc.Vector( [5, 1, 2])
        s1 = MathTex(a.latex(),        tex_template = myTemplate,)
        mas= MathTex(r"+",             tex_template = myTemplate,)
        s3 = MathTex(b.latex(),        tex_template = myTemplate,)
        igual = MathTex(r"=",              tex_template = myTemplate,)
        s5 = MathTex((a+b).latex(),    tex_template = myTemplate,)
        grp1 = VGroup(s1,mas,s3,igual,s5).arrange(RIGHT)

        self.add(operacionSuma)
        self.add(operacionDescripcion)

        with self.voiceover(text=r"""Podemos sumar dos vectores si ambos poseen el mismo número de
        componentes.""") as tracker:
            self.add(EjR3)
            self.add(grp1[0])
            self.add(grp1[2])

        # Definición de vector suma
        with self.voiceover(text=r"""El resultado es otro vector que se define componente a
        componente.""") as tracker:
            self.add(grp1[1])
            self.pausa_media()
            self.add(grp1[3])
            self.add(grp1[4][0][:2])
            self.add(grp1[4][0][-2:])

        with self.voiceover(text=r"""La primera es la suma de las primeras componentes de ambos
        vectores.""") as tracker:
            self.play(FadeIn(grp1[4][0][2]), run_time=tracker.duration/3)
            self.play(Circumscribe(grp1[0][0][2]), Circumscribe(grp1[2][0][2]), run_time=tracker.duration*2/3)
            self.pausa_corta()

        with self.voiceover(text=r"""La segunda es la suma de las segundas.""") as tracker:
            self.play(FadeIn(grp1[4][0][3]), run_time=tracker.duration/3)
            self.play(Circumscribe(grp1[0][0][3]), Circumscribe(grp1[2][0][3]), run_time=tracker.duration*2/3)
            self.pausa_corta()

        with self.voiceover(text=r"""Y así con todas las componentes
        del vector suma.""") as tracker:
            self.play(FadeIn(grp1[4][0][4]))
            self.pausa_corta()
            self.play(Circumscribe(grp1[0][0][4]), Circumscribe(grp1[2][0][4]) )
            self.pausa_corta(.3)
            self.play(FadeOut(grp1))

        self.pausa()

        v_generico_a  = nc.Vector(sp.symbols('a:5')[1:])
        vga = MathTex(v_generico_a.latex(), tex_template = myTemplate)

        v_generico_b  = nc.Vector(sp.symbols('b:5')[1:])
        vgb = MathTex(v_generico_b.latex(), tex_template = myTemplate)

        vgab = MathTex((v_generico_a + v_generico_b).latex(), tex_template = myTemplate)

        grp2 = VGroup(vga,mas,vgb,igual,vgab).arrange(RIGHT)
        with self.voiceover(text=r"""Por tanto, la siguiente expresión describe la suma de vectores en R
        4. """) as tracker:
            self.play(FadeTransform(EjR3,EjR4))
            self.play(FadeIn(grp2))
            self.pausa()

        with self.voiceover(text=r"""Definir la suma en R n requiere una estrategia distinta; una que no
        necesite escribir la lista completa de componentes. Piense que
        la lista puede ser muy larga para enes grandes.""") as tracker:
            self.wait(tracker.duration/3)
            self.play(Indicate(vga[0][4:-4]), Indicate(vgb[0][4:-4]), Indicate(vgab[0][4:-4]), run_time=2)
            self.pausa_corta()

        Defn = Tex(r"\textbf{Definición:}",
                 tex_template = myTemplate,
                 font_size=50).set_color(RED).next_to(operacionDescripcion, DOWN, aligned_edge=LEFT)

        with self.voiceover(text=r"""Una solución es definir la suma usando la notación
        descrita en el vídeo anterior. Con ella podemos expresar""") as tracker:
            self.play(FadeOut(grp2), FadeOut(EjR4), run_time=tracker.duration/3)
            self.add(Defn)

        cvab = MathTex(r"\elemRp{\Vect{a}+\Vect{b}}{i}", tex_template = myTemplate)
        cva  = MathTex(r"\eleVR{a}{i}", tex_template = myTemplate)
        cvb  = MathTex(r"\eleVR{b}{i}", tex_template = myTemplate)
        eq_suma = VGroup(cvab,igual,cva,mas,cvb).arrange(RIGHT).scale(1.5)

        donde = Tex("donde")
        indices = MathTex(r"i=1:n", tex_template = myTemplate)
        pc_indices = VGroup(donde,indices).arrange(RIGHT, buff=1)
        grp3 = VGroup(eq_suma, pc_indices).arrange(RIGHT, buff=1)

        with self.voiceover(text=r"""que la componente i-ésima del vector suma es igual a la suma de las i-ésimas componentes de
            self.play(FadeIn(grp3[0][:2], scale=1.5, rate_func=rate_functions.exponential_decay), run_time=2*tracker.duration/5)
            self.play(FadeIn(grp3[0][2:], scale=0.5, rate_func=rate_functions.exponential_decay), run_time=3*tracker.duration/5)
        with self.voiceover(text=r"""(donde el índice recorre los números naturales entre uno y n)""") as tracker:
            self.play(FadeIn(grp3[1]))
            self.pausa_corta()

        with self.voiceover(text=r"""Esta definición abstracta será muy util para demostrar algunas
        propiedades de las operaciones con vectores, pues arroja una
        primera regla de cálculo simbólico:""") as tracker:
            self.pausa(tracker.duration*2/3)
            self.play(Indicate(eq_suma[0][0][0]), Indicate(eq_suma[0][0][-3:]), Indicate(eq_suma[2][0][-2:]), Indicate(eq_suma[4


        with self.voiceover(text=r"""que la suma de las i ésimas componentes se puede sustituir por la
        i-ésima componente del vector suma.""") as tracker:
            source0 = MathTex(r"\eleVR{a}{i}+\eleVR{b}{i}", tex_template = myTemplate).next_to(grp3, DOWN, buff=1.2).scale(2)[0]
            target0 = MathTex(r"\elemRp{\Vect{a}+\Vect{b}}{i}", tex_template = myTemplate).next_to(grp3, DOWN, buff=1.2).scale(2
            source1 = target0.copy()
            target1 = source0.copy()

            VGroup(source0,target0)
            self.add(source0)
            transform_index0 = [
                [0,1,2,3,4,5,6],
                [1,0,4,2,3,5,6]
            ]
            self.play(
                *[
                    ReplacementTransform(source0[i],target0[j], rate_func=rate_functions.smooth)
                    for i,j in zip(*transform_index0)
                ],
                run_time=tracker.duration)

        with self.voiceover(text=r"""Y la i-ésima componente de una suma se puede sustituir por la
        suma de las i ésimas componentes.""") as tracker:
            self.play(ReplacementTransform(target0,source1))

