Learning Objectives

Learners will be able to...

- **Understand the concept of image animation and how it can be applied to generated images.**
- **Learn how to create a sequence of images with varying properties using the DALL-E 2 API.**
- **Explore the use of the Python Imaging Library (PIL) to process and manipulate the sequence of images for animation purposes.**
- **Learn how to create a GIF (Graphics Interchange Format) animation using the sequence of generated images.**
- **Understand the importance of animation settings, such as frame duration and loop count, and how they can be adjusted to create a smoother and more visually appealing animation.**

info

Make Sure You Know

You are familiar with Python.

Limitations

This is a gentle introduction. So there is a little bit of Python programming. The information might not be the most up to data as OpenAI releases new features.

Main File

Before we start, we are going to create a main with some functions. This way we can use the functions across different files. This way we have access to all our libraries, api keys and image generation file.

```
import os
import openai
from PIL import Image, ImageOps,ImageChops
from io import BytesIO
import requests

# Set environment variables
openai.api_key =  os.getenv('OPENAI_KEY')

# Generate the base image
def generate_base_image(prompt):
    response = openai.Image.create(
        prompt=prompt,
        n=1,
        size="512x512"
    )
    return response['data'][0]['url']

def download_image(image_url,x):
    response = requests.get(image_url)
    img_data = response.content
    img = Image.open(BytesIO(img_data))
    with open(x+'.jpg', 'wb') as handler:
      handler.write(img_data)
    return img
```

This script essentially provides the functionality to generate an image based on a text prompt using the OpenAI API and then download and save that image to the local file system.

Now in all files we can just , import main and we will have access to our library and functions.

Intro

we'll explore how to create animations using images generated by the DALL-E 2 API. Animations can be an effective way to showcase the capabilities of the DALL-E 2 API and bring generated images to life. We'll be using the Python Imaging Library (PIL) to process and manipulate the generated images and create a GIF animation.

Creating a Sequence of Images

The first step in creating an animation is to generate a sequence of images with varying properties. We'll use the DALL-E 2 API to generate these images. For this example, let's create an animation of a rotating Earth.

To generate a sequence of images, we can loop through different prompts or modify the prompt with changing properties. In this case, we'll generate images of Earth from different angles. Here's an example code snippet to generate the sequence of images:

```
def generate_earth_image(angle):
    prompt = f"Realistic Earth from space at {angle} degrees
        angle"
    image_url = generate_base_image(prompt)
    image_filename = f"earth_{angle}_degrees"
    image = download_image(image_url, image_filename)
    return image

angles = range(0, 360, 10)
earth_images = [generate_earth_image(angle) for angle in angles]
```

In this code, we define a function generate_earth_image that takes an angle as an argument and generates an image of Earth from that angle using the DALL-E 2 API. We then create a list of angles from 0 to 350 degrees in increments of 10 degrees and generate a corresponding image for each angle.

The files are provided for you because the code might take a bit to load since generating and saving a ton of pictures. IF YOU WANT YOU CAN REGENERATE THEM USING THE TRY IT BUTTON BELOW. Not recommended

Here is a sample of the earth at 230 degrees.

earth_230_degrees

After Generating your images please comment out the code that would generate the images.

Processing the Images

Now that we have generated and downloaded the Earth images at different angles, it's time to process them and create a GIF animation. We'll be using the Python Imaging Library (PIL) to accomplish this.

First, let's resize the images so the animation is smaller in size and easier to handle. In the following code, we resize each image in the earth_images list to 256x256 pixels:

```
angles = range(0, 360, 10)
image_filenames = [f"earth_{angle}_degrees.jpg" for angle in
        angles]

resized_earth_images = []
for filename in image_filenames:
    img = Image.open(filename)
    resized_img = img.resize((256, 256), Image.ANTIALIAS)
    resized_earth_images.append(resized_img)
```

we create a list of image filenames called image_filenames for the Earth images at different angles. We then create an empty list resized_earth_images to store the resized images. In the for loop, we open each image using its filename, resize it, and append it to the resized_earth_images list.

Creating the GIF Animation

Next, we'll create a GIF animation using the resized images. We can do this using the save method of the PIL Image class. We'll set the duration of each frame to 100 milliseconds and loop the animation indefinitely:

Creating the GIF Animation

Next, we'll create a GIF animation using the resized images. We can do this using the save method of the PIL Image class. We'll set the duration of each frame to 100 milliseconds and loop the animation indefinitely:

```
output_gif = "rotating_earth.gif"
resized_earth_images[0].save(
    output_gif,
    save_all=True,
    append_images=resized_earth_images[1:],
    duration=100,
    loop=0
)
```

In this code, we first specify the output GIF filename as “rotating_earth.gif”. Then, we call the save method on the first image in the resized_earth_images list. We set save_all=True to indicate that we want to save multiple frames. The append_images parameter is set to the rest of the images in the list, and we specify the duration and loop count using the duration and loop parameters, respectively.

Now, when you run this code, a GIF animation called “rotating_earth.gif” will be created, showcasing the Earth rotating using the images generated by the DALL-E 2 API.

