```
In [1]: #!pip install tensorflow_hub
```

```
In [2]: import tensorflow_hub as hub
        import tensorflow as tf
        from matplotlib import pyplot as plt
        import numpy as np
        import cv2
        import os #navigate path structure
        from PIL import Image
```

```
In [3]: model = hub.load('https://tfhub.dev/google/magenta/arbitrary-image-stylization-v1-
```

```
In [4]: def load_image(img_path):
            img = tf.io.read_file(img_path)
            img = tf.image.decode_image(img, channels=3)
            img = tf.image.convert_image_dtype(img, tf.float32)
            img = img[tf.newaxis, :]
            return img
```

Sample 1

```
In [5]: content_image = load_image('C:/Users/soumi/3.DMML_2/dataset/content/251 (3).jpg')
        style_image = load_image('C:/Users/soumi/3.DMML_2/dataset/art_work/Henri_Matisse_9
```

```
In [6]: content_image.shape

Out[6]: TensorShape([1, 533, 800, 3])
```

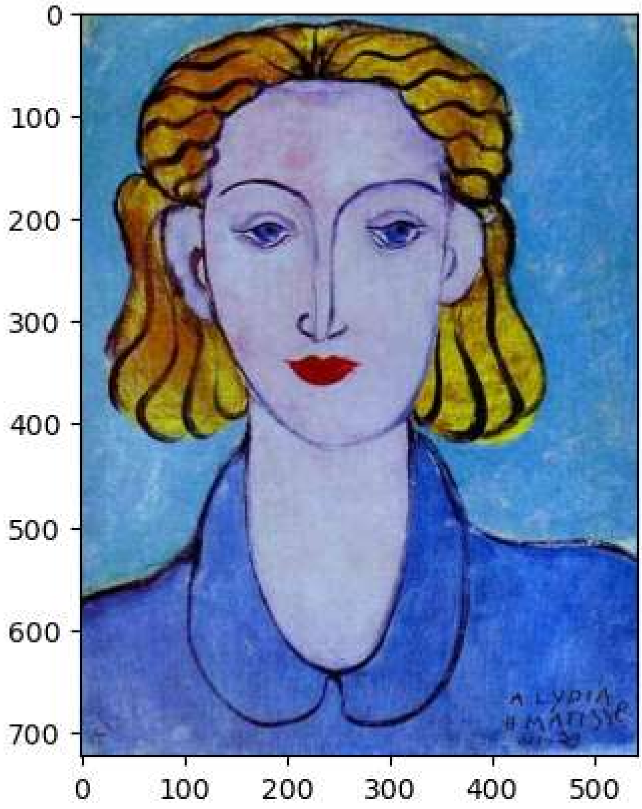
```
In [7]: style_image.shape

Out[7]: TensorShape([1, 723, 542, 3])
```

```
In [8]: plt.imshow(np.squeeze(content_image))
        plt.show()
```

In [9]:

```
plt.imshow(np.squeeze(style_image))
plt.show()
```

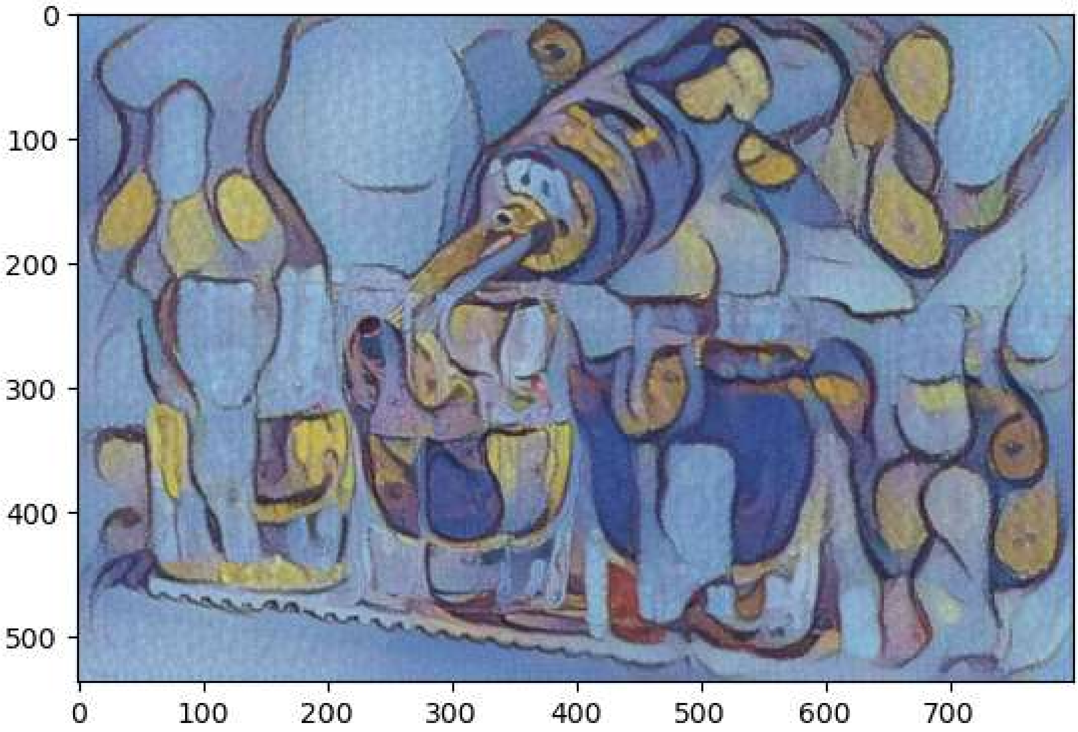


In [10]:

```
stylized_image = model(tf.constant(content_image), tf.constant(style_image))[0]
stylized_image1 = stylized_image
```