            VGroup(source1,target1)
            transform_index1 = [
                [0,1,2,3,4,5,6],
                [1,0,3,4,2,5,6]
            ]
            self.play(
                *[
                    ReplacementTransform(source1[i],target1[j], rate_func=rate_functions.smooth)
                    for i,j in zip(*transform_index1)
                ],
                run_time=tracker.duration)

        with self.voiceover(text=r"""Esta regla se denomina propiedad distributiva del operador selector
        respecto de la suma.""") as tracker:
            self.play(FadeOut(target1))
            self.play(Indicate(eq_suma[0][0][0]), Indicate(eq_suma[0][0][-3:]), Indicate(eq_suma[2][0][-2:]), Indicate(eq_suma[4
            self.pausa()
```

1.1.2. Escena 2 - Propiedad conmutativa de la suma

```
class L01_V02_E02_PropiedadConmutativaDeLaSuma(MiEscenaConVoz):
    def construct(self):
        self.set_speech_service( AzureService(voice="es-ES-AlvaroNeural" ) )
        #self.set_speech_service(GTTSService(lang="es", tld="com"))

        myTemplate = TexTemplate()
        myTemplate.add_to_preamble(r"""\usepackage{nacal} """)

        self.creditos(7)

        # Definición de vector suma
        operacionSuma = Tex(r"Suma de vectores de \R[n]",
                         tex_template = myTemplate, font_size=70
                         ).to_edge(UP).set_color(BLUE)

        self.add(operacionSuma)

        str0  = MathTex(r"\elemRp{\Vect{a}+\Vect{b}}{i}", tex_template = myTemplate).scale(2)[0]
        str1  = MathTex(r"\elemRp{\Vect{b}+\Vect{a}}{i}", tex_template = myTemplate).scale(2)[0]
        igual = MathTex(r"=",             tex_template = myTemplate,).scale(2)[0]
        vgr1  = VGroup(str0, igual, str1).arrange(RIGHT, buff=1)

        with self.voiceover(text=r"""Antes de continuar, demostremos la propiedad conmutativa de la suma
        de vectores. Es decir, que el orden en que se sumen los
        vectores es irrelevante.""") as tracker:
            self.play(FadeIn(str0[1:-3]), FadeIn(vgr1[1]), FadeIn(str1[1:-3]))
            self.pausa(tracker.duration/2)
            self.play(Indicate(str0[1]),  Indicate(str1[-4]), run_time=tracker.duration/4)
            self.play(Indicate(str0[-4]),  Indicate(str1[1]), run_time=tracker.duration/4)
            self.pausa_corta()

        with self.voiceover(text=r"""Sabemos que dos vectores son iguales si lo son sus correspondientes
        listas de componentes. Por tanto, para demostrar la igualdad
        entre vectores debemos probar la igualdad componente a
        componente.""") as tracker:
            self.pausa(tracker.duration*2/5)
            self.play(FadeIn(str0[0]), FadeIn(str0[-3:]), FadeIn(str1[0]), FadeIn(str1[-3:]) )
            self.pausa(tracker.duration/4)
            self.play(Indicate(str0[-2:]), Indicate(str1[-2:]), run_time=tracker.duration/4)
            self.pausa_corta()

        with self.voiceover(text=r"""Para ello comenzaremos escribiendo uno cualquiera de
        sus lados. Después operaremos hasta obtener la expresión del
        lado opuesto de la igualdad.""") as tracker:
            self.pausa(tracker.duration/4)
            self.play(Indicate(vgr1[0]))
            self.play(Indicate(vgr1[2]))

        vgr2=vgr1.copy().scale(1/2).next_to(operacionSuma, DOWN).to_edge(LEFT)
        vgr3=vgr1.copy().scale(1/2).to_edge(LEFT)
        item1 = MathTex(r"\eleVR{x}{i} \in \R",tex_template = myTemplate)
        item2 = MathTex(r"\alpha + \beta = \beta + \alpha\quad (\alpha,\beta\in\R)",tex_template = myTemplate)
        item3 = MathTex(r"\elemRp*{\Vect{x}+\Vect{y}}{i} = \eleVR{x}{i} + \eleVR{y}{i}",tex_template = myTemplate)
        items = VGroup(item1, item2, item3).arrange(DOWN).scale(.8).align_to(vgr2, UP).to_edge(RIGHT).shift(DOWN*0.15)
        box =  SurroundingRectangle(items, color=YELLOW )

        paso1 = MathTex(r"=\eleVR{a}{i}+\eleVR{b}{i}",tex_template = myTemplate).next_to(vgr3[0],RIGHT)
        paso2 = MathTex(r"=\eleVR{b}{i}+\eleVR{a}{i}",tex_template = myTemplate).next_to(paso1,DOWN, aligned_edge=LEFT)
        paso3 = MathTex(r"=\elemRp{\Vect{b}+\Vect{a}}{i}",tex_template = myTemplate).next_to(paso2,DOWN, aligned_edge=LEFT)
        demo = VGroup(paso1, paso2, paso3)

        with self.voiceover(text=r"""Con operar nos referimos a sustituir una expresión por otra que
        sabemos que es equivalente. Para esta demostración solo necesitamos considerar tres cosas""") as tracker:
            self.play(FadeTransformPieces(vgr1,vgr2), run_time=tracker.duration/2 )
            self.add(box,items)
            self.pausa_muy_larga()

        with self.voiceover(text=r"""que los elementos de un vector son números reales, que entre
        números reales la suma es conmutativa, y que el operador
        selector es distributivo respecto de la suma""") as tracker:
            self.play(Indicate(items[0]), run_time=tracker.duration/3 )
            self.play(Indicate(items[1]), run_time=tracker.duration/3 )
            self.play(Indicate(items[2]), run_time=tracker.duration/3 )

        with self.voiceover(text=r"""Comencemos escribiendo uno de los lados, por ejemplo el izquierdo.""") as tracker:
            self.play( FadeTransformPieces(vgr2[0].copy(),vgr3[0]), FadeToColor(vgr2[0], color=TEAL), run_time=tracker.duration/
            self.pausa_media()

        with self.voiceover(text=r"""En primer lugar, el operador selector es distributivo respecto de la suma""") as tracker:
            self.play(Indicate(items[2], run_time=tracker.duration/2) )
            self.play(FadeIn(demo[0],    run_time=tracker.duration/2) )

        with self.voiceover(text=r"""En segundo lugar, dado que los componentes son números reales, el
        resultado no cambia si intercambiamos el orden de su suma.""") as tracker:
            self.play(Indicate(items[:2], run_time=tracker.duration/2) )
            self.play(FadeIn(demo[1],     run_time=tracker.duration/2) )

        with self.voiceover(text=r"""Por último, el operador selector es distributivo respecto de la suma""") as tracker:
            self.play(Indicate(items[2], run_time=tracker.duration/2) )
            self.play(FadeIn(demo[2],    run_time=tracker.duration/2) )

        with self.voiceover(text=r"""Con esto hemos terminado la demostración.""") as tracker:
            self.play(FadeToColor(vgr2[0], color=TEAL))
            self.play(Indicate(vgr3[0]), Indicate(demo[2]), FadeToColor(vgr2[1:], color=TEAL), run_time=tracker.duration)
            self.pausa()
```

1.1.3. Escena 3 - Interpretación geométrica de la suma en $\mathbb{R}^2$

