O USO DO MOTOR GRÁFICO MANIM PARA ELABORAÇÃO DE ANIMAÇÕES MATEMÁTICAS AUTOEXPLICATIVAS

JOÃO ANTONIO NEVES SOARES[1]; JANICE NERY[2]

[1]*Universidade Federal de Pelotas – jansoares.samsung@gmail.com*

[2]*Universidade Federal de Pelotas – nery.janice@ufpel.edu.br*

1. INTRODUÇÃO

As disciplinas de matemática do ensino superior podem ser percebidas como as mais desafiadoras nos cursos de graduação das engenharias e ciências exatas. Uma das dificuldades observadas em trabalhos anteriores está na capacidade dos alunos de desenvolverem o pensamento matemático necessário para deduzirem conceitos fundamentais de Cálculo e serem capazes de aplicá-los a problemas de suas respectivas áreas, como exemplifica NACHTIGALL, et al.(2021).

Neste contexto, o uso de animações gráficas com o objetivo de ensinar tópicos matemáticos podem ajudar no aprendizado guiado de conceitos essenciais necessários para desenvolver habilidades como pensamento abstrato, capacidade de síntese de informações e ideias, formalizar conceitos através de exemplos, estimar padrões lógicos e entre outros a capacidade de resolver problemas.

Em Byrne et al.(1999) afirma-se que o uso de animações de conceitos é válido como ajuda educacional para a dedução cognitiva sobre os passos envolvidos na formulação de problemas e teoremas assim como facilita a criação de um mapa mental interconectado com os conhecimentos já adquiridos pelo aluno durante sua aprendizagem.

Deste modo este trabalho visa apresentar e discutir o uso da ferramenta gráfica MANIM, Mathematical Animation Engine (Motor de animações matemáticas), com o intuito de familiarizar os educadores e professores com o uso desta ferramenta como auxílio no ensino de disciplinas de matemática no ensino superior.

2. METODOLOGIA

Na presente seção propomos examinar algumas das principais funcionalidades do motor gráfico de animações, MANIM, bem como o processo prático que envolve os passos de criação de uma animação utilizando os recursos desta ferramenta.

Como COLUCI, V.R. explica, o Manim é uma biblioteca gratuita de funções do Python, para criar animações de forma precisa por meio de programação computacional, criada por Grant Sanderson. Atualmente possui uma versão da comunidade mantida por mais de 200 desenvolvedores que contribuem com a documentação, tutoriais e correções de erros, além de atualizações.

O processo de instalação e configuração em computadores pessoais está disponível na documentação escrita por The Manim Community Developers. (2022) e, portanto, não entrará no escopo deste trabalho.

Os conceitos envolvidos na criação das animações do Manim são: cenas, transições, câmera e em especial o objeto matemático (Mobject) que, de forma geral, é a estrutura mais importante de uma animação uma vez que são os objetos gráficos que serão renderizados na cena. Scene é a principal classe da biblioteca Manim e ela pode ser vista como a tela (canvas) onde as animações tomam forma.

O seu papel principal é fornecer recursos para manipular Mobjects e animações. Na prática, a maioria dos scripts de animações criados com o Manim conterá uma classe derivada de Scene. O método construct() é o mais importante de Scene pois conterá as descricoes programáticas das animacoes do usuário. A partir deste método é possível criar formas geométricas como retângulos e círculos, fórmulas matemáticas usando a linguagem LATEX e elementos gráficos 2D e 3D como pontos, retas e planos através do uso de métodos e atributos fornecidos pela biblioteca. Um exemplo é o uso dos métodos add() e remove() para exibir e remover um objeto da tela respectivamente. Já a animação de um ou mais objetos da cena se dá através do comando play().