In [11]:

```
plt.imshow(np.squeeze(stylized_image))
plt.show()
```



In [12]:
```
cv2.imwrite('generated_img.jpg', cv2.cvtColor(np.squeeze(stylized_image)*255, cv2.(
```

Out[12]:
```
True
```

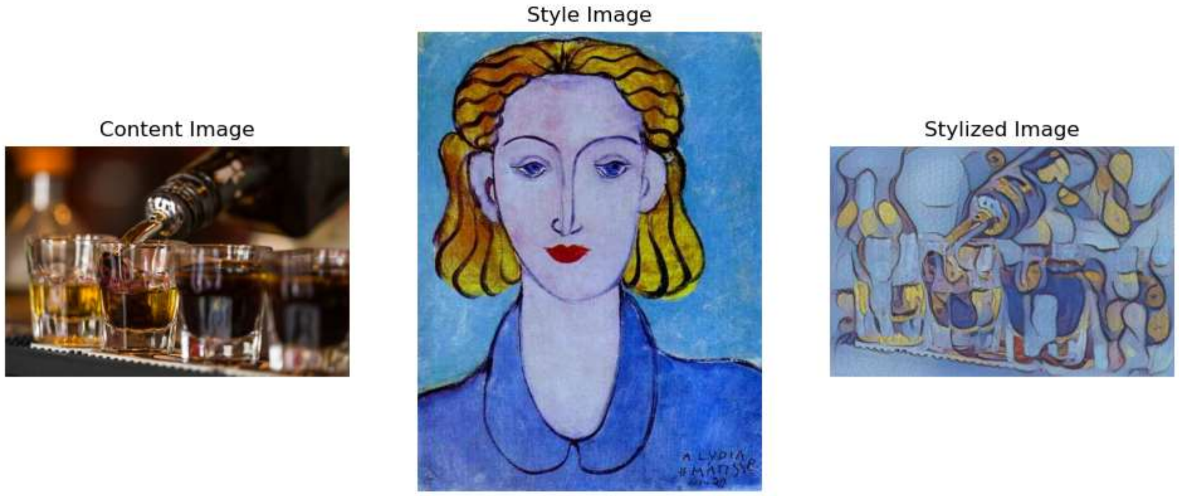
In [13]:
```
# Display the images using matplotlib
plt.figure(figsize=(12, 8))

plt.subplot(1, 3, 1)
plt.title('Content Image')
plt.imshow(np.squeeze(content_image))
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title('Style Image')
plt.imshow(np.squeeze(style_image))
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title('Stylized Image')
plt.imshow(np.squeeze(stylized_image))
plt.axis('off')

plt.show()
```



In [14]:

```
# Resize the style_image to match the shape of the content_image
style_image_resized = tf.image.resize(style_image, content_image.shape[1:3], metho

# Calculate Mean Squared Error
mse1 = tf.reduce_mean(tf.square(content_image - style_image_resized)).numpy()
print(f"Mean Squared Error: {mse1}")
```

```
Mean Squared Error: 0.1771637201309204
```

In [15]:

```
vgg = tf.keras.applications.VGG16(include_top=False, weights="imagenet")

def preprocess_for_vgg(image):
    return tf.keras.applications.vgg16.preprocess_input(image * 255)

content_image_vgg = preprocess_for_vgg(content_image)
stylized_image_vgg = preprocess_for_vgg(stylized_image)

content_features = vgg(content_image_vgg)
stylized_features = vgg(stylized_image_vgg)

perceptual_loss1 = tf.reduce_mean(tf.square(content_features - stylized_features))
print(f"Perceptual Loss: {perceptual_loss1}")
```

```
Perceptual Loss: 73.82260131835938
```

In [16]:

```
def gram_matrix(feature_maps):
    num_channels = feature_maps.shape[-1]
    flattened = tf.reshape(feature_maps, (-1, num_channels))
    gram = tf.matmul(flattened, flattened, transpose_a=True)
    gram /= tf.cast(tf.reduce_prod(feature_maps.shape), tf.float32)
    return gram

# Assuming you have already defined vgg and preprocess_for_vgg

style_image_vgg = preprocess_for_vgg(style_image)
stylized_image_vgg = preprocess_for_vgg(stylized_image)  # You might have missed t

style_features = vgg(style_image_vgg)
stylized_features = vgg(stylized_image_vgg)

style_gram = gram_matrix(style_features)
stylized_gram = gram_matrix(stylized_features)


gram_loss1 = tf.reduce_mean(tf.square(style_gram - stylized_gram)).numpy()
print(f"Gram Matrix Loss: {gram_loss1}")
```

```
Gram Matrix Loss: 0.0008529422921128571
```