```
class L01_V02_E03_SumaEnR2(MiEscenaConVoz):
    def construct(self):
        self.set_speech_service( AzureService(voice="es-ES-AlvaroNeural" ) )
        #self.set_speech_service(GTTSService(lang="es", tld="com"))
        myTemplate = TexTemplate()
        myTemplate.add_to_preamble(r"""\usepackage{nacal} """)

        self.creditos()

        axes = NumberPlane(x_range=(-4.5, 6.5, 1),
                           y_range=(-1.5, 6.5, 1),
                           background_line_style={
                               "stroke_width":  3,
                               "stroke_opacity": 0.4 }
                           ).add_coordinates()


        item0 = MathTex(r"\elemRp*{\Vect{a}+\Vect{b}}{i} = \eleVR{a}{i} + \eleVR{b}{i}",tex_template = myTemplate)
        item1 = MathTex(r"\Vect{a}+\Vect{x} = \Vect{x}+\Vect{a}",tex_template = myTemplate).next_to(item0, DOWN, buff=0.5)
        props_suma = VGroup(item0,item1).scale(1.5)

        with self.voiceover(text=r"""Que la operación suma sea una
        operación componente a componente""") as tracker:
            self.play(FadeIn(props_suma[0]),
                      run_time=tracker.duration)

        with self.voiceover(text=r"""y que sea conmutativa""") as tracker:
            self.play(Write(props_suma[1], run_time=tracker.duration/5))

        with self.voiceover(text=r"""dota a la suma de interpretación
        geométrica tanto en R 2 como en R 3.""") as tracker:
            self.pausa(tracker.duration)
            self.play(FadeOut(props_suma))

        x     = VectorR2([4,5], color=GREEN_B)
        x_dot = x.dot(axes, radio=0.12)
        x_tex = x.tex.scale(1.4)
        vgr_x = VGroup(x.tex).next_to(x_dot, RIGHT).shift(RIGHT*.1)
        x_v_line = x.v_line(axes)
        x_h_line = x.h_line(axes)
        with self.voiceover(text=r""" Para verlo debemos interpretar
        los vectores como puntos en el espacio, de manera que las
        componentes de cada vector sean las coordenadas de un
        punto.""") as tracker:
            self.play(Create(axes), run_time=tracker.duration/2)

        with self.voiceover(text=r"""En R 2, el convenio es considerar
        que la primera componente es la coordenada respecto al eje
        horizontal""") as tracker:
            self.add(vgr_x)
            self.pausa(tracker.duration/3)
            self.play(Circumscribe(x_tex[0][1]), Indicate(x_v_line), run_time=tracker.duration*2/3)

        with self.voiceover(text=r"""y la segunda como la coordenada
        respecto al eje vertical.""") as tracker:
            self.play(Circumscribe(x_tex[0][2]), Indicate(x_h_line), run_time=tracker.duration/2)
            self.add(x_dot)
            self.play(Indicate(x_dot), run_time=tracker.duration/2)

        with self.voiceover(text=r"""Consecuentemente, vectores
        distintos corresponden a puntos distintos.""") as tracker:
            self.pausa(tracker.duration)
            self.play(FadeOut(vgr_x, x_dot, x_h_line, x_v_line))

        a = VectorR2([0,0], rpr='colum', color=YELLOW)
        a_dot = a.dot(axes, radio=0.12)
        a_tex = a.tex
        vgr_a = VGroup(a.tex).next_to(a_dot, DOWN).shift(LEFT*.5)
        with self.voiceover(text=r"""El vector cero corresponde con el
        origen del sistema de coordenadas""") as tracker:
            self.play(Indicate(a_dot), Indicate(a_tex), run_time=tracker.duration)
            self.pausa

        #añado punto en el eje horizontal quitando el anterior
        b1 = VectorR2([3,0], rpr='colum')
        b1_dot = b1.dot(axes, radio=0.12)
        b1_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(b1_dot)
        b1_tex = b1.tex
        vgr_b1= VGroup(b1_tex).next_to(b1_dot, DOWN)

        with self.voiceover(text=r"""La primera componente de un
        vector indica su coordenada respecto al eje horizontal. Los
        valores positivos corresponden a posiciones a la derecha del
        origen de coordenadas.""") as tracker:
            self.play(FadeOut(a_dot, a_tex), FadeIn(b1_diamond))

        #lo muevo y le pongo etiqueta
        b1n    = VectorR2([-2.5,0])
        vgr_b1n= VGroup(b1n.tex).next_to(b1n.dot(axes, radio=0.12), DOWN)
        b1n_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(b1n.dot(
        with self.voiceover(text=r"""Y valores negativos a posiciones a la
        izquierda. Así, el vector 3 0 corresponde al punto del eje
        horizontal que está 3 unidades a la derecha del origen.""") as tracker:
            self.play(ReplacementTransform(b1_diamond, b1n_diamond), rate_function=exponential_decay, run_time=tracker.duration/
            self.play(ReplacementTransform(b1n_diamond, b1_dot), FadeIn(b1.tex), rate_function=smooth, run_time=2*tracker.durati

        #añado punto inicial en el eje vertical
        b2i    = VectorR2([0,4])
        vgr_b2i= VGroup(b2i.tex).next_to(b2i.dot(axes, radio=0.12), LEFT)
        #b2i_dot = b2i.dot(axes, radio=0.12)
        b2i_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(b2i.dot(
        with self.voiceover(text=r"""La segunda componente indica la
        coordenada respecto al eje vertical. Valores positivos
        corresponden a posiciones por encima del origen de
        coordenadas. """) as tracker:
            self.add(b2i_diamond)
            self.pausa

        # punto con oordenada negativa
        b2n    = VectorR2([0,-1])
        vgr_b2n= VGroup(b2n.tex).next_to(b2n.dot(axes, radio=0.12), DOWN)
        #b2n_dot = b2n.dot(axes, radio=0.12)
        b2n_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(b2n.dot(

        #lo muevo y pongo etiqueta
        b2    = VectorR2([0,2], rpr='colum')
        b2_dot = b2.dot(axes, radio=0.12)
        b2_tex = b2.tex
        vgr_b2= VGroup(b2_tex).next_to(b2_dot, LEFT)
        with self.voiceover(text=r"""Y valores negativos a posiciones
        por debajo. Por tanto el vector 0 2 corresponde al punto del
        eje vertical que está 2 unidades por encima del origen.""")  as tracker:
            self.play(ReplacementTransform(b2i_diamond, b2n_diamond, rate_function=exponential_decay, run_time= tracker.duration
            self.play(ReplacementTransform(b2n_diamond, b2_dot, rate_function=smooth,  run_time=2*tracker.duration/3))
            self.add(b2_tex)
            self.pausa(n=3)

        b     = VectorR2([3,2], color=TEAL_A)
        b_dot = b.dot(axes, radio=0.12)
        b_tex = b.tex
        vgr_b = VGroup(b_tex).next_to(b_dot, RIGHT)

