UNIVERSIDADE ESTADUAL DE MARINGÁ

TRABALHO DE PROCESSAMENTO DIGITAL DE IMAGENS

MATHEUS AUGUSTO SCHIAVON PARISE

RA: 107115

MARINGÁ, 2022

1. INTRODUÇÃO

O intuito desse trabalho é desenvolver um programa em python 3.8 que seja capaz de receber uma imagem “a” colorida no formato RGB, um valor inteiro “x”, um valor inteiro de matiz “m" que esteja dentro do intervalo 0 <= m < 360 (ou seja, de 0 até 359) e a partir disso converter a imagem a para o sistema HSV. Em seguida, manipular a banda H, de matiz substituir todas as matizes no intervalo [m - x, m + x] por suas matizes inversas. Isto é, se a matiz q está no intervalo, você deve substituí-la pela matriz q + 180.

2. FUNCIONAMENTO DO PROGRAMA

O programa criado para esse trabalho é uma pequena interface que possui as entradas para informações solicitadas. Primeiramente, as entradas M e X, **elas devem receber números inteiros**, caso um número float seja digitado (lembrando que a parte fracionada é delimitado por um ponto, exemplo “10.1” ao invés de “10,1”) o programa irá arredondar para baixo, se alguma entrada inválida seja digitada o programa irá gerar um erro. Se a altura máxima da imagem original exceder 512px o programa irá alterar o tamanho da imagem exibida para ter no máximo 512px para facilitar a visualização, o programa **NÃO ALTERA** a imagem original apenas faz com que ela seja exibida de forma reduzida, ele ainda irá fazer as transformações e salvar a imagem com tamanho original, o mesmo vale para a largura mas esta foi limitada à 864 px .

Em seguida a parte dos arquivos da imagem, não há necessidade de digitar o caminho manualmente basta clicar no botão Buscar e selecionar a imagem (ela não precisa estar em tons de cinza o programa faz isso para você), se ela não aparecer basta mudar a extensão dos arquivos na busca de .jpg e jpeg para *(todos). Após selecionar a imagem clique no botão Carregar Imagem, ele irá fazer exatamente isso, após isso o botão Transformar imagem serve para aplicar as transformações considerando as variáveis T, Brilho e Acima na imagem carregada (ou seja, **é obrigatório carregar a imagem, é válido ressaltar que o arquivo precisa existir, estar com o nome correto e ser de fato uma imagem caso contrário o programa irá levantar erro**). Por fim o botão de salvar, ao ter uma imagem carregada e transformada se for de interesse, é possível salvar a imagem ao clicar no botão salvar, o programa irá salvá-la automaticamente na mesma pasta da imagem original com o

formato .png e o nome tendo a seguinte formatação: Nome original da imagem + "_M" +Valor de M + "_X" + Valor de X + ".png". A seguir uma imagem da interface do programa.