In [17]:
```
def total_variation_loss(image):
    x_deltas = image[:, :, 1:, :] - image[:, :, :-1, :]
    y_deltas = image[:, 1:, :, :] - image[:, :-1, :, :]
    tv_loss = tf.reduce_sum(tf.abs(x_deltas)) + tf.reduce_sum(tf.abs(y_deltas))
    return tv_loss

# Calculate total variation loss for the stylized image
tv_loss1 = total_variation_loss(stylized_image)
print(f"Total Variation Loss: {tv_loss1.numpy()}")
```

```
Total Variation Loss: 67402.3125
```

Sample 2

In [18]:
```
content_image = load_image('C:/Users/soumi/3.DMML_2/dataset/content/391.jpg')
style_image = load_image('C:/Users/soumi/3.DMML_2/dataset/art_work/Claude_Monet_56
```

In [19]:
```
content_image.shape
```

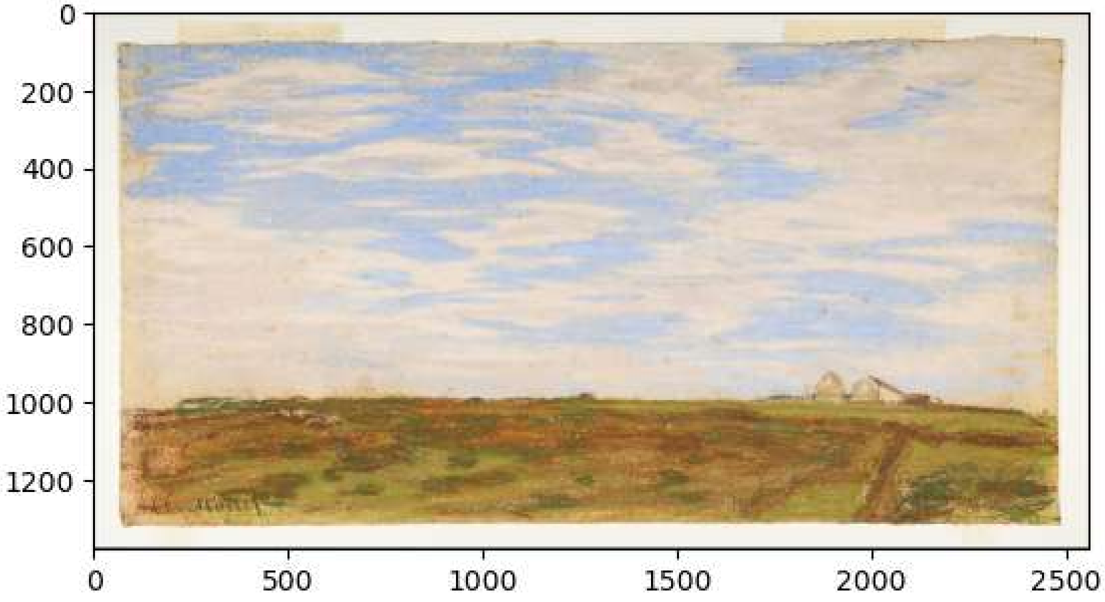
Out[19]:
```
TensorShape([1, 1367, 2050, 3])
```

In [20]:
```
style_image.shape
```

Out[20]:
```
TensorShape([1, 1379, 2560, 3])
```

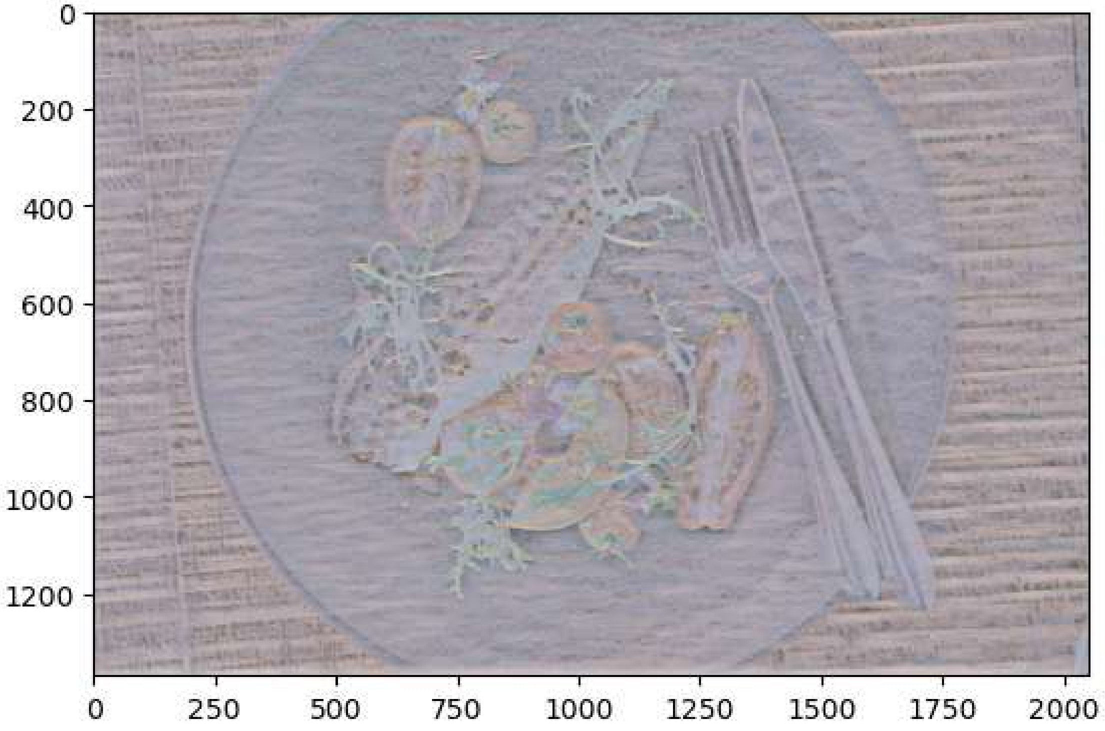
In [21]:
```
plt.imshow(np.squeeze(content_image))
plt.show()
```