        # arriba añadir (0,3)+(1,0) = (3,1)
        suma1_gr = VGroup(VectorR2([3,0]).tex,
                          MathTex(r"+"),
                          VectorR2([0,2]).tex,
                          MathTex(r"="),
                          b.tex.copy(),
                          ).arrange(RIGHT).to_corner(UL)

        with self.voiceover(text=r"""Ahora consideremos la suma de
        estos dos vectores. Se realiza componente a componente.""")  as tracker:
            self.add(suma1_gr[:3])
            #self.pausa(3*tracker.duration/4)
            #self.play(FadeIn(suma1_gr[3:]), run_time=tracker.duration/4)
            #self.pausa_larga

        with self.voiceover(text=r"""Por una parte se suman las
        coordenadas respecto al eje horizontal, y por otra las
        coordenadas correspondientes al eje vertical. Así, el vector
        suma es el vector 3 2.""")  as tracker:
            self.play(Circumscribe(suma1_gr[0][0][1]),
                      Circumscribe(suma1_gr[2][0][1]),
                      run_time=tracker.duration/3
                      )
            self.play(Circumscribe(suma1_gr[0][0][2]),
                      Circumscribe(suma1_gr[2][0][2]),
                      run_time=tracker.duration/3
                      )
            self.play(FadeIn(suma1_gr[3:]), run_time=tracker.duration/3)
            self.pausa_larga

        # pintar b con un punto y ejes y etiqueta
        b_v_line = b.v_line(axes)
        b_h_line = b.h_line(axes)
        with self.voiceover(text=r"""Sus componentes nos indican que
        el punto está tres unidades a la derecha del origen y dos
        unidades por encima.""")  as tracker:
            self.play(FadeIn(b_dot, b_tex, b_v_line, b_h_line))
            self.pausa

        # Añadir flechas ejes (quitando puntos) y desplazar para mostrar suma
        flechab1 = b1.arrow(axes)
        flechab2 = b2.arrow(axes)
        with self.voiceover(text=r"""Señalando la posición de cada
        sumando con una flecha, podemos interpretar dicha flecha como
        una indicación para llegar al punto.""") as tracker:
            self.play(GrowArrow(flechab1),
                      FadeOut(b1_dot),
                      GrowArrow(flechab2),
                      FadeOut(b2_dot),
                      FadeOut(b_dot) )

        with self.voiceover(text=r"""Por ejemplo, al primer sumando se
        llega desplazandose desde el origen tres unidades a la
        derecha. De este modo dotamos a la suma de interpretación
        geométrica.""") as tracker:
            self.play(Indicate(b1_tex),
                      run_time=tracker.duration/2)

        # SUMA b1 + b2
        a_dot_copy  = a_dot.copy()
        b1_dot_copy = b1_dot.copy()
        b_dot_copy  = b_dot.copy()
        with self.voiceover(text=r"""Sumar el primer vector con el
        segundo corresponde a seguir las indicaciones del primer
        vector""") as tracker:
            self.play(#Indicate(flechab1),
                      Indicate(b1_tex),
                      Indicate(suma1_gr[0]),
                      ReplacementTransform(a_dot_copy, b1_dot_copy),
                      run_time=tracker.duration)

        with self.voiceover(text=r"""y luego seguir las indicaciones
        del segundo.""") as tracker:
            self.play(Indicate(b2_tex),
                      Indicate(suma1_gr[2]),
                      #Wiggle(flechab2),
                      ReplacementTransform(b1_dot_copy, b_dot_copy),
                      run_time=tracker.duration)

        self.play(FadeOut(b_dot_copy))

        # SUMA b2 + b1
        a_dot_copy  = a_dot.copy()
        b2_dot_copy = b2_dot.copy()
        b_dot_copy  = b_dot.copy()
        with self.voiceover(text=r"""Pero invertir el orden y seguir
        primero las indicaciones del segundo vector""") as tracker:
            self.play(#Wiggle(flechab2),
                      Indicate(b2_tex),
                      Indicate(suma1_gr[2]),
                      ReplacementTransform(a_dot_copy, b2_dot_copy),
                      run_time=tracker.duration)

        flechab = b.arrow(axes)
        with self.voiceover(text=r"""y después las indicaciones del
        primero, nos conduce al mismo vector suma.""") as tracker:
            self.play(Indicate(b1_tex),
                      Indicate(suma1_gr[0]),
                      #Wiggle(flechab1),
                      ReplacementTransform(b2_dot_copy, b_dot_copy),
                      run_time=tracker.duration/2)
            self.play(GrowArrow(flechab),
                      FadeOut(b_dot_copy),
                      FadeOut(flechab1, b1_tex),
                      FadeOut(flechab2, b2_tex),
                      run_time=tracker.duration/2)

        self.pausa
        self.play(FadeOut(flechab), FadeIn(b_dot))
        self.pausa_media

        # arriba añadir (3,2)+(-2,1) = (1,3)
        c     = VectorR2([-2,1], color=PURPLE_A)
        c_dot = c.dot(axes, radio=0.12)
        c_tex = c.tex
        c_v_line = c.v_line(axes)
        c_h_line = c.h_line(axes)
        vgr_c = VGroup(c.tex).next_to(c.dot(axes, radio=0.12), LEFT)

        d     = VectorR2([1,3], color=YELLOW_A)
        d_dot = d.dot(axes, radio=0.12)
        d_tex = d.tex
        d_v_line = d.v_line(axes)
        d_h_line = d.h_line(axes)
        vgr_d = VGroup(d.tex).next_to(d.dot(axes, radio=0.12), UP)

        suma2_gr = VGroup(b.tex.copy(),
                          MathTex(r"+"),
                          c.tex.copy(),
                          MathTex(r"="),
                          d.tex.copy(),
                          ).arrange(RIGHT).to_corner(UL)

        with self.voiceover(text=r"""Veamos otro ejemplo.""") as tracker:
            self.play(FadeOut(suma1_gr),
                      run_time=tracker.duration )
            self.pausa_corta

        with self.voiceover(text=r"""Sumemos el último vector con el
        vector -2 1.""") as tracker:
            self.play(FadeIn(c_dot, c_tex, c_v_line, c_h_line))

        with self.voiceover(text=r"""La suma de ambos es el vector 1
        3.""") as tracker:
            self.add(suma2_gr)
            self.pausa
            self.play(FadeOut(b_h_line, b_v_line, c_h_line, c_v_line))
            self.add(d_dot, d.tex, d_v_line, d_h_line)
            self.pausa_larga

        # Añadir flechas ejes (quitando puntos) y desplazar para mostrar suma
        flechab = b.arrow(axes)
        flechac = c.arrow(axes)
        flechad = d.arrow(axes)

        with self.voiceover(text=r"""Una vez más, señalemos los
        vectores con flechas.""") as tracker:
            self.play(FadeOut(d_dot), #d_h_line, d_v_line),
                      GrowArrow(flechab),
                      FadeOut(b_dot),
                      GrowArrow(flechac),
                      FadeOut(c_dot))
            self.pausa_corta

        line_graph_b = axes.plot_line_graph(
            x_values = [-2, 1],
            y_values = [1, 3],
            line_color=TEAL_E,
            add_vertex_dots=False,
            stroke_width = 3,
        )

        line_graph_c = axes.plot_line_graph(
            x_values = [3, 1],
            y_values = [2, 3],
            line_color=PURPLE_E,
            add_vertex_dots=False,
            stroke_width = 3,
        )

        self.add(line_graph_b,line_graph_c)
        a_dot_copy  = a_dot.copy()
        b_dot_copy  = b_dot.copy()
        d_dot_copy  = d_dot.copy()

        a1_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(a_dot)
        a2_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(a_dot)
        a1_diamond_copy = a1_diamond.copy()
        a2_diamond_copy = a2_diamond.copy()
        b1_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(b1_dot)
        b2_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(b2_dot)
        c1_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(Dot(axes.
        c2_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(Dot(axes.
        d1_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(Dot(axes.
        d2_diamond = Square(color=BLUE, fill_opacity=1, side_length=.12, fill_color=ORANGE).rotate(45*DEGREES).move_to(Dot(axes.