3. CÓDIGO

A seguir estão os prints relacionados ao código do programa descrito acima.

```python
import numpy as np
import PySimpleGUI as sg
import os, io
from PIL import Image
import math

file_types = [("All files (*.*)", "*.*"),
              ("PNG (*.png)", "*.png"),
              ("JPG (*.jpg)", "*.jpg"),
              ("JPEG (*.jpeg)", "*.jpeg")]

def tela():
    sg.ChangeLookAndFeel("DarkBlue14")
    # Layout
    layout = [  [sg.Text("Trabalho 2 de PROCESSAMENTO DIGITAL DE IMAGENS (RA107115)", font="Roboto")],
                [sg.Text("Insira as informações:", font="Roboto")],
                [
                    sg.Text("M (0 <= m < 360):", font="Roboto"), sg.Input(default_text="0", size=(10, 0), key="txtM"),
                    sg.Text("X:", font="Roboto"), sg.Input(default_text="0", size=(10, 0), key="txtX"),
                ],
                [
                    sg.Text("Arquivo de Imagem:",  font="Roboto"),
                    sg.Input(size=(25, 1), key="-FILE-"),
                    sg.FileBrowse(button_text="Buscar Imagem",file_types=file_types),
                    sg.Button("Carregar Imagem"),
                    sg.Button("Transformar Imagem"),
                    sg.Button("Salvar Imagem")
                ],
                [sg.Image(key="-IMAGE-")]
            ]
    # Janela
    janela = sg.Window('Alteração de faixa de Matizes em uma imagem no Sistema HSV', layout, icon=r'./ic.ico')
    imagem = None
```

```python
while True:
    # Extrair os dados da tela
    event, values = janela.Read()

    # Função para sair do programa
    if event == "Exit" or event == sg.WIN_CLOSED:
        break

    if event == "Carregar Imagem":
        filename = values["-FILE-"]
        if os.path.exists(filename):
            imagem = Image.open(values["-FILE-"]) # Objeto PIL Image
            height, width =  imagem.size

            # Limitando altura maxima
            width = (864 if width > 864 else width)
            height = (512 if height > 512 else height)
            imagem_resized = imagem.resize([width, height])

            # Salvando a imagem na memoria em binario
            bio = io.BytesIO()
            imagem_resized.save(bio, format="PNG")

            # Mostrando a imagem na janela
            janela["-IMAGE-"].update(data=bio.getvalue())
```

```
    if event == "Transformar Imagem":
        if imagem is not None:
            hsv = np.array(imagem.convert('HSV')) # Transforma a imagem HSV

            m = float(values["txtM"])
            x = float(values["txtX"])

            # Conversão inpropriada
            v_inf = int(math.floor(255 * (m - x) / 360))
            v_sup = int(math.floor(255 * (m + x) / 360))

            # Se Extourar 256
            v_inf = v_inf % 255
            v_sup = v_sup % 255

            # Se for Negativo trata
            v_sup = (v_sup+255 if v_sup < 0 else v_sup)
            v_inf = (v_inf+255 if v_inf < 0 else v_inf)

            a = hsv[..., 0]
            a_bool = ((v_inf <= a) & (a <= v_sup) if v_inf <= v_sup else (v_inf <= a) | (a <= v_sup))
            a[a_bool] += 128

            # Monstando o HSV a partir da matriz e convertendo para RGB
            nova_imagem = (Image.fromarray(hsv,'HSV')).convert('RGB')

            # Limitando altura maxima
            width, height = imagem.size
            width = (864 if width > 864 else width)
            height = (512 if height > 512 else height)
            nova_imagem_resized = nova_imagem.resize([width,height])

            # Salvando a imagem na memoria em binario
            bio = io.BytesIO()
            nova_imagem_resized.save(bio, format="PNG")

            # Mostrando a imagem na janela
            janela["-IMAGE-"].update(data=bio.getvalue())
```

```
    if event == "Salvar Imagem":
        if (imagem is not None) and (nova_imagem is not None):
            # Salvando a imagem no local da pasta da imagem original
            tam_formato = len(imagem.format)
            tam_nome = len(imagem.filename)
            tam_f = tam_nome - tam_formato - 1
            nome_final = imagem.filename[:tam_f] + "_M" + str(m) + "_X" + str(x) + ".jpg"
            nova_imagem.save(nome_final)

janela.close()

if __name__ == '__main__':
    tela()
```

1. IMAGENS

Foram selecionadas três imagens diferentes de preto, branco e transparente em consideração a sanidade mental do professor. As seguintes variações de configurações para cada uma das imagens:

Imagem 1: HSV HUE

- M = 0, X = 90
- M = 0, X = 180
- M = 180, X = 90

Imagem 2: Ferrari 296 gt3

- M = 240, X = 50
- M = 120, X = 80
- M = 0, X = 180

Imagem 3: James Webb

- M = 0, X = 180
- M = 30, X = 60
- M = 240, X = 60

Isso gera 3 configurações para 3 imagens dando 9 imagens, como irei mostrar as versões originais isso totalizará 12 imagens.