In [22]:
```
plt.imshow(np.squeeze(style_image))
plt.show()
```



In [23]:
```
stylized_image = model(tf.constant(content_image), tf.constant(style_image))[0]
stylized_image2 = stylized_image
```

In [24]:
```
plt.imshow(np.squeeze(stylized_image))
plt.show()
```



In [25]:
```
cv2.imwrite('generated_img.jpg', cv2.cvtColor(np.squeeze(stylized_image)*255, cv2.(
```

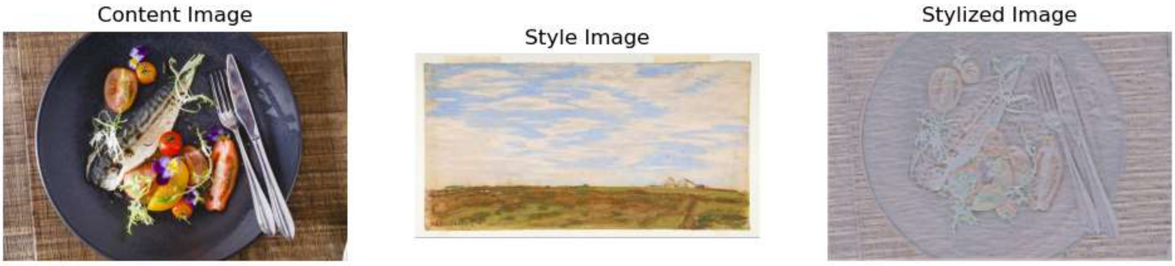
Out[25]:
```
True
```

In [26]:
```
# Display the images using matplotlib
plt.figure(figsize=(12, 8))

plt.subplot(1, 3, 1)
plt.title('Content Image')
plt.imshow(np.squeeze(content_image))
plt.axis('off')
plt.subplot(1, 3, 2)
plt.title('Style Image')
plt.imshow(np.squeeze(style_image))
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title('Stylized Image')
plt.imshow(np.squeeze(stylized_image))
plt.axis('off')

plt.show()
```



Content Image

Style Image

Stylized Image

In [27]:

```
# Resize the style_image to match the shape of the content_image
style_image_resized = tf.image.resize(style_image, content_image.shape[1:3], metho

# Calculate Mean Squared Error
mse2 = tf.reduce_mean(tf.square(content_image - style_image_resized)).numpy()
print(f"Mean Squared Error: {mse2}")
```

```
Mean Squared Error: 0.2287299931049347
```

In [28]:

```
content_image_vgg = preprocess_for_vgg(content_image)
stylized_image_vgg = preprocess_for_vgg(stylized_image)

content_features = vgg(content_image_vgg)
stylized_features = vgg(stylized_image_vgg)

perceptual_loss2 = tf.reduce_mean(tf.square(content_features - stylized_features))
print(f"Perceptual Loss: {perceptual_loss2}")
```

```
Perceptual Loss: 44.7812385559082
```

In [29]:

```
style_image_vgg = preprocess_for_vgg(style_image)
stylized_image_vgg = preprocess_for_vgg(stylized_image)  # You might have missed t

style_features = vgg(style_image_vgg)
stylized_features = vgg(stylized_image_vgg)

style_gram = gram_matrix(style_features)
stylized_gram = gram_matrix(stylized_features)


gram_loss2 = tf.reduce_mean(tf.square(style_gram - stylized_gram)).numpy()
print(f"Gram Matrix Loss: {gram_loss2}")
```

```
Gram Matrix Loss: 3.1323754228651524e-05
```

In [30]:

```
# Calculate total variation loss for the stylized image
tv_loss2 = total_variation_loss(stylized_image)
print(f"Total Variation Loss: {tv_loss2.numpy()}")
```

```
Total Variation Loss: 406243.75
```

In [ ]:

Sample 3

In [52]:

```
content_image = load_image('C:/Users/soumi/3.DMML_2/dataset/content/56.jpg')
style_image = load_image('C:/Users/soumi/3.DMML_2/dataset/art_work/Eugene_Delacroi
```