        with self.voiceover(text=r"""De nuevo, sumar el primer vector
        con el segundo corresponde a seguir las indicaciones del
        primer vector""") as tracker:
            self.play(Indicate(suma2_gr[0]),
                      ReplacementTransform(a_dot_copy, b_dot_copy),
                      ReplacementTransform(a1_diamond, b1_diamond),
                      ReplacementTransform(a2_diamond, b2_diamond),
                      GrowArrow(flechab1),
                      GrowArrow(flechab2),
                      run_time=3)
            self.play(FadeOut(flechab1,
                              flechab2))

        c1 = VectorR2([-2,0])
        c2 = VectorR2([0,1])
        flechac1 = c1.arrow(axes)
        flechac2 = c2.arrow(axes)
        flechac1d = flechac1.copy().move_to(axes.c2p(2,2,0))
        flechac2d = flechac2.copy().move_to(axes.c2p(3,2.5,0))

        with self.voiceover(text=r"""y luego seguir las indicaciones
        del segundo.""") as tracker:
            self.play(Indicate(suma2_gr[2]),
                      ReplacementTransform(b_dot_copy, d_dot_copy),
                      ReplacementTransform(b1_diamond, d1_diamond),
                      ReplacementTransform(b2_diamond, d2_diamond),
                      GrowArrow(flechac1d),
                      GrowArrow(flechac2d),
                      run_time=3)
            self.play(FadeOut(d_dot_copy,
                              d1_diamond,
                              d2_diamond,
                              flechac1d,
                              flechac2d))

        a_dot_copy  = a_dot.copy()
        c_dot_copy  = c_dot.copy()
        d_dot_copy  = d_dot.copy()
        flechab1d = flechab1.copy().move_to(axes.c2p(-0.5,1,0))
        flechab2d = flechab2.copy().move_to(axes.c2p(- 2,2,0))
        with self.voiceover(text=r"""Pero invertir el orden y seguir
        primero las indicaciones del segundo vector""") as tracker:
            self.play(Indicate(suma2_gr[2]),
                      ReplacementTransform(a_dot_copy, c_dot_copy),
                      ReplacementTransform(a1_diamond_copy, c1_diamond),
                      ReplacementTransform(a2_diamond_copy, c2_diamond),
                      GrowArrow(flechac1),
                      GrowArrow(flechac2),
                      run_time=3)
            self.play(FadeOut(flechac1,
                              flechac2))

        with self.voiceover(text=r"""y después las indicaciones del
        primero, nos conduce al mismo punto.""") as tracker:
            self.play(Indicate(suma2_gr[0]),
                      ReplacementTransform(c_dot_copy, d_dot_copy),
                      ReplacementTransform(c1_diamond, d1_diamond),
                      ReplacementTransform(c2_diamond, d2_diamond),
                      GrowArrow(flechab1d),
                      GrowArrow(flechab2d),
                      run_time=2*tracker.duration/3)
            self.play(FadeOut(d_dot_copy),
                      FadeOut(d1_diamond),
                      FadeOut(d2_diamond),
                      FadeOut(d_v_line),
                      FadeOut(d_h_line),
                      FadeOut(flechab1d,
                              flechab2d))
            self.play(GrowArrow(flechad),
                      run_time=tracker.duration/3)
            self.pausa_larga


        with self.voiceover(text=r"""Esta descripción geométrica de la
        suma, donde los sumandos forman un vértice de un
        paralelogramo, y su suma es la diagonal que parte de dicho
        vértice se denomina "regla del paralelogramo".""") as tracker:
            self.play(Indicate(flechab),
                      Indicate(flechac),
                      run_time=tracker.duration/2 )
            self.play(Indicate(flechad),
                      run_time=tracker.duration/2 )

        with self.voiceover(text=r"""A pesar de la utilidad de las
        flechas, recuerde que un vector es una lista de números, y que
        podemos hacer corresponder dichos números con las coordenadas
        de un punto en el espacio. Por ello, la representación
        geométrica del vector es el punto. La flecha tan solo lo
        señala.""") as tracker:
            self.play(Indicate(b_tex),
                      Indicate(c_tex),
                      Indicate(d_tex),
                      run_time=tracker.duration/2 )
            self.play(FadeOut(flechab,
                              flechac,
                              flechad,
                              line_graph_b,
                              line_graph_c),
                      FadeIn(b_dot, c_dot, d_dot),
                      run_time=tracker.duration/2 )

        b_dot_copia=Dot(axes.c2p(*b.coords), radius=0.01)
        c_dot_copia=Dot(axes.c2p(*c.coords), radius=0.01)
        d_dot_copia=Dot(axes.c2p(*d.coords), radius=0.01)
        with self.voiceover(text=r"""Una de las dificultades para
        representar los puntos es que su dimensión es cero.""") as tracker:
            self.play(
                Transform(b_dot, b_dot_copia),
                Transform(c_dot, c_dot_copia),
                Transform(d_dot, d_dot_copia),
                run_time = 6*tracker.duration/5 )

        with self.voiceover(text=r"""Una solución es indicar para
        cada punto su coordenada en el eje horizontal (es decir, el
        primer número de la lista).""") as tracker:
            self.play(FadeIn(b_v_line),
                      FadeIn(c_v_line),
                      FadeIn(d_v_line),
                      run_time = tracker.duration/2)
            self.play(Circumscribe(b_tex[0][1]),
                      Circumscribe(c_tex[0][1:3]),
                      Circumscribe(d_tex[0][1]),
                      run_time = tracker.duration/2)

        with self.voiceover(text=r"""y su coordenada en el eje
        vertical (es decir, el segundo número de la lista).""") as tracker:
            self.play(FadeIn(b_h_line),
                      FadeIn(c_h_line),
                      FadeIn(d_h_line),
                      run_time = tracker.duration/2)
            self.play(Circumscribe(b_tex[0][2]),
                      Circumscribe(c_tex[0][3]),
                      Circumscribe(d_tex[0][2]),
                      run_time = tracker.duration/2)

        with self.voiceover(text=r"""Sin embargo, la representación
        más frecuente son las flechas. Se ven bien y arrojan una
        interpretación intuitiva de la suma de vectores.  """) as tracker:
            self.play(FadeIn(flechab,flechac,flechad),
                      FadeOut(b_h_line, b_v_line),
                      FadeOut(c_h_line, c_v_line),
                      FadeOut(d_h_line, d_v_line),
                      run_time = tracker.duration/2 )
            self.play(FadeIn(line_graph_b, line_graph_c),
                      run_time = tracker.duration/2 )