3. RESULTADOS E DISCUSSÃO

Nesta seção iremos entender como as estruturas apresentadas anteriormente se interligam através de um exemplo de animacao matemática utilizando o script abaixo.

```
from manim import *
class RiemannRectangles(Scene):
    def construct(self):
        eixo_xy = Axes(y_range=[-1,10,1], x_range=[-1,10,1], tips=False,
axis_config={"include_numbers":True})
        #Rótulos para o eixo vertical e horizontal
        y_rotulo = eixo_xy.get_y_axis_label("y")
        x_rotulo = eixo_xy.get_x_axis_label("x")
        #Gráfico da funcao f(x)
        fx = eixo_xy.plot(lambda x: 0.1 * (x + 3-5) * (x - 3-5) * (x-5) +
5,x_range=[0.3,9.2], use_smoothing=False, color=BLUE)
        ln_label = eixo_xy.get_graph_label(graph=fx,label="y = f(x)")
        #Retangulos de Riemann
        rect =
eixo_xy.get_riemann_rectangles(graph=fx,x_range=[2,8],color=[PURPLE,RED,OR
ANGE],dx=0.5)
        rect_peq =
eixo_xy.get_riemann_rectangles(graph=fx,x_range=[2,8],color=[PURPLE,RED,OR
ANGE],dx=0.25)
        #Inclui o sinal gráfico de chaves para informar o tamanho de um retangulo
        bc_rect = Brace(rect[2],sharpness=1,direction=[UP,RIGHT])
        bc_label = bc_rect.get_text("dx")
        bc2_rect = Brace(rect_peq[5],sharpness=1,direction=[UP,RIGHT])
        self.add(eixo_xy,fx,y_rotulo,x_rotulo,rect,bc_rect,bc_label,ln_label)
        self.wait()

self.play(ReplacementTransform(mobject=rect,target_mobject=rect_peq),Replace
mentTransform(mobject=bc_rect,target_mobject=bc2_rect))
```

No código do exemplo acima começamos por construir o plano cartesiano com eixos horizontal e vertical com 11 marcações de tamanho 1 de escala decimal, partindo de -1 até 10. Adicionamos os rótulos "x" e "y" para os eixos horizontal e vertical respectivamente. A partir disto o próximo passo foi construir o gráfico da função y=f(x). O método responsável por plotar um gráfico de uma função de variável real no Manim é plot(). Como argumento obrigatório passamos uma expressão polinomial através do recurso da linguagem python chamado de expressão lambda. Em seguida, para facilitar a visualização adicionamos a cor azul a linha e um rótulo texto "y = f(x)".

Um dos conceitos essenciais no Cálculo integral é a Soma de Riemann para aproximar a "área" sob uma curva contínua "y=f(x)" a partir da área de n retângulos de mesma largura dx. Manim fornece um método simples e preciso para criar animações de aproximação de uma área através de retângulos. Em nossos exemplos para criar os retângulos sobre um intervalo determinado sob a curva "f(x)" utilizamos o método get_riemann_rectangles() e colorimos os retangulos através de um gradiente de roxo, vermelho e laranja para deixar a animação mais atrativa visualmente. Outro recurso interessante é utilizar sinais matemáticos para delimitar representações de comprimento. No exemplo utilizamos chaves para evidenciar o comprimento, dx, de um retângulo subscrito no gráfico. O comando .add() é responsável por adicionar todos os Mobjects criados anteriormente no script e chamar internamente o render para exibir na tela os gráficos, já o comando .play() é encarregado de realizar a animação entre os objetos especificados pelo usuário. Em nosso exemplo utilizamos o método replacementTransform em conjunto com .play(), para criar uma transição entre os retângulos de tamanho dx = 0.5 para os de tamanho 0.25. Ao final utilizamos a expressão de linha de comando: manim -pqh riemann.py RiemannRectangles para rodar o script para renderizar um vídeo contendo a animacao matemática em formato de arquivo .mp4. Nas Figuras(1) e (2) abaixo está a captura de 2 frames distintos da animação.