In [53]:

```
content_image.shape
```

Out[53]:

```
TensorShape([1, 1600, 2560, 3])
```

In [54]:

```
style_image.shape
```

Out[54]:

```
TensorShape([1, 600, 735, 3])
```

In [55]:

```
plt.imshow(np.squeeze(content_image))
plt.show()
```

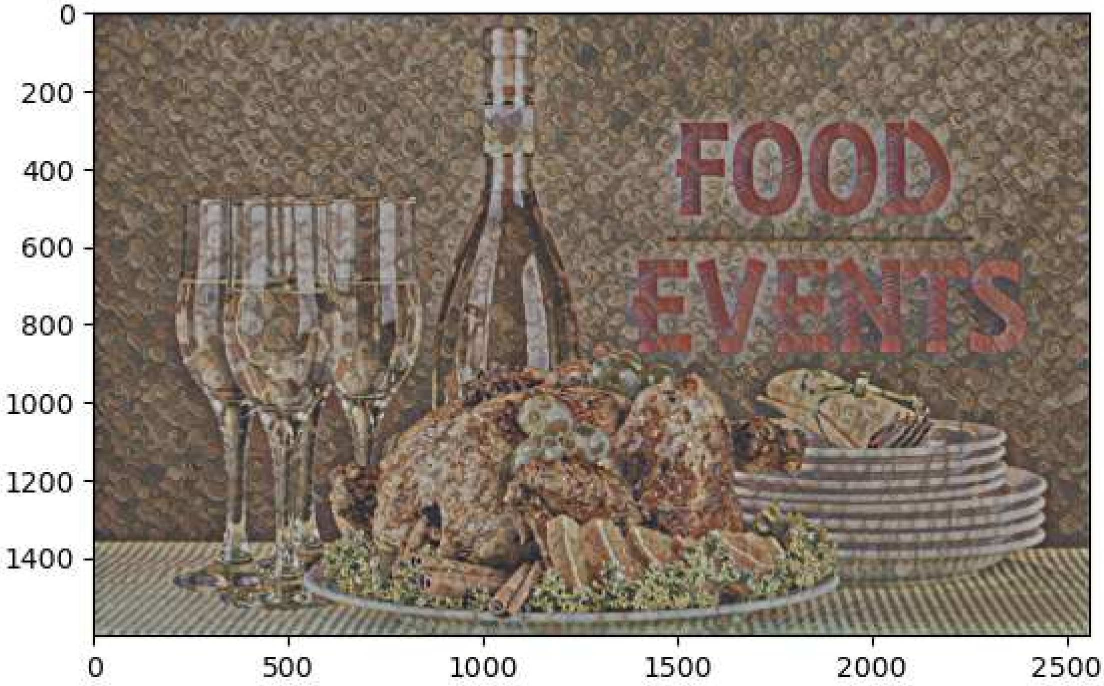
In [56]:

```
plt.imshow(np.squeeze(style_image))
plt.show()
```

0
100
200
300
400
500
0
100
200
300
400
500
600
700

In [57]:
```
stylized_image = model(tf.constant(content_image), tf.constant(style_image))[0]
stylized_image3 = stylized_image
```

In [58]:
```
plt.imshow(np.squeeze(stylized_image))
plt.show()
```



In [59]:
```
cv2.imwrite('generated_img.jpg', cv2.cvtColor(np.squeeze(stylized_image)*255, cv2.(
```

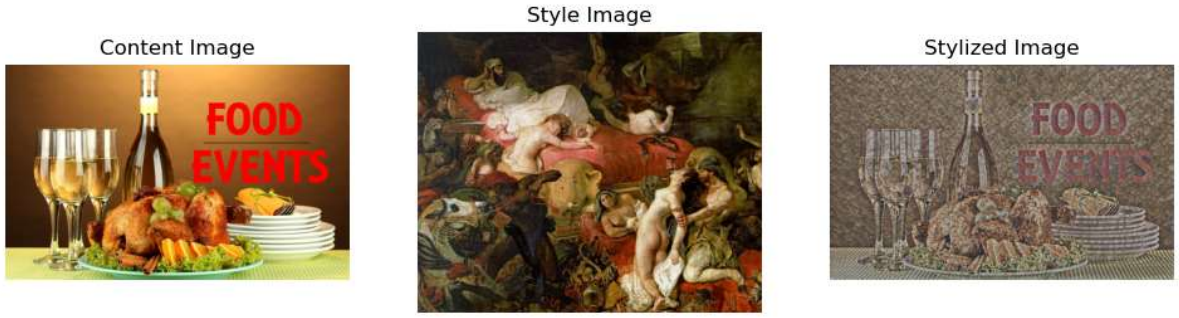
Out[59]:
```
True
```

In [60]:
```
# Display the images using matplotlib
plt.figure(figsize=(12, 8))
plt.subplot(1, 3, 1)
plt.title('Content Image')
plt.imshow(np.squeeze(content_image))
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title('Style Image')
plt.imshow(np.squeeze(style_image))
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title('Stylized Image')
plt.imshow(np.squeeze(stylized_image))
plt.axis('off')

plt.show()
```