        with self.voiceover(text=r"""Pero no debe olvidar que nuestra
        definición de vector de Rn es que es una lista de números. Y
        que su representación geométrica hace corresponder dichos
        números con las coordenadas de puntos en el espacio. Por
        tanto, cuando veamos un vector representado con una flecha,
        debemos recordar que el vector no es la flecha. El vector es
        el punto señalado por la flecha.""") as tracker:
            self.play(FadeOut(line_graph_b, line_graph_c),
                      Indicate(b_tex),
                      Indicate(c_tex),
                      Indicate(d_tex),
                      run_time=tracker.duration/2 )
            self.play(FadeOut(flechab, flechac, flechad,),
                      FadeIn(b.dot(axes)),
                      FadeIn(c.dot(axes)),
                      FadeIn(d.dot(axes)),
                      run_time=tracker.duration/2)

        self.pausa_muy_larga
```

1.1.4. Escena 4 - Interpretación geométrica de la suma en $\mathbb{R}^3$

1. Escena 4(voz)

```
class L01_V02_E04_SumaEnR3_voz(MiEscenaConVoz):
    def construct(self):
        self.set_speech_service( AzureService(voice="es-ES-AlvaroNeural" ) )
        #self.set_speech_service(GTTSService(lang="es", tld="com"))
        myTemplate = TexTemplate()
        myTemplate.add_to_preamble(r"""\usepackage{nacal} """)

                # Portada
        titulo = Title(r"Interpretación de la suma en $\R[3]$",
                     tex_template = myTemplate,
                     font_size=70).set_color(BLUE)

        a    = nc.Vector(sp.symbols('a:4')[1:])
        b    = nc.Vector(sp.symbols('b:4')[1:])
        s1 = MathTex(a.latex(),        tex_template = myTemplate,)
        mas= MathTex(r"+",             tex_template = myTemplate,)
        s3 = MathTex(b.latex(),        tex_template = myTemplate,)
        igual = MathTex(r"=",             tex_template = myTemplate,)
        s5 = MathTex((a+b).latex(),    tex_template = myTemplate,)
        grp1 = VGroup(s1,mas,s3,igual,s5,igual.copy(),s3.copy(),mas.copy(),s1.copy()).arrange(RIGHT)

        self.creditos(17)

        with self.voiceover(text=r"""La representación geométrica en
        R3 es similar. """) as tracker:
            self.add(titulo)
            self.play(FadeIn(grp1[0]))

        with self.voiceover(text=r"""El convenio es interpretar las
        dos primeras componentes como coordenadas respecto a un plano
        horizontal""") as tracker:
            self.play(Indicate(grp1[0][0][2:6]),
                      run_time=tracker.duration)

        with self.voiceover(text=r"""y la tercera como la coordenada respecto a un eje
        perpendicular al plano.""") as tracker:
            self.play(Indicate(grp1[0][0][6:8]),
                      run_time=tracker.duration)

        with self.voiceover(text=r"""De nuevo, como la suma se realiza
        componente a componente y es conmutativa""") as tracker:
            self.play(FadeIn(grp1[1:5]),
                      run_time=tracker.duration/2)
            self.play(FadeIn(grp1[5:]),
                      run_time=tracker.duration/2)

        with self.voiceover(text=r"""su representación geométrica en
        R3 también verifica la regla del paralelogramo.""") as tracker:
            self.pausa(tracker.duration)
```

2. Escena 4 (Visión 3D)

```
class L01_V02_E04_SumaEnR3_3D(ThreeDScene):
    <<Créditos en distintas partes de la pantalla>>

    def construct(self):
        axes = ThreeDAxes()

        x_label = axes.get_x_axis_label(Tex("1ª comp."))
        y_label = axes.get_y_axis_label(Tex("2ª comp.")).shift(UP * 2.4).shift(LEFT * 0.6)


        self.creditos(17)

        # zoom out so we see the axes
        self.set_camera_orientation(zoom=0.5)

        self.play(FadeIn(axes), FadeIn(x_label), FadeIn(y_label))

        self.wait(1)

        # animate the move of the camera to properly see the axes
        self.move_camera(phi=75 * DEGREES, theta=60 * DEGREES, zoom=1, run_time=1.5)

        # built-in updater which begins camera rotation
        self.begin_ambient_camera_rotation(rate=0.2)

        self.wait(2)


        b     = VectorR3([3,2,3], color=TEAL_A)
        c     = VectorR3([-2,1,1], color=PURPLE_A)
        d     = VectorR3([1,3,4], color=YELLOW_A)

        b_dot = b.dot(axes)
        c_dot = c.dot(axes)
        d_dot = d.dot(axes)

        line_x = Line3D(start=np.array(axes.c2p(3,0,0,)), end=np.array(axes.c2p(3,2,0)), thickness=0.01)
        line_y = Line3D(start=np.array(axes.c2p(0,2,0,)), end=np.array(axes.c2p(3,2,0)), thickness=0.01)
        line_z = Line3D(start=np.array(axes.c2p(3,2,0,)), end=np.array(axes.c2p(3,2,3)), thickness=0.01)

        flechab = b.arrow(axes)
        flechac = c.arrow(axes)
        flechad = d.arrow(axes)

        linebd = Line3D(start=np.array(axes.c2p(*b.coords)), end=np.array(axes.c2p(*d.coords)), thickness=0.01)
        linecd = Line3D(start=np.array(axes.c2p(*c.coords)), end=np.array(axes.c2p(*d.coords)), thickness=0.01)


        self.play(FadeIn(line_x))
        self.play(FadeIn(line_y))
        self.wait(4)
        self.play(FadeIn(line_z))
        self.add(b_dot)

        self.wait(1.5)

        self.play(FadeIn(flechab),
                  FadeOut(b_dot),)

        self.wait(1.5)

        self.play(FadeIn(flechac))
        self.play(FadeOut(line_x, line_y, line_z))

        self.wait(1.5)

        self.play(FadeIn(linebd),
                  FadeIn(linecd),)

        self.wait(1.5)

        self.play(FadeIn(flechad))

        self.wait(2)

        #self.play(FadeOut(flechab, flechac, flechad, linebd, linecd),
        #          FadeIn(b_dot, c_dot, d_dot))

        #self.wait(2)


        #self.move_camera(phi=0 * DEGREES, theta=0 * DEGREES, zoom=1, run_time=1.5)

        #self.wait(2)
```

- Fusión audio y vídeo poca calidad

```
rm -f L01_V02_E04_SumaEnR3.mp4
ffmpeg -i L01_V02_E04_SumaEnR3_3D.mp4 -i L01_V02_E04_SumaEnR3_voz.mp4 -c:v copy -c:a aac -strict experimental L01_V02_E04_
#mv L01_V02_E04_SumaEnR3_voz.srt L01_V02_E04_SumaEnR3.srt
mkdir -p aux_movie_files
mv L01_V02_E04_SumaEnR3_3D.mp4 aux_movie_files/
mv L01_V02_E04_SumaEnR3_voz.mp4 aux_movie_files/
```

- Fusión audio y vídeo poca calidad HD1080