Figura (1) - Frame 1

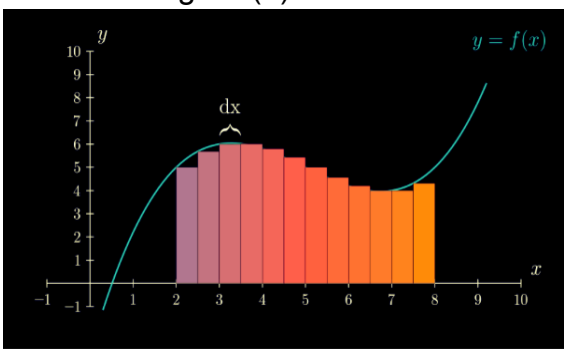
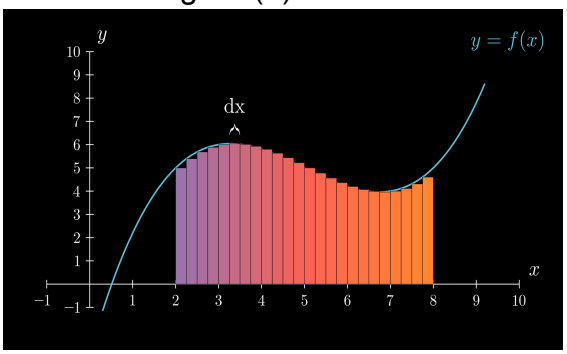


Figura (2) - Frame 2



4. CONCLUSÕES

Concluímos que o motor gráfico de animações matemáticas podem ajudar os professores e educadores a produzirem animações matemáticas auto-explicativas, interativas e engajadoras para os alunos, de forma a fomentar o aprendizado do estudante e a capacidade de desenvolver a intuição lógica que acelere o aprendizado a respeito de um certo tópico da matemática, visto que os vídeos produzidos com essa ferramenta podem ser facilmente integrados como material de apoio em ambientes virtuais de aprendizagem (Moodle, por exemplo). As vantagens de se utilizar o Manim para esse propósito é dispor de uma ferramenta de software livre e código aberto, que possui comandos concisos e uma comunidade abrangente de desenvolvedores e criadores contribuindo com uma boa documentação assim

como uma ampla gama de exemplos e tutoriais. Um contraponto é o requisito de possuir conhecimento básico em programação, já que isso pode ser uma barreira para algumas pessoas, felizmente essa habilidade pode ser facilmente aprendida através de tutoriais e livros em questão de algumas semanas.

Em síntese, acreditamos que este trabalho vai impulsionar novos usuários a se interessarem e a desenvolverem animações robustas capazes de contribuir com a propagação de ideias assim como facilitar o processo de ensino em diversos contextos da matemática.

5. REFERÊNCIAS BIBLIOGRÁFICAS

NACHTIGALL, C.; CAMPELO, H. D.; DA SILVA, P. T.; PERGHER, R. **The use of digital technologies in calculus teaching at UFPel: An analysis of GAMA project initiatives.** Ciência e Natura, *[S. l.]*, v. 43, p. e33, 2021. DOI: 10.5902/2179460X40950. Disponível em: https://periodicos.ufsm.br/cienciaenatura/article/view/40950. Acesso em: 22 jul. 2022.

BYRNE, M., CATRAMBONE, R. & STASKO, T. (1999) **Evaluating animations as student aids in learning computer algorithms, Computers and Education**, 33, 253–278.

COLUCI, VITOR R. **Animações de conceitos da teoria de erros usando Manim/Python.** Revista Brasileira de Ensino de Física, v. 44, 2021.

THE MANIM COMUNITY DEVELOPERS. (2022). **Manim – Mathematical Animation Framework** (Version v0.16.0) [Computer software]. Acesso em: 22 jul. 2022. Disponível em: https://www.manim.community/

3BLUE1BROWN, **Playlist: Essense of Calculus.** Acesso em: 22 jul 2022. Disponível em: https://youtube.com/playlist?list=PLZHQObOWTQDMsr9K-rj53DwVRMYO3t5Yr

M. TAYLOR , D. POUNTNEY & I. MALABAR (2007) **Animation as an aid for the teaching of mathematical concepts**, Journal of Further and Higher Education, 31:3, 249-261, DOI: 10.1080/03098770701424975