Content Image

Style Image

Stylized Image

In [61]:

```
# Resize the style_image to match the shape of the content_image
style_image_resized = tf.image.resize(style_image, content_image.shape[1:3], metho

# Calculate Mean Squared Error
mse3 = tf.reduce_mean(tf.square(content_image - style_image_resized)).numpy()
print(f"Mean Squared Error: {mse3}")
```

```
Mean Squared Error: 0.13691847026348114
```

In [62]:

```
content_image_vgg = preprocess_for_vgg(content_image)
stylized_image_vgg = preprocess_for_vgg(stylized_image)

content_features = vgg(content_image_vgg)
stylized_features = vgg(stylized_image_vgg)

perceptual_loss3 = tf.reduce_mean(tf.square(content_features - stylized_features))
print(f"Perceptual Loss: {perceptual_loss3}")
```

```
Perceptual Loss: 44.0880126953125
```

In [63]:

```
style_image_vgg = preprocess_for_vgg(style_image)
stylized_image_vgg = preprocess_for_vgg(stylized_image)  # You might have missed t

style_features = vgg(style_image_vgg)
stylized_features = vgg(stylized_image_vgg)

style_gram = gram_matrix(style_features)
stylized_gram = gram_matrix(stylized_features)


gram_loss3 = tf.reduce_mean(tf.square(style_gram - stylized_gram)).numpy()
print(f"Gram Matrix Loss: {gram_loss3}")
```

```
Gram Matrix Loss: 0.0013236101949587464
```

In [64]:

```
# Calculate total variation loss for the stylized image
tv_loss3 = total_variation_loss(stylized_image)
```

```
print(f"Total Variation Loss: {tv_loss3.numpy()}")
```

```
Total Variation Loss: 852280.375
```

In [ ]:

Sample 4

In [66]:
```
content_image = load_image('C:/Users/soumi/3.DMML_2/dataset/content/35 (2).jpg')
style_image = load_image('C:/Users/soumi/3.DMML_2/dataset/art_work/Andy_Warhol_24.j
```

In [67]:
```
content_image.shape
```

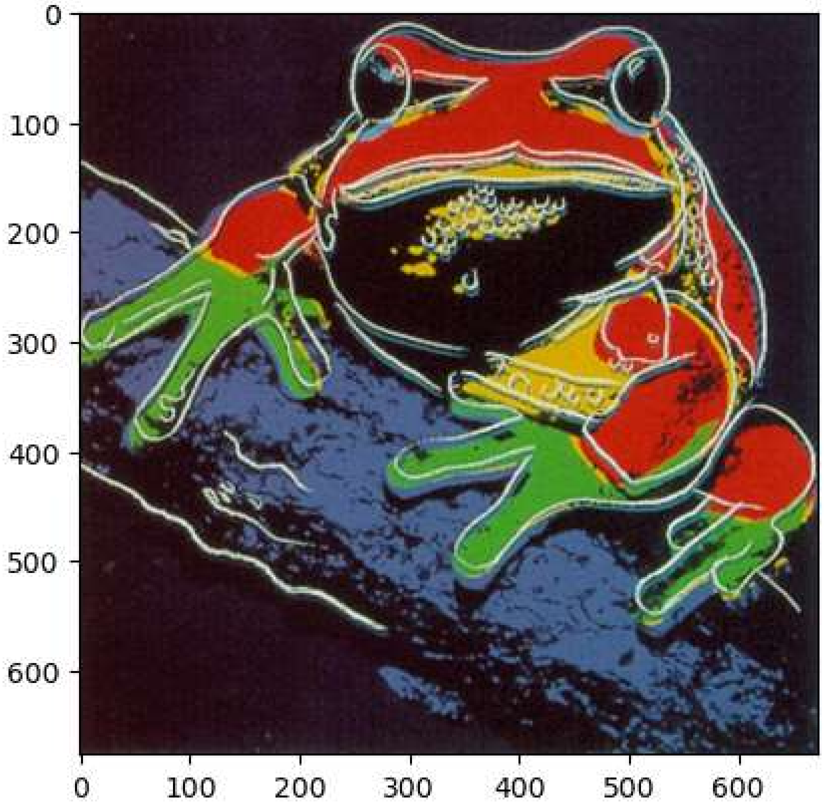
Out[67]:
```
TensorShape([1, 1000, 1500, 3])
```