```
rm -f L01_V02_E04_SumaEnR3.mp4
ffmpeg -i L01_V02_E04_SumaEnR3_3D.mp4 -i L01_V02_E04_SumaEnR3_voz.mp4 -c:v copy -c:a aac -strict experimental L01_V02_E04_
cp L01_V02_E04_SumaEnR3_voz.srt L01_V02_E04_SumaEnR3.srt
mkdir -p aux_movie_files
mv L01_V02_E04_SumaEnR3_3D.mp4 aux_movie_files/
mv L01_V02_E04_SumaEnR3_voz.mp4 aux_movie_files/
```

1.1.5. Escena 5 - Interpretación geométrica de la suma en $\mathbb{R}^n$

1. Escena 5 (voz)

```
class L01_V02_E05_SumaEnRn_voz(MiEscenaConVoz):
    def construct(self):
        self.set_speech_service( AzureService(voice="es-ES-AlvaroNeural" ) )
        #self.set_speech_service(GTTSService(lang="es", tld="com"))
        myTemplate = TexTemplate()
        myTemplate.add_to_preamble(r"""\usepackage{nacal} """)

                # Portada
        titulo = Title(r"Interpretación de la suma en $\R[n]$",
                    tex_template = myTemplate,
                    font_size=70).set_color(BLUE)

        a    = nc.Vector(sp.symbols('a:4')[1:])
        b    = nc.Vector(sp.symbols('b:4')[1:])
        s1 = MathTex(a.latex(),        tex_template = myTemplate,)
        mas= MathTex(r"+",             tex_template = myTemplate,)
        s3 = MathTex(b.latex(),        tex_template = myTemplate,)
        igual = MathTex(r"=",             tex_template = myTemplate,)
        s5 = MathTex((a+b).latex(),    tex_template = myTemplate,)
        grp1 = VGroup(s1,mas,s3,igual,s5,igual.copy(),s3.copy(),mas.copy(),s1.copy()).arrange(RIGHT)

        self.creditos(3)

        with self.voiceover(text=r"""Los vectores en Rn son puntos en
        un espacio ene-dimensional. Para representarlos sería
        necesario dibujar tantos ejes de coordenadas como elementos
        tiene el vector. Esto no es posible cuando el número de
        componentes es mayor a tres.""")  as tracker:
            self.add(titulo)
            self.play(FadeIn(grp1[0]))

        with self.voiceover(text=r"""No obstante, sí que podemos
        recurrir a una interpretación geométrica. Dicha interpretación
        no describe literalmente las componentes de cada vector. Es
        tan solo un ESQUEMA geométrico.""") as tracker:
            self.play(Indicate(grp1[0][0][2:6]),
                      run_time=tracker.duration)

        with self.voiceover(text=r"""En dicho esquema, los vectores
        son puntos de un espacio ene-dimensional. Como en los casos
        anteriores, se suman componente a componente, es decir, se
        suman las coordenadas respecto a cada eje de manera separada,
        y su suma es conmutativa.""") as tracker:
            self.play(Indicate(grp1[0][0][6:8]),
                      run_time=tracker.duration)

        with self.voiceover(text=r"""Por tanto, como esquema
        geométrico, la regla del paralelogramo es válida incluso en
        espacios de dimension arbitraria. Lo es incluso en dimensión
        infinita.""") as tracker:
            self.play(FadeIn(grp1[1:5]),
                      run_time=tracker.duration/2)
            self.play(FadeIn(grp1[5:]),
                      run_time=tracker.duration/2)
```

2. Escena 5 (Visión 3D)

```
class L01_V02_E05_SumaEnRn_3D(ThreeDScene):
    <<Créditos en distintas partes de la pantalla>>

    def construct(self):
        #self.set_speech_service( AzureService(voice="es-ES-AlvaroNeural" ) )
        #self.set_speech_service(GTTSService(lang="es", tld="com"))
        myTemplate = TexTemplate()
        myTemplate.add_to_preamble(r"""\usepackage{nacal} """)

        #self.creditos(17)

        #plane = NumberPlane(background_line_style={"stroke_opacity": 0.1})

        axes = ThreeDAxes()

        b     = VectorR3([2,2,3], color=PURE_RED)
        c     = VectorR3([-3,1,-1], color=PURE_GREEN)
        d     = VectorR3([-1,3,2], color=PURE_BLUE)

        b_dot = b.dot(axes)
        c_dot = c.dot(axes)
        d_dot = d.dot(axes)

        flechab = b.arrow(axes)
        flechac = c.arrow(axes)
        flechad = d.arrow(axes)

        linebd = Line3D(start=np.array(axes.c2p(*b.coords)), end=np.array(axes.c2p(*d.coords)))
        linecd = Line3D(start=np.array(axes.c2p(*c.coords)), end=np.array(axes.c2p(*d.coords)))

        #self.add(axes, plane)

        self.move_camera(phi=75 * DEGREES, theta=60 * DEGREES, zoom=1, run_time=1)
        self.add(b_dot,
                 c_dot)
        self.begin_ambient_camera_rotation(rate=0.2)
        self.wait(15)

        self.play(FadeIn(flechab,
                         flechac),
                  FadeOut(b_dot,
                          c_dot))
        self.wait(23)

        self.add(linebd,
                 linecd)
        self.play(FadeIn(flechad))

        self.wait(3)

        self.begin_ambient_camera_rotation(rate=0.6, about='gamma')
        self.wait(5)

        self.begin_ambient_camera_rotation(rate=0.6, about='theta')
        self.play(FadeIn(b_dot,
                         c_dot,
                         d_dot),
                  FadeOut(flechab,
                          flechac,
                          flechad,
                          linebd,
                          linecd))
        self.wait(5)
```

- Fusión audio y vídeo poca calidad

```
rm -f L01_V02_E05_SumaEnRn.mp4
ffmpeg -i L01_V02_E05_SumaEnRn_3D.mp4 -i L01_V02_E05_SumaEnRn_voz.mp4 -c:v copy -c:a aac -strict experimental L01_V02_E05
#mv L01_V02_E05_SumaEnRn_voz.srt L01_V02_E05_SumaEnRn.srt
mkdir -p aux_movie_files
```

```
mv L01_V02_E05_SumaEnRn_3D.mp4 aux_movie_files/
mv L01_V02_E05_SumaEnRn_voz.mp4 aux_movie_files/
```

- Fusión audio y vídeo poca calidad HD1080

```
rm -f L01_V02_E05_SumaEnRn.mp4
ffmpeg -i L01_V02_E05_SumaEnRn_3D.mp4 -i L01_V02_E05_SumaEnRn_voz.mp4 -c:v copy -c:a aac -strict experimental L01_V02_E05_
mkdir -p aux_movie_files
cp L01_V02_E05_SumaEnRn_voz.srt L01_V02_E05_SumaEnRn.srt
mv L01_V02_E05_SumaEnRn_3D.mp4 aux_movie_files/
mv L01_V02_E05_SumaEnRn_voz.mp4 aux_movie_files/
```

1.1.6. Escena 6 - Resumen