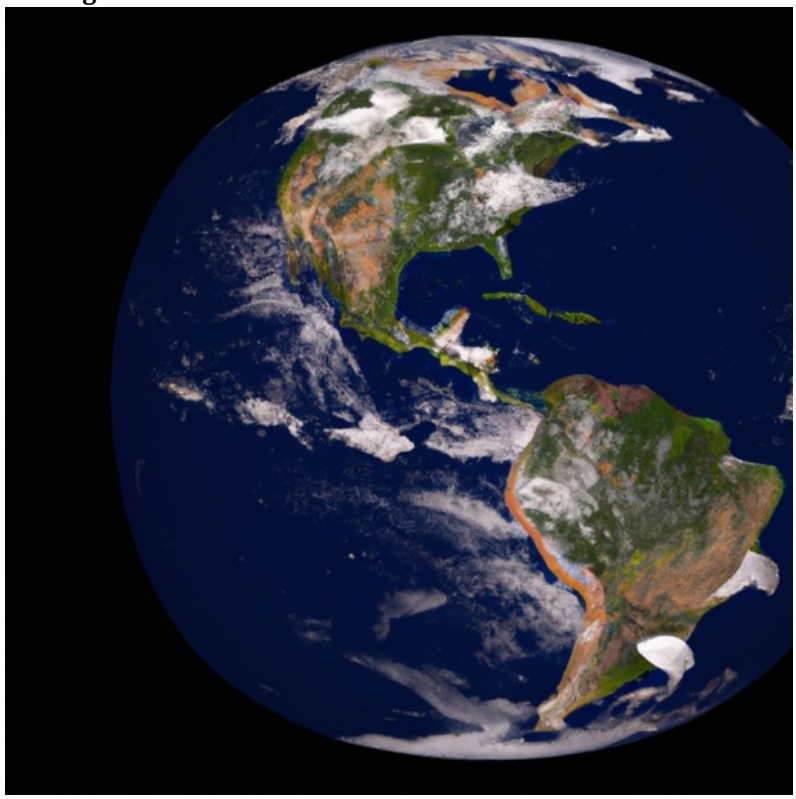
rotating_earth

The images that the AI Genrated did not all look the same hence our current gif. The different images dont have the same center there fore our gif looks less smooth

0 degrees

270 degrees

350 degrees



Optimizing Animation Settings

we will explore various options to optimize the GIF animation created using the resized Earth images. We will focus on adjusting the frame duration, loop count, and image optimization settings to make the animation smoother and more visually appealing.

You can adjust the duration of each frame in the animation to make the rotation appear faster or slower. The duration parameter in the save method determines the time each frame is displayed in milliseconds. A lower duration will result in a faster rotation, while a higher duration will make the rotation slower. Experiment with different values to find the optimal speed for your animation. Change the duration from 100 to 50 to 200 and compare.

```
duration = 50  # Adjust the frame duration to your preference
```

and

```
duration = 200  # Adjust the frame duration to your preference
```

Adjusting Loop Count

You can control the number of times the animation loops by setting the loop parameter in the save method. A value of 0 indicates an infinite loop, while a higher value specifies the exact number of loops. For example, if you want the animation to loop only three times, set the loop parameter to 3

```
loop_count = 3  # Adjust the loop count to your preference
```

Image Optimization
By default, the PIL library doesn't optimize the images in the animation. You can enable image optimization by setting the optimize parameter to True in the save method. This can reduce the file size of the animation while maintaining visual quality.

```

optimize = True  # Enable image optimization

```

Now, let's put everything together and create an optimized GIF animation:

```
output_gif = "optimized_rotating_earth.gif"
resized_earth_images[0].save(
    output_gif,
    save_all=True,
    append_images=resized_earth_images[1:],
    duration=duration,
    loop=loop_count,
    optimize=optimize
)
```

With these optimizations in place, you should now have a smoother, more visually appealing animation that showcases the Earth rotating using the images generated by the DALL-E 2 API.

Adding Transitions and Visual Effects

In this section, we will explore how to enhance the GIF animation by adding transitions and visual effects to the images generated by the DALL-E 2 API. Adding transitions can make the animation smoother, while visual effects can create a more engaging and dynamic presentation.

1. **Adding Crossfade Transitions**:

A crossfade transition blends two consecutive images, gradually transitioning from one image to the next. This can create a smoother rotation effect in the animation. Here's an example of how to create a crossfade transition using the ImageChops module:

```
from PIL import ImageChops

def crossfade(image1, image2, alpha):
    return ImageChops.blend(image1, image2, alpha)

crossfade_frames = []
for i in range(len(resized_earth_images) - 1):
    for alpha in (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9):
        frame = crossfade(resized_earth_images[i],
        resized_earth_images[i + 1], alpha)
        crossfade_frames.append(frame)
```

In this code snippet, we create a `crossfade` function that blends two images using a specified alpha value. We then generate a list of crossfade frames by blending consecutive images in the resized_earth_images list.

2.Adding Visual Effects

You can also add visual effects to the animation, such as changing the brightness, contrast, or color of the images. In this example, we will adjust the brightness of the images using the ImageEnhance module:

```
from PIL import ImageEnhance

def adjust_brightness(image, factor):
    enhancer = ImageEnhance.Brightness(image)
    return enhancer.enhance(factor)

brightness_factor = 1.5
brightened_frames = [adjust_brightness(frame, brightness_factor)
        for frame in crossfade_frames]
```

In this code snippet, we create an adjust_brightness function that modifies the brightness of an image using a specified factor. We then create a list of brightened frames by applying the brightness adjustment to the crossfade_frames list.

3. **Creating the Enhanced Animation**
   Now that we have the crossfade transitions and visual effects in place, we can create an enhanced GIF animation using the modified frames:

```
output_gif = "enhanced_rotating_earth.gif"
brightened_frames[0].save(
    output_gif,
    save_all=True,
    append_images=brightened_frames[1:],
    duration=duration,
    loop=loop_count,
    optimize=optimize
)
```

With these transitions and visual effects, your animation should be more engaging and dynamic, showcasing the power and flexibility of the DALL-E 2 API.