In [68]:
```
style_image.shape
```

Out[68]:
```
TensorShape([1, 676, 672, 3])
```

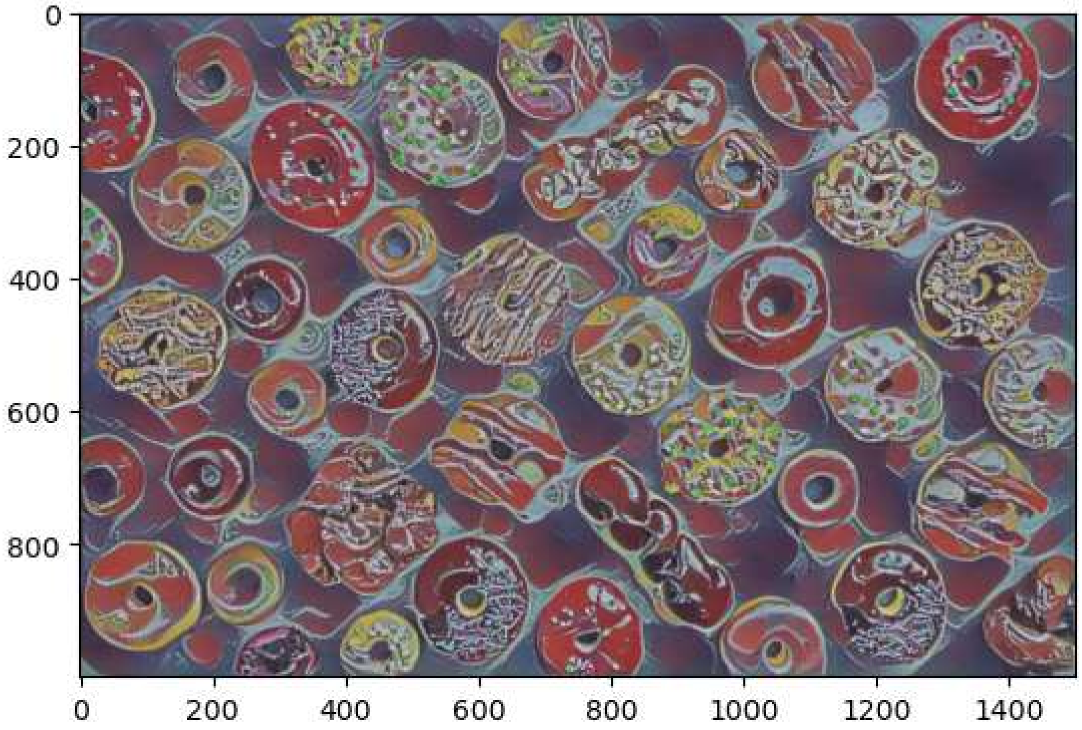
In [69]:
```
plt.imshow(np.squeeze(content_image))
plt.show()
```

In [70]:
```
plt.imshow(np.squeeze(style_image))
plt.show()
```



In [71]:
```
stylized_image = model(tf.constant(content_image), tf.constant(style_image))[0]
stylized_image4 = stylized_image
```

In [72]:
```
plt.imshow(np.squeeze(stylized_image))
plt.show()
```



In [73]:
```
cv2.imwrite('generated_img.jpg', cv2.cvtColor(np.squeeze(stylized_image)*255, cv2.(
```

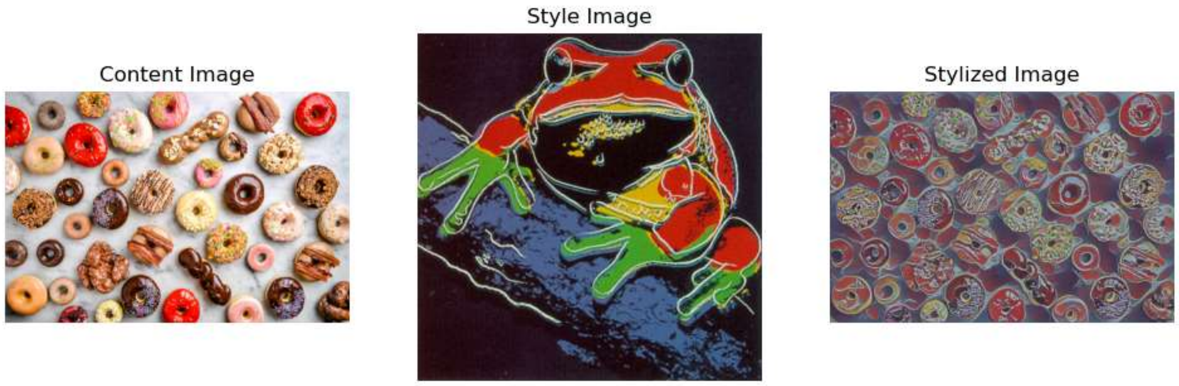
Out[73]:
```
True
```

In [74]:
```
# Display the images using matplotlib
plt.figure(figsize=(12, 8))
plt.subplot(1, 3, 1)
plt.title('Content Image')
plt.imshow(np.squeeze(content_image))
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title('Style Image')
plt.imshow(np.squeeze(style_image))
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title('Stylized Image')
plt.imshow(np.squeeze(stylized_image))
plt.axis('off')

plt.show()
```



In [75]:

```
# Resize the style_image to match the shape of the content_image
style_image_resized = tf.image.resize(style_image, content_image.shape[1:3], metho

# Calculate Mean Squared Error
mse4 = tf.reduce_mean(tf.square(content_image - style_image_resized)).numpy()
print(f"Mean Squared Error: {mse4}")
```

```
Mean Squared Error: 0.25097450613975525
```