```
class L01_V02_E06_Resumen(MiEscenaConVoz):
    def construct(self):
        self.set_speech_service( AzureService(voice="es-ES-AlvaroNeural" ) )
        #self.set_speech_service(GTTSService(lang="es", tld="com"))
        myTemplate = TexTemplate()
        myTemplate.add_to_preamble(r"""\usepackage{nacal} """)

        self.creditos()

        titulo = Title(r"Suma de vectores de \R[n]",
                     tex_template = myTemplate,
                     font_size=70).set_color(BLUE)
        self.play(Write(titulo))
        self.pausa()

        # Resumen
        resumen = Tex(r"\textbf{Lo más importante:}",
                 tex_template = myTemplate,
                 font_size=50).set_color(ORANGE).next_to(titulo, DOWN, aligned_edge=LEFT)


        with self.voiceover(text=r"""Por último, quiero subrayar que
        la interpretación geométrica se deriva de la definición de la
        suma.""") as tracker:
            self.add(resumen)
            self.pausa(tracker.duration)

        cvab  = MathTex(r"\elemRp{\Vect{a}+\Vect{b}}{i}", tex_template = myTemplate)
        cva   = MathTex(r"\eleVR{a}{i}", tex_template = myTemplate)
        cvb   = MathTex(r"\eleVR{b}{i}", tex_template = myTemplate)
        igual = MathTex(r"=",              tex_template = myTemplate,)
        mas   = MathTex(r"+",              tex_template = myTemplate,)
        eq_suma = VGroup(cvab,igual,cva,mas,cvb).arrange(RIGHT).scale(1.5)
        cva_copy   = cva.copy().move_to(cvb)
        cvb_copy   = cvb.copy().move_to(cva)

        item1 = MathTex(r"\Vect{a}+\Vect{b} = \Vect{b}+\Vect{a}",tex_template = myTemplate).next_to(eq_suma, DOWN, buff=1.5).sca

        props_suma = VGroup(eq_suma, item1)

        with self.voiceover(text=r"""Por tanto, lo más importante es
        destacar que la definición indica que la suma es una operación
        componente a componente""") as tracker:
            self.play(FadeIn(props_suma[0]),
                      run_time=tracker.duration+0.3)

        with self.voiceover(text=r"""Ello se traduce en una regla de
        cálculo simbólico. Dicha regla nos dice que el operador
        selector es distributivo respecto de la suma.""") as tracker:
            self.pausa(tracker.duration/2)
            self.play(Indicate(eq_suma[0][0][0]),
                      Indicate(eq_suma[0][0][-3:]),
                      Indicate(eq_suma[2][0][-2:]),
                      Indicate(eq_suma[4][0][-2:]),
                      run_time = tracker.duration/2)
            self.pausa(0.3)

        with self.voiceover(text=r"""Además, como las componentes son
        números reales, también hay que destacar que la suma es
        conmutativa""") as tracker:
            self.play(Transform(cva,cva_copy),
                      Transform(cvb,cvb_copy),
                      run_time = 3*tracker.duration/4)
            self.play(Indicate(item1),
                      run_time = 3*tracker.duration/10)

        self.pausa_larga()

```

2. Trozos comunes de código

2.1. Carga de la librería Manim y NacAL

```
from manim import *
from manim_voiceover import VoiceoverScene
from manim_voiceover.services.gtts import GTTSService
import nacal as nc
import sympy as sp

# PARA LA TRADUCCIÓN (pero no me ha funcionado)

#from manim_voiceover.translate import get_gettext
# # It is good practice to get the LOCALE and DOMAIN from environment variables
#import os
#LOCALE = os.getenv("LOCALE")
#DOMAIN = os.getenv("DOMAIN")
# The following function uses LOCALE and DOMAIN to set the language, and
# returns a gettext function that is used to insert translations.
#_ = get_gettext()
```

```
from manim import *
from manim_voiceover import VoiceoverScene
from manim_voiceover.services.azure import AzureService
import nacal as nc
import sympy as sp

# PARA LA TRADUCCIÓN (pero no me ha funcionado)

#from manim_voiceover.translate import get_gettext
# # It is good practice to get the LOCALE and DOMAIN from environment variables
#import os
#LOCALE = os.getenv("LOCALE")
#DOMAIN = os.getenv("DOMAIN")
# The following function uses LOCALE and DOMAIN to set the language, and
# returns a gettext function that is used to insert translations.
#_ = get_gettext()
```

2.2. Creditos

```
copyright = Tex(r"\textcopyright{\;} 2024\; Marcos Bujosa  ")
CGG  = VGroup(copyright).rotate(PI/2).scale(0.5).to_edge(RIGHT, buff=0.1).set_color(GRAY_D)
self.add(CGG)
```

```
class ZCreditos(Scene):
    def construct(self):
        copyright = Tex(r"\textcopyright{\;} 2024 \; Marcos Bujosa")
        github = Tex(r"\texttt{https://github.com/mbujosab}").next_to(copyright, DOWN)
        CGG  = VGroup(copyright,github).scale(1.1)
        self.add(CGG)
        self.wait(10)
```

3. Rodando: 1,2,3. . . ¡acción!

1. Generamos un fichero `mpeg` por cada escena

    - Versión de poca calidad

```
echo $escena | manim -pql $fichero.py --disable_caching
```

    - Versión calidad HD1080

```
echo $escena | manim -qh $fichero.py --disable_caching
```

2. Concatenamos las escenas en un único fichero `mpeg` y añadimos música de fondo.

    - Montando la versión de baja resolución

```
ln -s -f "$(pwd)/$subdir/ZCreditos/$calidad/ZCreditos.mp4" "$(pwd)/$subdir/$video/$calidad/ZCreditos.mp4"
rm -f $subdir/$video/$calidad/$video.mp4 list.txt
for f in $subdir/$video/$calidad/*.mp4 ; do echo file \'$f\' >> list.txt; done && ffmpeg -f concat -safe 0 -i list.txt -c

mkdir -p tmp

ffmpeg -i $subdir/$video/$calidad/$video.mp4 -i $music.mp3 -filter_complex "[0:a]apad[main]; [1:a]volume=0.04,apad[A]; [m
```

    - Montando la versión de resolución HD1080

```
ln -s -f "$(pwd)/$subdir/ZCreditos/$calidad/ZCreditos.mp4" "$(pwd)/$subdir/$video/$calidad/ZCreditos.mp4"
rm -f $subdir/$video/$calidad/$video.mp4 list.txt
for f in $subdir/$video/$calidad/*.mp4 ; do echo file \'$f\' >> list.txt; done && ffmpeg -f concat -safe 0 -i list.txt -c

mkdir -p tmp

ffmpeg -i $subdir/$video/$calidad/$video.mp4 -i $music.mp3 -filter_complex "[0:a]apad[main]; [1:a]volume=0.04,apad[A]; [m
```

3. Fundimos a negro los últimos segundos del vídeo (y la música).

```
dur=$(ffprobe -loglevel error -show_entries format=duration -of default=nk=1:nw=1 "tmp/$video.mp4") && offset=$(bc -l <<<
```

4. Copiamos el resultado a un lugar público

```
cp -f $video.mp4 $subdir/$video.mp4
```