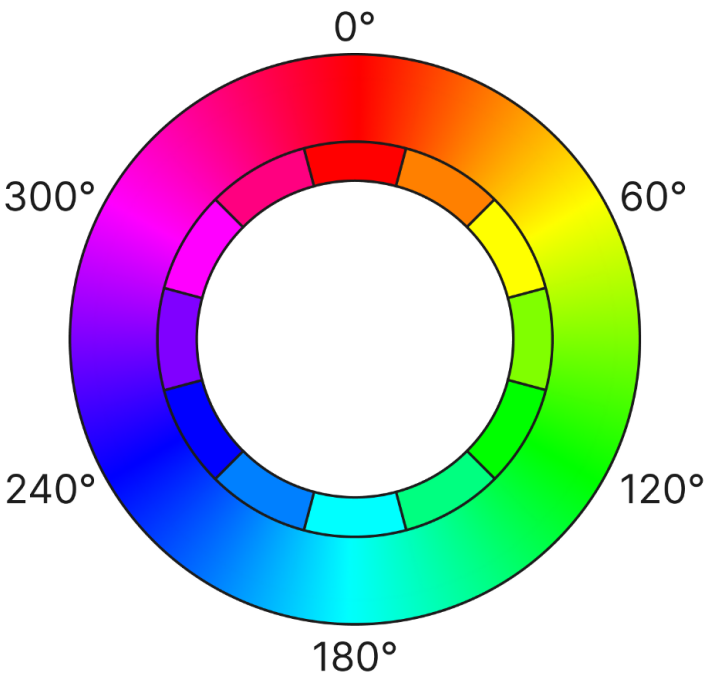

Imagem 1 - HSV HUE Original

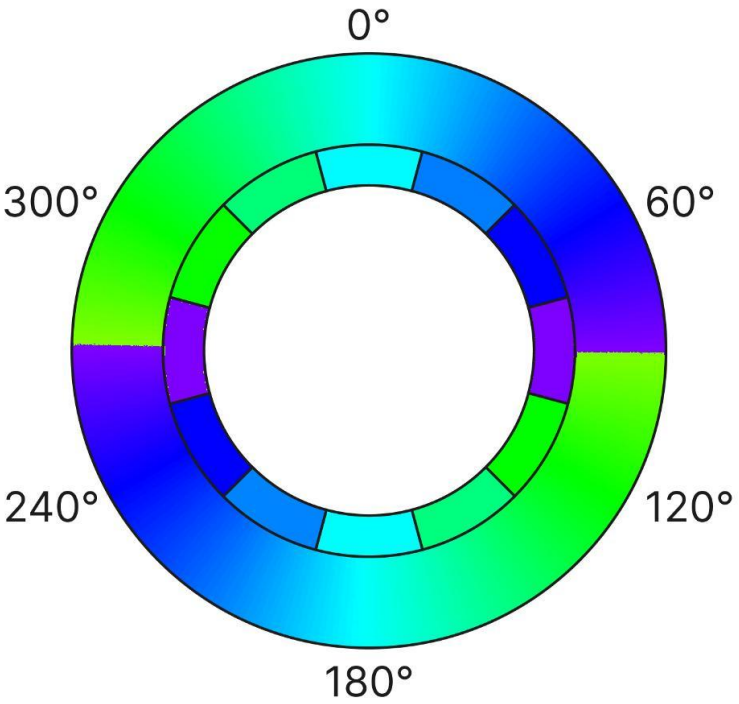

Imagem 2 - HSV HUE, M = 0 e X = 90

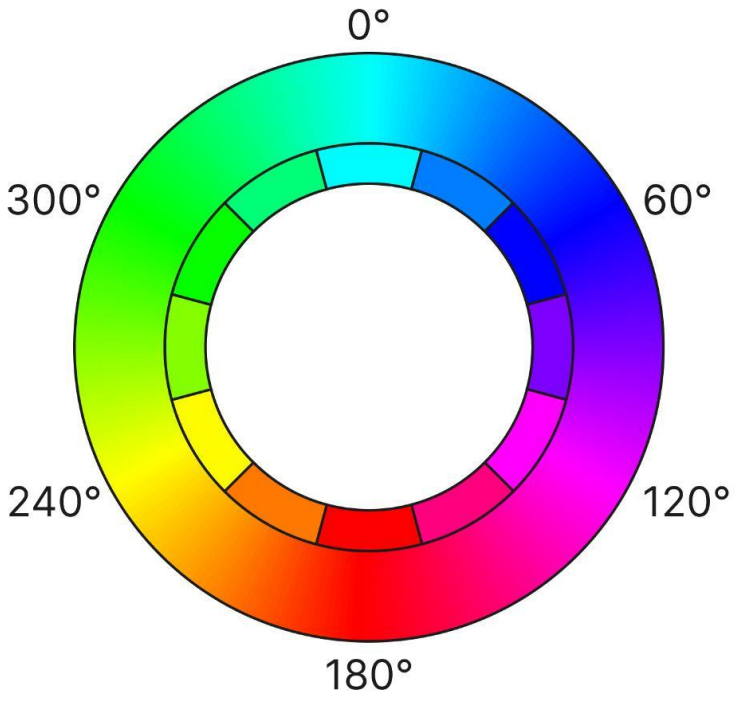

Imagem 3 - HSV HUE, M = 0 e X = 180

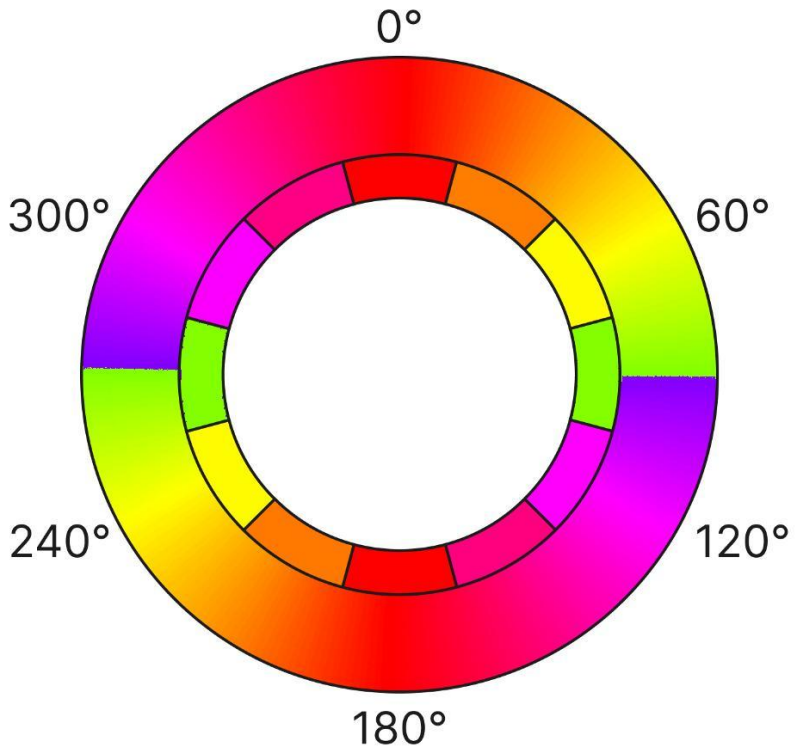

Imagem 4 - HSV HUE, M = 180 e X = 90

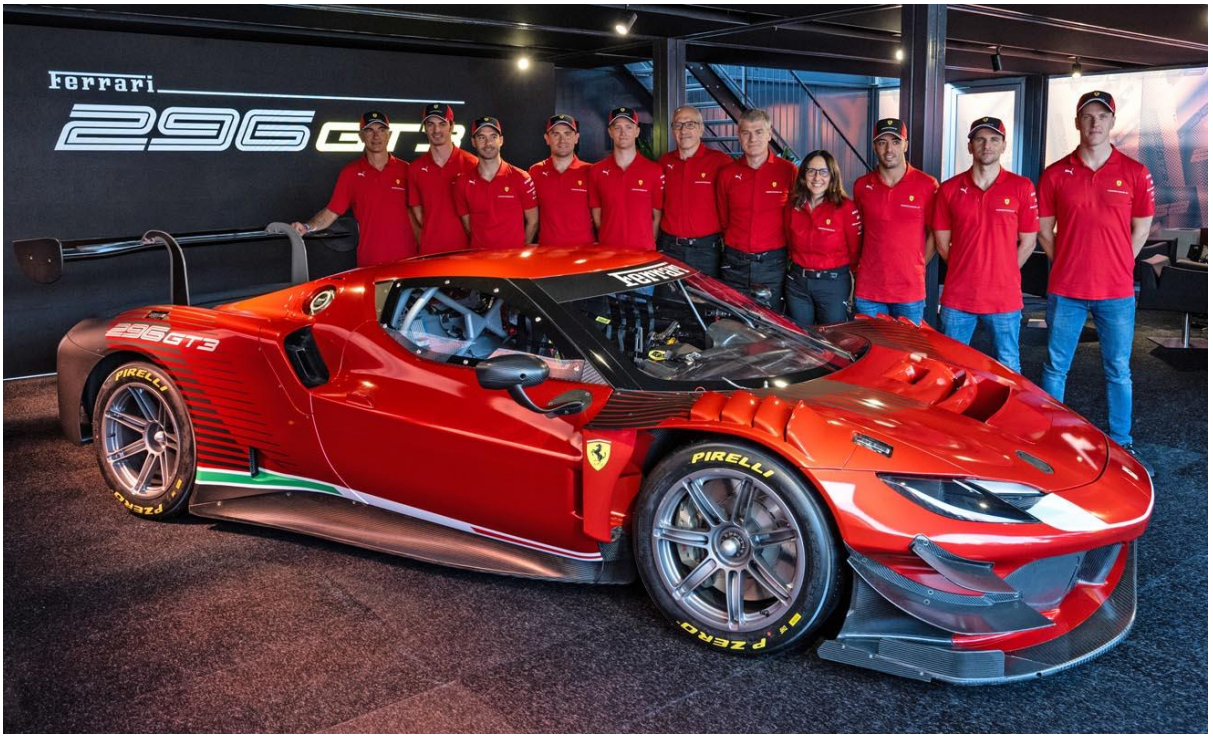

Imagem 5 - Ferrari 296 gt3 Original

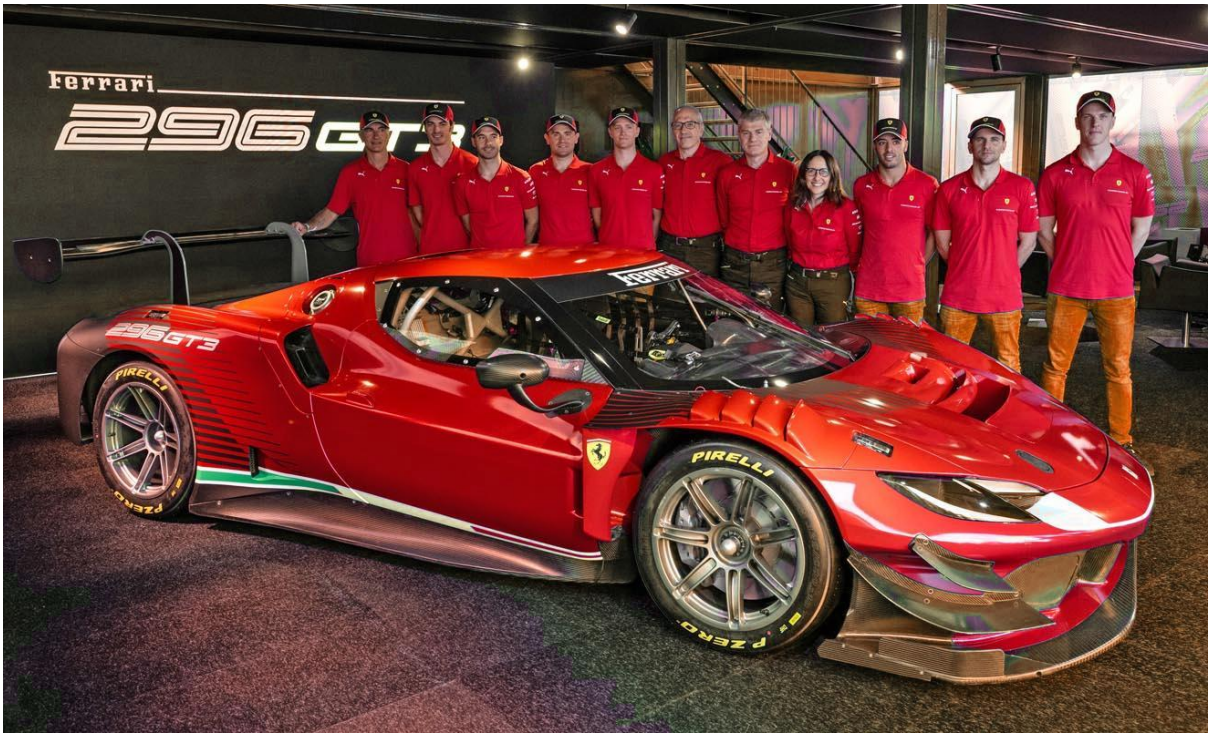

Imagem 6 - Ferrari 296 gt3, M = 240 e X = 50

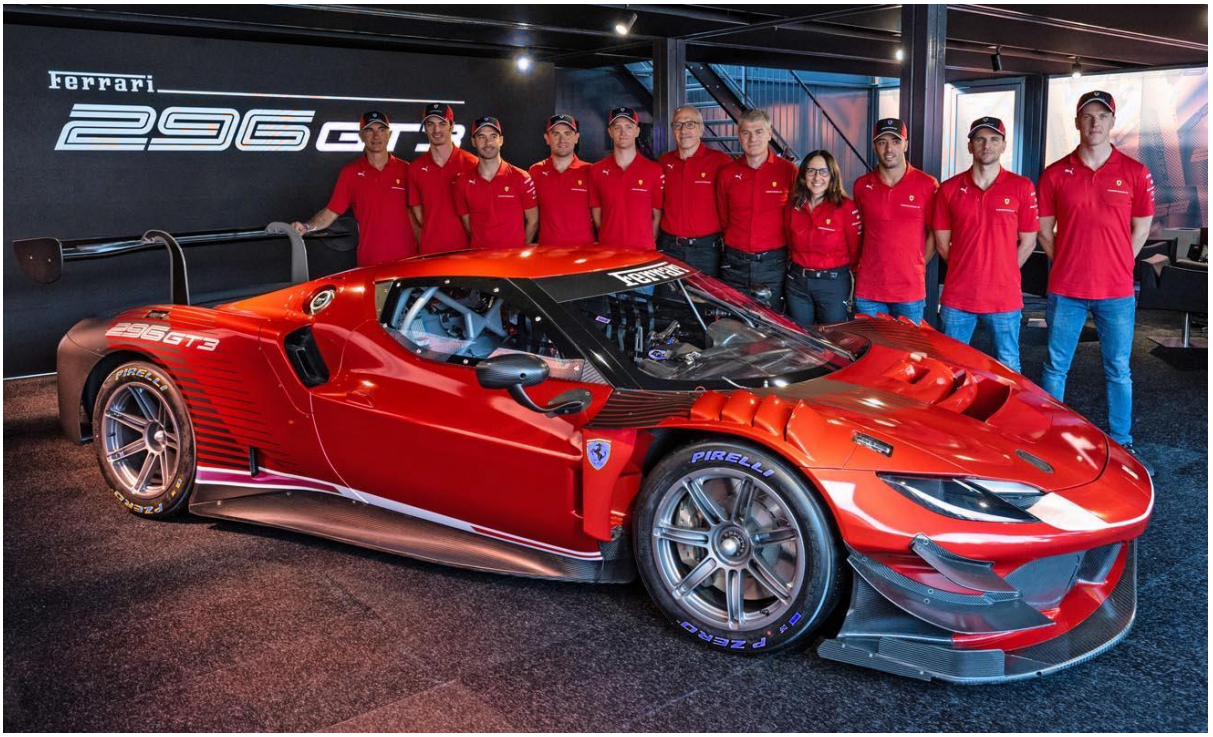

Imagem 7 - Ferrari 296 gt3, M = 120 e X = 80

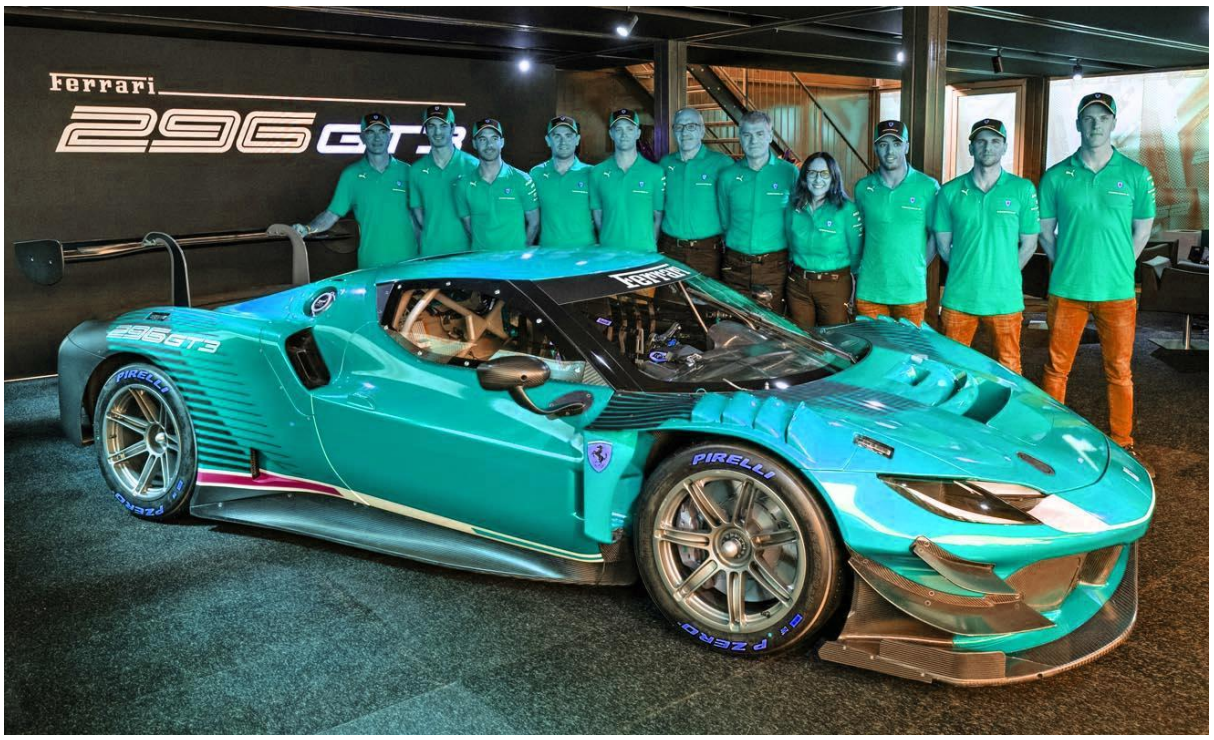

Imagem 8 - Ferrari 296 gt3, M = 0 e X = 180

Imagem 11 - Espaço Sideral James Webb Original

Imagem 12 - Espaço Sideral James Webb, M = 0 e X = 180

Imagem 13 - Espaço Sideral James Webb, M = 30 e X = 60

Imagem 14 -Espaço Sideral James Webb, M = 240 e X = 60

4. BIBLIOGRAFIA

Para o desenvolvimento deste trabalho foi necessário Python 3.8, com as bibliotecas *NUMPY* para manipular as matrizes das imagens, *PySimpleGUI* para criar a interface de usuário, *PILLOW* para abrir, manipular e salvar as imagens, *OS* e *IO* para interagir com os arquivos e seus formatos, *MATH* para arredondar para baixo e por fim *PYINSTALLER* para gerar o arquivo executável. As documentações do python 3.8 e dessas bibliotecas também foram utilizadas.