In [76]:

```
content_image_vgg = preprocess_for_vgg(content_image)
stylized_image_vgg = preprocess_for_vgg(stylized_image)

content_features = vgg(content_image_vgg)
stylized_features = vgg(stylized_image_vgg)

perceptual_loss4 = tf.reduce_mean(tf.square(content_features - stylized_features))
print(f"Perceptual Loss: {perceptual_loss4}")
```

```
Perceptual Loss: 171.49356079101562
```

In [77]:

```
style_image_vgg = preprocess_for_vgg(style_image)
stylized_image_vgg = preprocess_for_vgg(stylized_image)  # You might have missed th

style_features = vgg(style_image_vgg)
stylized_features = vgg(stylized_image_vgg)

style_gram = gram_matrix(style_features)
stylized_gram = gram_matrix(stylized_features)


gram_loss4 = tf.reduce_mean(tf.square(style_gram - stylized_gram)).numpy()
print(f"Gram Matrix Loss: {gram_loss4}")
```

```
Gram Matrix Loss: 0.000519386085215956
```

In [78]:

```
# Calculate total variation loss for the stylized image
tv_loss4 = total_variation_loss(stylized_image)
print(f"Total Variation Loss: {tv_loss4.numpy()}")
```

```
Total Variation Loss: 277654.90625
```

Comparing MSE

In [104…

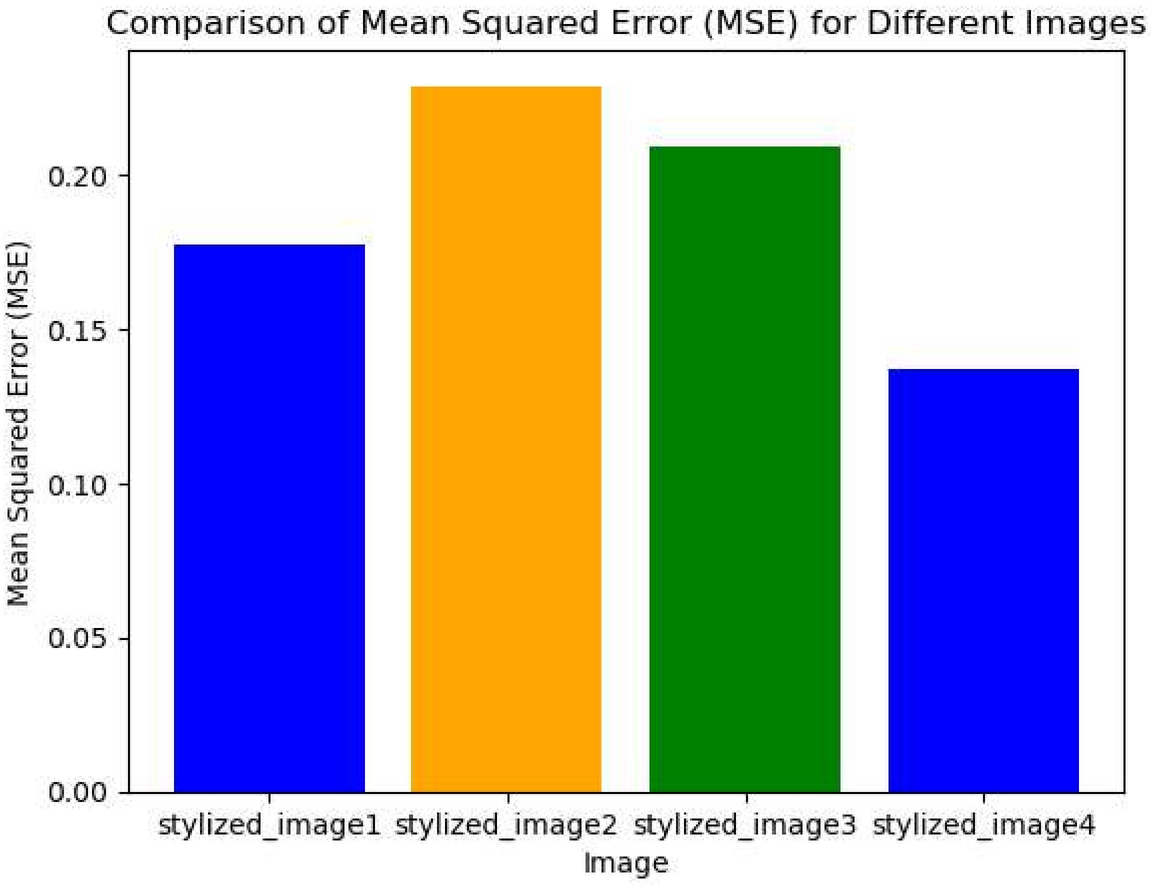
```
import matplotlib.pyplot as plt

# List of image names and their corresponding MSE values
image_names = ['stylized_image1', 'stylized_image2', 'stylized_image3', 'stylized_
mse_values = [mse1, mse2, mse3, mse4, ]  # Replace with actual MSE values

# Create a bar chart
plt.bar(image_names, mse_values, color=['blue', 'orange', 'green'])

# Add labels and title
plt.xlabel('Image')
plt.ylabel('Mean Squared Error (MSE)')
plt.title('Comparison of Mean Squared Error (MSE) for Different Images')

# Display the chart
plt.show()
```



In [ ]: