Edward Goonzalez

Non-Linear Narrative

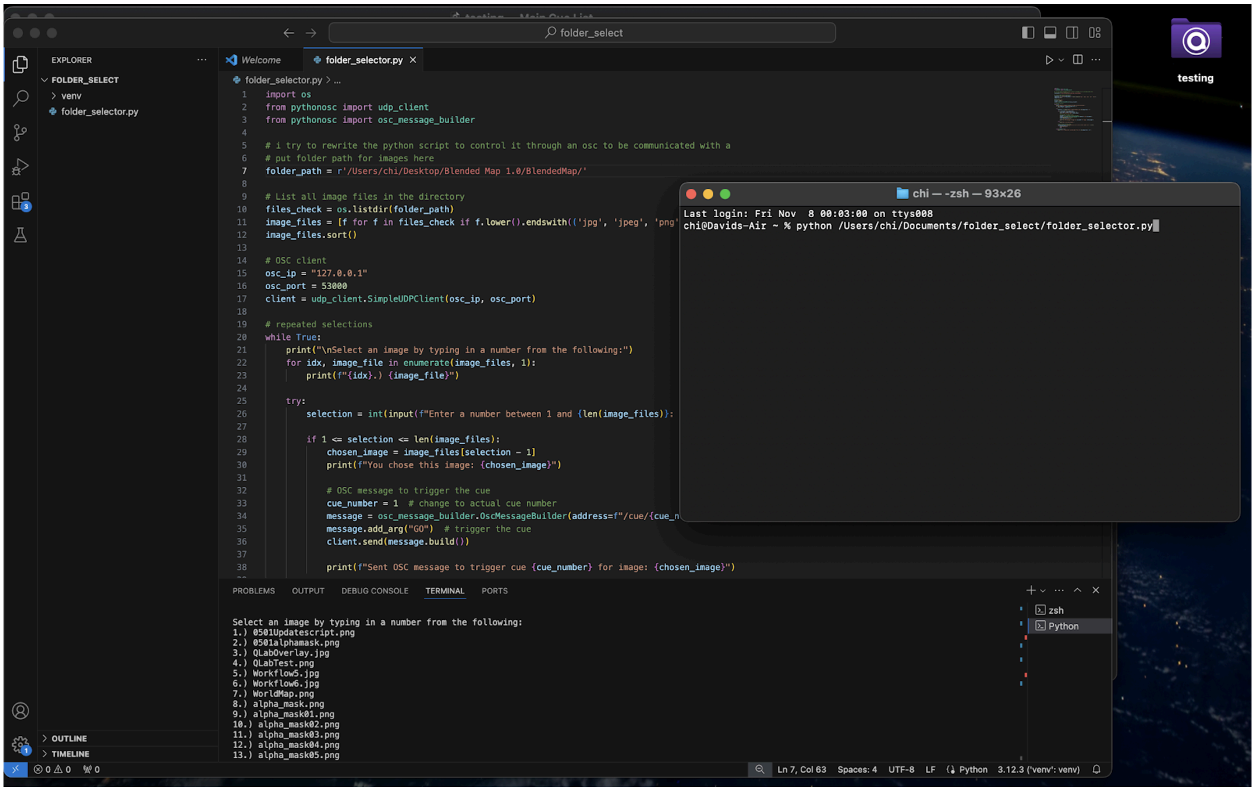
December 5th, 2024

Basic Media Controls in QLab

Introduction:

This Python script allows users to select an image from a specified folder while sending an OSC (Open Sound Control) message to trigger an event, such as a “cue”, based on the selected image. The selected image will show that it has been selected from the number “cue”.This research project will serve as a resource for future courses and projects at CityTech that require students to use QLab. The proposal provides a foundational understanding of how QLab can be controlled and integrated with external programs via OSC communication. If this communication is successfully established, it opens the door to various types of interactions between the external program and QLab.

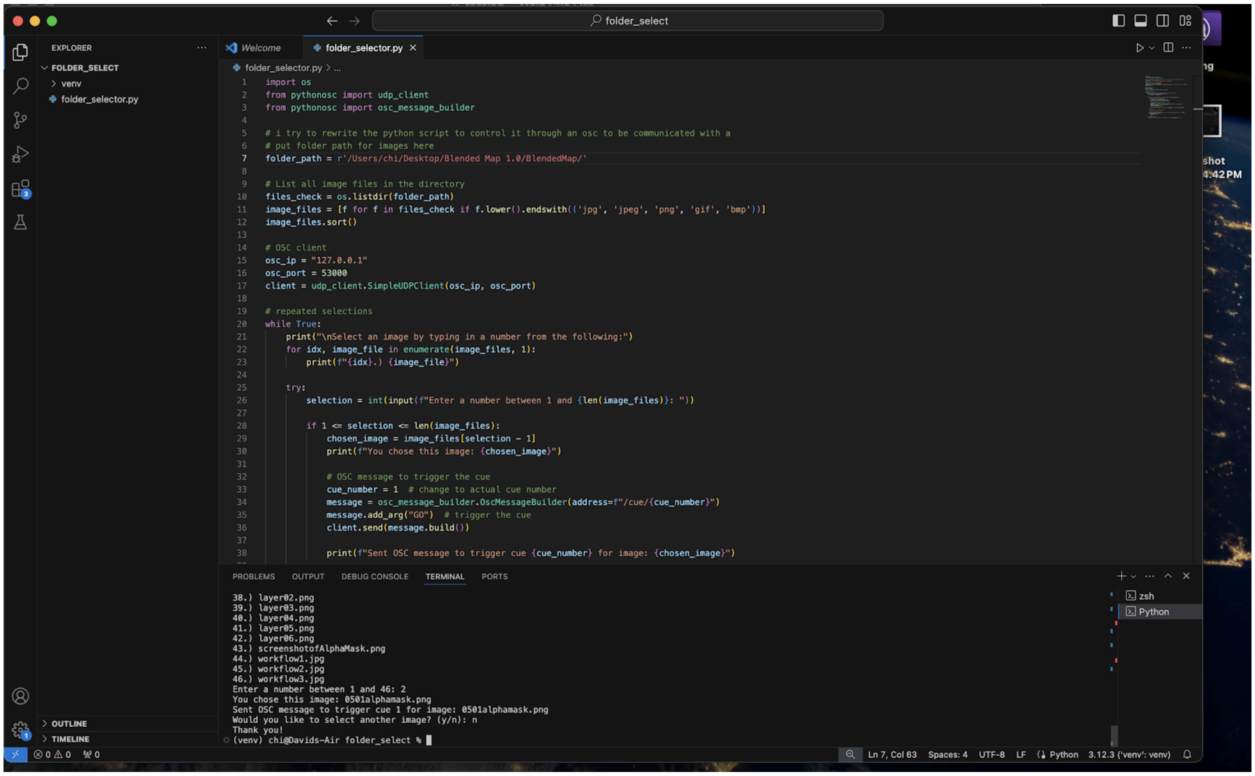
Images:



```python
import os
from pythonosc import udp_client
from pythonosc import osc_message_builder

# i try to rewrite the python script to control it through an osc to be communicated with a
# put folder path for images here
folder_path = r'/Users/chi/Desktop/Blended Map 1.0/BlendedMap/'

# List all image files in the directory
files_check = os.listdir(folder_path)
image_files = [f for f in files_check if f.lower().endswith(('jpg', 'jpeg', 'png'
image_files.sort()

# OSC client
osc_ip = "127.0.0.1"
osc_port = 53000
client = udp_client.SimpleUDPClient(osc_ip, osc_port)

# repeated selections
while True:
    print("\nSelect an image by typing in a number from the following:")
    for idx, image_file in enumerate(image_files, 1):
        print(f"{idx}.) {image_file}")

    try:
        selection = int(input(f"Enter a number between 1 and {len(image_files)}:

        if 1 <= selection <= len(image_files):
            chosen_image = image_files[selection - 1]
            print(f"You chose this image: {chosen_image}")

            # OSC message to trigger the cue
            cue_number = 1  # change to actual cue number
            message = osc_message_builder.OscMessageBuilder(address=f"/cue/{cue_n
            message.add_arg("GO")  # trigger the cue
            client.send(message.build())

            print(f"Sent OSC message to trigger cue {cue_number} for image: {chosen_image}")
```

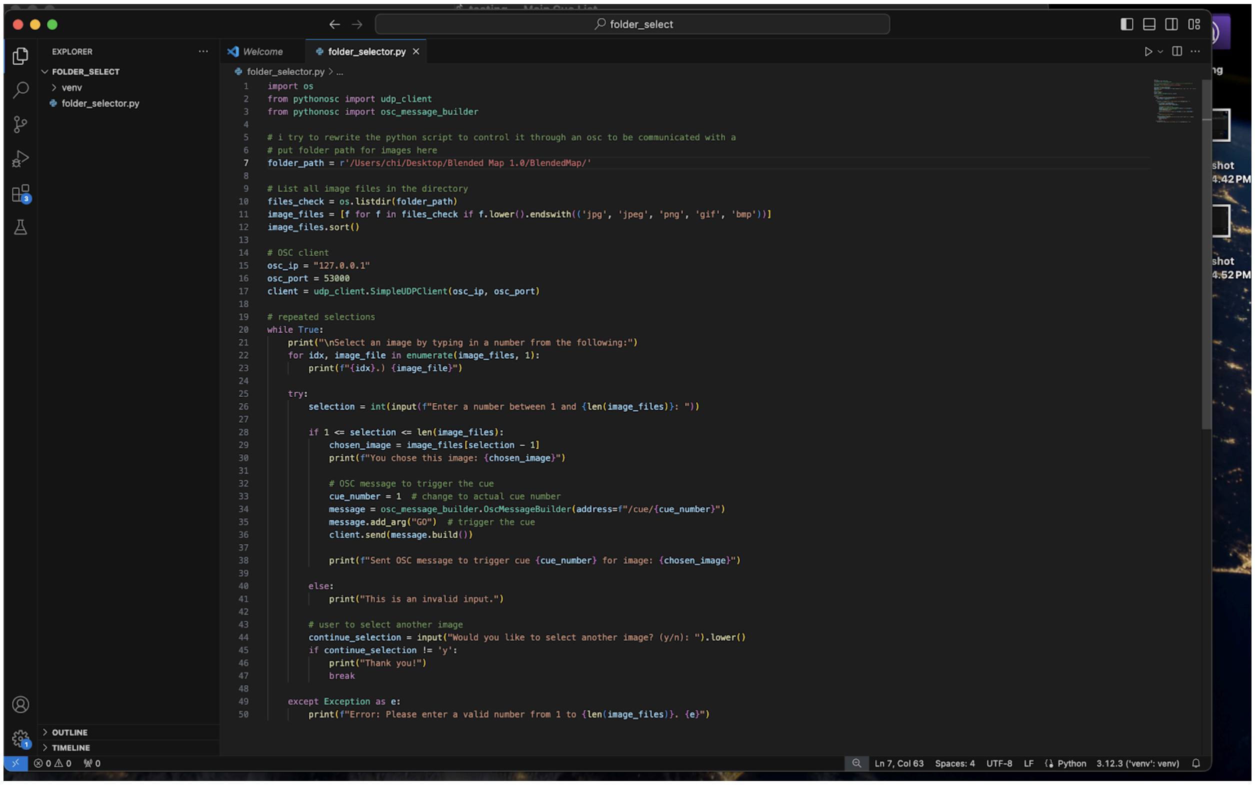
```
Last login: Fri Nov  8 00:03:00 on ttys008
chi@Davids-Air ~ % python /Users/chi/Documents/folder_select/folder_selector.py
```

```
Select an image by typing in a number from the following:
1.) 0501Updatescript.png
2.) 0501alphamask.png
3.) QLabOverlay.jpg
4.) QLabTest.png
5.) Workflow5.jpg
6.) Workflow6.jpg
7.) WorldMap.png
8.) alpha_mask.png
9.) alpha_mask01.png
10.) alpha_mask02.png
11.) alpha_mask03.png
12.) alpha_mask04.png
13.) alpha_mask05.png
```



```python
import os
from pythonosc import udp_client
from pythonosc import osc_message_builder

# i try to rewrite the python script to control it through an osc to be communicated with a
# put folder path for images here
folder_path = r'/Users/chi/Desktop/Blended Map 1.0/BlendedMap/'

# List all image files in the directory
files_check = os.listdir(folder_path)
image_files = [f for f in files_check if f.lower().endswith(('jpg', 'jpeg', 'png', 'gif', 'bmp'))]
image_files.sort()

# OSC client
osc_ip = "127.0.0.1"
osc_port = 53000
client = udp_client.SimpleUDPClient(osc_ip, osc_port)

# repeated selections
while True:
    print("\nSelect an image by typing in a number from the following:")
    for idx, image_file in enumerate(image_files, 1):
        print(f"{idx}.) {image_file}")

    try:
        selection = int(input(f"Enter a number between 1 and {len(image_files)}: "))

        if 1 <= selection <= len(image_files):
            chosen_image = image_files[selection - 1]
            print(f"You chose this image: {chosen_image}")

            # OSC message to trigger the cue
            cue_number = 1  # change to actual cue number
            message = osc_message_builder.OscMessageBuilder(address=f"/cue/{cue_number}")
            message.add_arg("GO")  # trigger the cue
            client.send(message.build())

            print(f"Sent OSC message to trigger cue {cue_number} for image: {chosen_image}")
```

```
38.) layer02.png
39.) layer03.png
40.) layer04.png
41.) layer05.png
42.) layer06.png
43.) screenshotofAlphaMask.png
44.) workflow1.jpg
45.) workflow2.jpg
46.) workflow3.jpg
Enter a number between 1 and 46: 2
You chose this image: 0501alphamask.png
Sent OSC message to trigger cue 1 for image: 0501alphamask.png
Would you like to select another image? (y/n): n
Thank you!
(venv) chi@Davids-Air folder_select %
```



```python
import os
from pythonosc import udp_client
from pythonosc import osc_message_builder

# i try to rewrite the python script to control it through an osc to be communicated with a
# put folder path for images here
folder_path = r'/Users/chi/Desktop/Blended Map 1.0/BlendedMap/'

# List all image files in the directory
files_check = os.listdir(folder_path)
image_files = [f for f in files_check if f.lower().endswith(('jpg', 'jpeg', 'png', 'gif', 'bmp'))]
image_files.sort()

# OSC client
osc_ip = "127.0.0.1"
osc_port = 53000
client = udp_client.SimpleUDPClient(osc_ip, osc_port)

# repeated selections
while True:
    print("\nSelect an image by typing in a number from the following:")
    for idx, image_file in enumerate(image_files, 1):
        print(f"{idx}.) {image_file}")

    try:
        selection = int(input(f"Enter a number between 1 and {len(image_files)}: "))

        if 1 <= selection <= len(image_files):
            chosen_image = image_files[selection - 1]
            print(f"You chose this image: {chosen_image}")

            # OSC message to trigger the cue
            cue_number = 1  # change to actual cue number
            message = osc_message_builder.OscMessageBuilder(address=f"/cue/{cue_number}")
            message.add_arg("GO")  # trigger the cue
            client.send(message.build())

            print(f"Sent OSC message to trigger cue {cue_number} for image: {chosen_image}")

        else:
            print("This is an invalid input.")

        # user to select another image
        continue_selection = input("Would you like to select another image? (y/n): ").lower()
        if continue_selection != 'y':
            print("Thank you!")
            break

    except Exception as e:
        print(f"Error: Please enter a valid number from 1 to {len(image_files)}. {e}")
```

Key Components:

- Image File Path Folder: The script lists all image files (JPG, JPEG, PNG, GIF, BMP) from a specified folder (folder_path).
- OSC Communication: The script uses the python-osc library to send OSC messages by getting a specified IP and port based on the QLab setup.
- User Interaction: The script prompts the user to choose an image by entering a number "cue" which corresponds to the list of images in the folder. Each image has an assigned number attached to them to be called back.
- OSC Trigger: Upon selecting an image, the OSC message is sent to a specific address to trigger an event (event = cue).

Description:

- Directory Listing: The folder path (folder_path) is set to a specific directory on the local system. All image files within the folder are listed, and only files with supported image extensions (jpg, jpeg, png, gif, bmp) are considered.
- User Image Selection: The user is shown a list of images; with numbers assigned to each of them. The user enters a number that corresponds to their choice of image.
- OSC Message: Once an image is selected, an OSC message is sent to trigger an event (e.g., a cue) with a predefined cue number (default is 1).

Repetition & Exit: After triggering the cue. The user is given a choice to either select another image or exit the program. This process repeats itself until the user opts to exit by entering 'n'.

Error Handling:

- If the user enters an invalid number (outside the list range), the script catches the error and prompts for a valid selection. Any other exceptions are caught and displayed as an error message.

Dependencies:

- python-osc: Library for OSC communication.
- os: For directory operations (Folder path).

Instructions to Run:

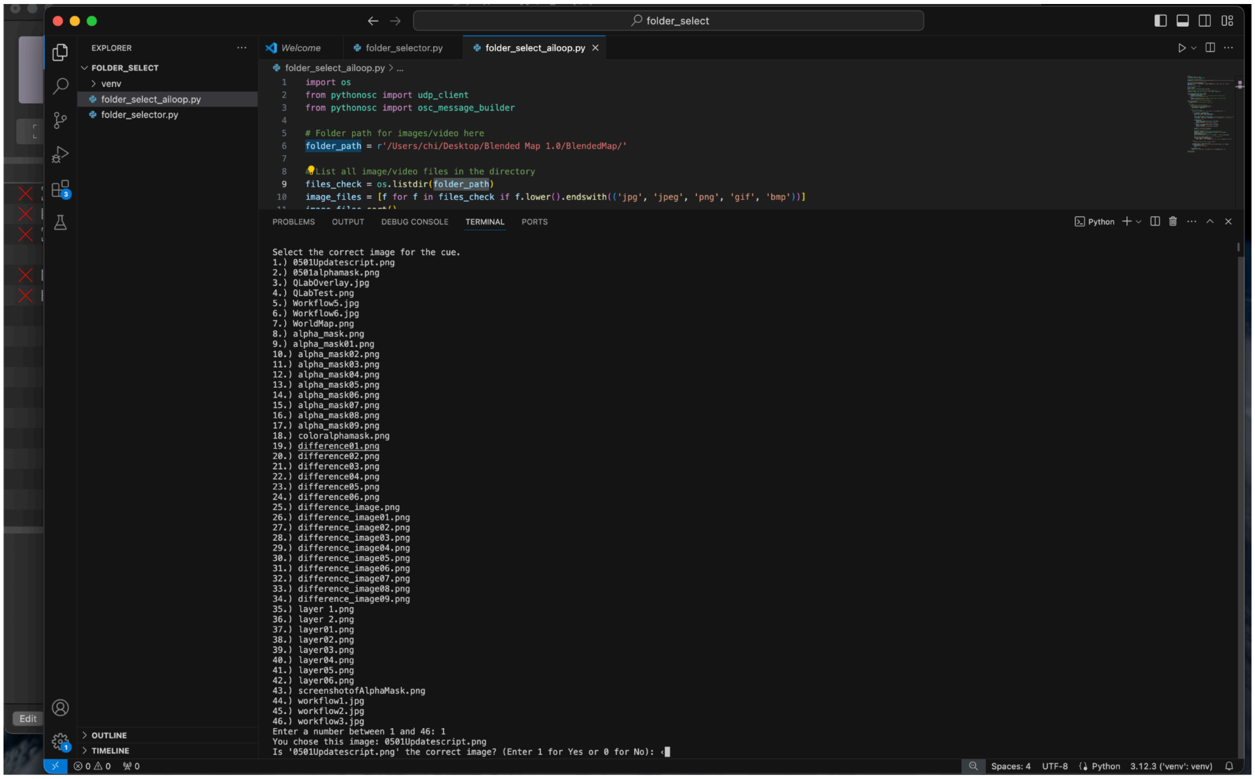
- Set the path to your image folder in folder_path.
- The OSC message is sent to the specified IP and port (osc_ip, osc_port).
- Run the script, and choose an image until you exit the program. Enjoy!

Implement feedback loop system (AI training)

Introduction:

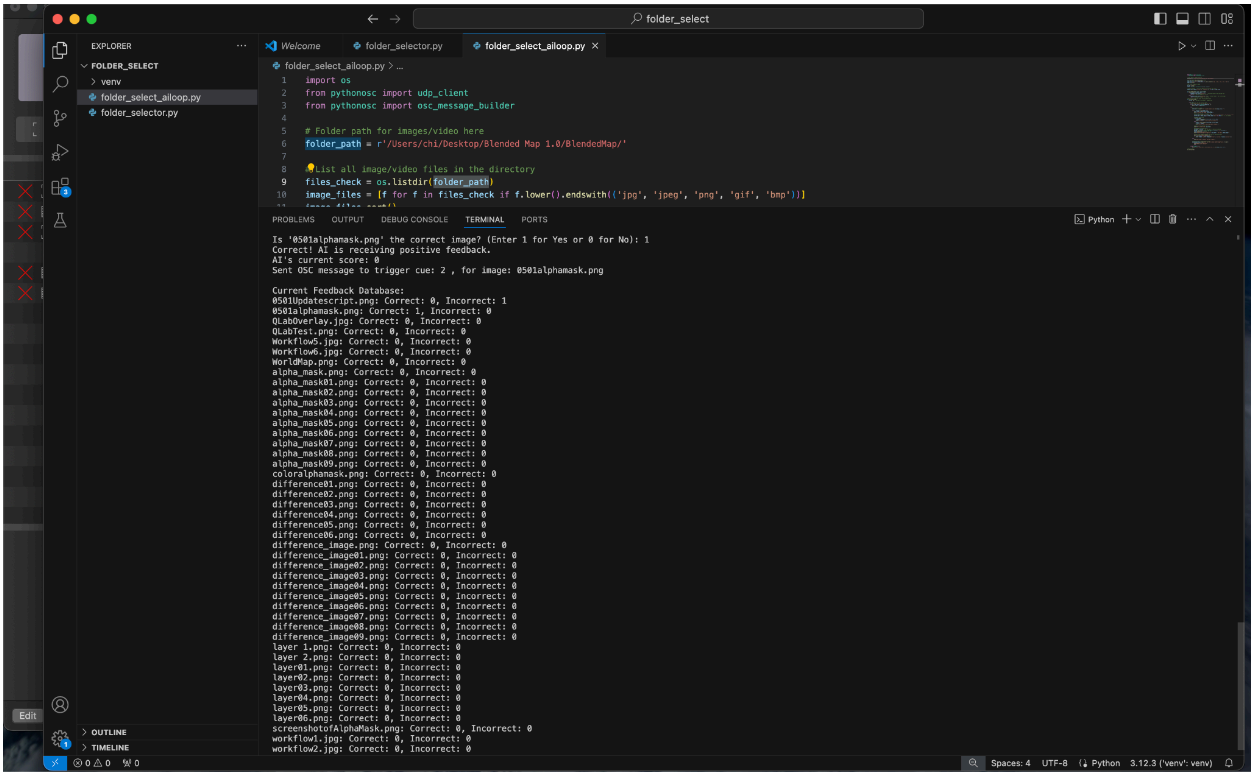
This Python script allows users to select an image from a specified folder while sending an OSC (Open Sound Control) message to trigger an event, such as a “cue”, based on the selected image. The selected image will show that it has been selected from the number “cue”. However, this program has an additional feature: an AI model that learns from user input. The model collects user feedback on selected images/videos from the specified folder, using a "good" or "bad" feedback system to improve its performance.

Images:



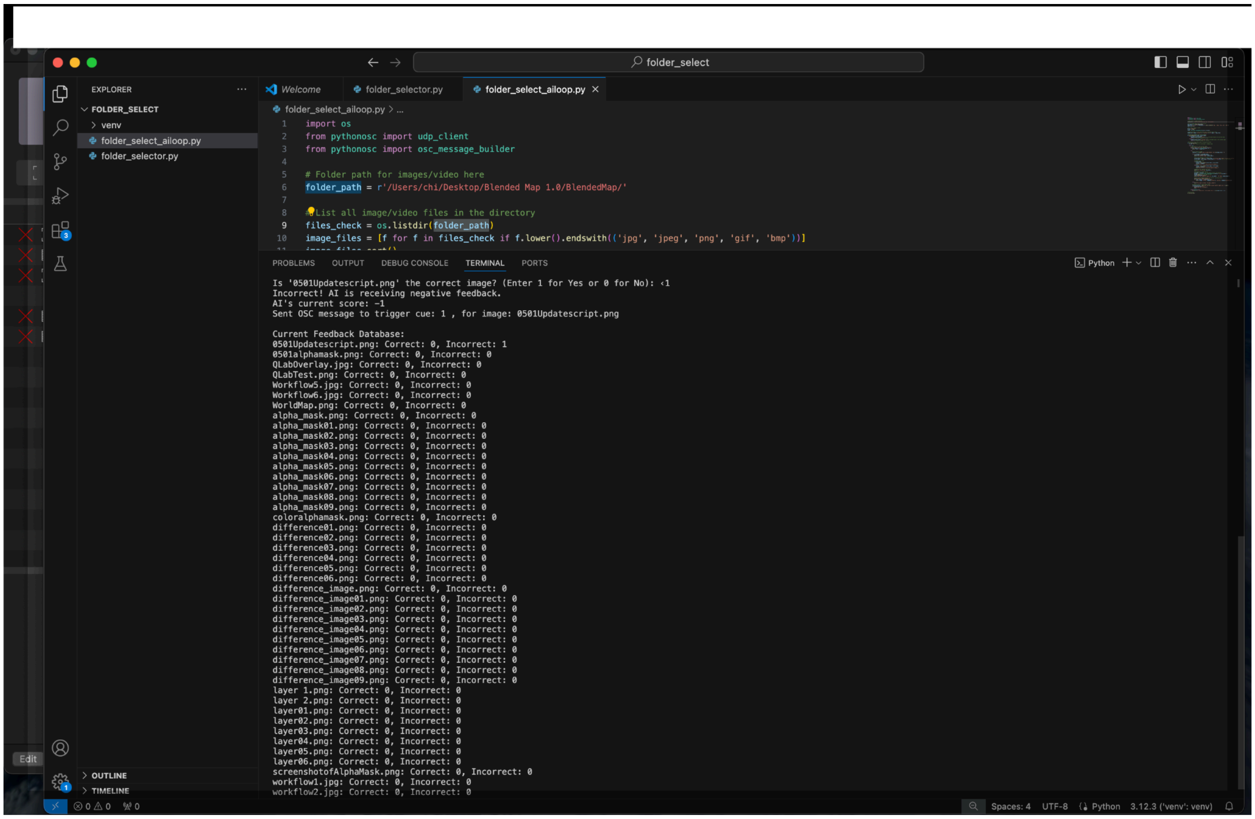
```python
import os
from pythonosc import udp_client
from pythonosc import osc_message_builder

# Folder path for images/video here
folder_path = r'/Users/chi/Desktop/Blended Map 1.0/BlendedMap/'

#List all image/video files in the directory
files_check = os.listdir(folder_path)
image_files = [f for f in files_check if f.lower().endswith(('jpg', 'jpeg', 'png', 'gif', 'bmp'))]
```

```
Select the correct image for the cue.
1.) 0501Updatescript.png
2.) 0501alphamask.png
3.) QLabOverlay.jpg
4.) QLabTest.png
5.) Workflow5.jpg
6.) Workflow6.jpg
7.) WorldMap.png
8.) alpha_mask.png
9.) alpha_mask01.png
10.) alpha_mask02.png
11.) alpha_mask03.png
12.) alpha_mask04.png
13.) alpha_mask05.png
14.) alpha_mask06.png
15.) alpha_mask07.png
16.) alpha_mask08.png
17.) alpha_mask09.png
18.) coloralphamask.png
19.) difference01.png
20.) difference02.png
21.) difference03.png
22.) difference04.png
23.) difference05.png
24.) difference06.png
25.) difference_image.png
26.) difference_image01.png
27.) difference_image02.png
28.) difference_image03.png
29.) difference_image04.png
30.) difference_image05.png
31.) difference_image06.png
32.) difference_image07.png
33.) difference_image08.png
34.) difference_image09.png
35.) layer 1.png
36.) layer 2.png
37.) layer01.png
38.) layer02.png
39.) layer03.png
40.) layer04.png
41.) layer05.png
42.) layer06.png
43.) screenshotofAlphaMask.png
44.) workflow1.jpg
45.) workflow2.jpg
46.) workflow3.jpg
Enter a number between 1 and 46: 1
You chose this image: 0501Updatescript.png
Is '0501Updatescript.png' the correct image? (Enter 1 for Yes or 0 for No): 
```



```python
import os
from pythonosc import udp_client
from pythonosc import osc_message_builder

# Folder path for images/video here
folder_path = r'/Users/chi/Desktop/Blended Map 1.0/BlendedMap/'

#List all image/video files in the directory
files_check = os.listdir(folder_path)
image_files = [f for f in files_check if f.lower().endswith(('jpg', 'jpeg', 'png', 'gif', 'bmp'))]
```

```
Is '0501alphamask.png' the correct image? (Enter 1 for Yes or 0 for No): 1
Correct! AI is receiving positive feedback.
AI's current score: 0
Sent OSC message to trigger cue: 2 , for image: 0501alphamask.png

Current Feedback Database:
0501Updatescript.png: Correct: 0, Incorrect: 1
0501alphamask.png: Correct: 1, Incorrect: 0
QLabOverlay.jpg: Correct: 0, Incorrect: 0
QLabTest.png: Correct: 0, Incorrect: 0
Workflow5.jpg: Correct: 0, Incorrect: 0
Workflow6.jpg: Correct: 0, Incorrect: 0
WorldMap.png: Correct: 0, Incorrect: 0
alpha_mask.png: Correct: 0, Incorrect: 0
alpha_mask01.png: Correct: 0, Incorrect: 0
alpha_mask02.png: Correct: 0, Incorrect: 0
alpha_mask03.png: Correct: 0, Incorrect: 0
alpha_mask04.png: Correct: 0, Incorrect: 0
alpha_mask05.png: Correct: 0, Incorrect: 0
alpha_mask06.png: Correct: 0, Incorrect: 0
alpha_mask07.png: Correct: 0, Incorrect: 0
alpha_mask08.png: Correct: 0, Incorrect: 0
alpha_mask09.png: Correct: 0, Incorrect: 0
coloralphamask.png: Correct: 0, Incorrect: 0
difference01.png: Correct: 0, Incorrect: 0
difference02.png: Correct: 0, Incorrect: 0
difference03.png: Correct: 0, Incorrect: 0
difference04.png: Correct: 0, Incorrect: 0
difference05.png: Correct: 0, Incorrect: 0
difference06.png: Correct: 0, Incorrect: 0
difference_image.png: Correct: 0, Incorrect: 0
difference_image01.png: Correct: 0, Incorrect: 0
difference_image02.png: Correct: 0, Incorrect: 0
difference_image03.png: Correct: 0, Incorrect: 0
difference_image04.png: Correct: 0, Incorrect: 0
difference_image05.png: Correct: 0, Incorrect: 0
difference_image06.png: Correct: 0, Incorrect: 0
difference_image07.png: Correct: 0, Incorrect: 0
difference_image08.png: Correct: 0, Incorrect: 0
difference_image09.png: Correct: 0, Incorrect: 0
layer 1.png: Correct: 0, Incorrect: 0
layer 2.png: Correct: 0, Incorrect: 0
layer01.png: Correct: 0, Incorrect: 0
layer02.png: Correct: 0, Incorrect: 0
layer03.png: Correct: 0, Incorrect: 0
layer04.png: Correct: 0, Incorrect: 0
layer05.png: Correct: 0, Incorrect: 0
layer06.png: Correct: 0, Incorrect: 0
screenshotofAlphaMask.png: Correct: 0, Incorrect: 0
workflow1.jpg: Correct: 0, Incorrect: 0
workflow2.jpg: Correct: 0, Incorrect: 0
```



```python
import os
from pythonosc import udp_client
from pythonosc import osc_message_builder

# Folder path for images/video here
folder_path = r'/Users/chi/Desktop/Blended Map 1.0/BlendedMap/'

#List all image/video files in the directory
files_check = os.listdir(folder_path)
image_files = [f for f in files_check if f.lower().endswith(('jpg', 'jpeg', 'png', 'gif', 'bmp'))]
```

```
Is '0501Updatescript.png' the correct image? (Enter 1 for Yes or 0 for No): ‹1
Incorrect! AI is receiving negative feedback.
AI's current score: -1
Sent OSC message to trigger cue: 1 , for image: 0501Updatescript.png

Current Feedback Database:
0501Updatescript.png: Correct: 0, Incorrect: 1
0501alphamask.png: Correct: 0, Incorrect: 0
QLabOverlay.jpg: Correct: 0, Incorrect: 0
QLabTest.png: Correct: 0, Incorrect: 0
Workflow5.jpg: Correct: 0, Incorrect: 0
Workflow6.jpg: Correct: 0, Incorrect: 0
WorldMap.png: Correct: 0, Incorrect: 0
alpha_mask.png: Correct: 0, Incorrect: 0
alpha_mask01.png: Correct: 0, Incorrect: 0
alpha_mask02.png: Correct: 0, Incorrect: 0
alpha_mask03.png: Correct: 0, Incorrect: 0
alpha_mask04.png: Correct: 0, Incorrect: 0
alpha_mask05.png: Correct: 0, Incorrect: 0
alpha_mask06.png: Correct: 0, Incorrect: 0
alpha_mask07.png: Correct: 0, Incorrect: 0
alpha_mask08.png: Correct: 0, Incorrect: 0
alpha_mask09.png: Correct: 0, Incorrect: 0
coloralphamask.png: Correct: 0, Incorrect: 0
difference01.png: Correct: 0, Incorrect: 0
difference02.png: Correct: 0, Incorrect: 0
difference03.png: Correct: 0, Incorrect: 0
difference04.png: Correct: 0, Incorrect: 0
difference05.png: Correct: 0, Incorrect: 0
difference06.png: Correct: 0, Incorrect: 0
difference_image.png: Correct: 0, Incorrect: 0
difference_image01.png: Correct: 0, Incorrect: 0
difference_image02.png: Correct: 0, Incorrect: 0
difference_image03.png: Correct: 0, Incorrect: 0
difference_image04.png: Correct: 0, Incorrect: 0
difference_image05.png: Correct: 0, Incorrect: 0
difference_image06.png: Correct: 0, Incorrect: 0
difference_image07.png: Correct: 0, Incorrect: 0
difference_image08.png: Correct: 0, Incorrect: 0
difference_image09.png: Correct: 0, Incorrect: 0
layer 1.png: Correct: 0, Incorrect: 0
layer 2.png: Correct: 0, Incorrect: 0
layer01.png: Correct: 0, Incorrect: 0
layer02.png: Correct: 0, Incorrect: 0
layer03.png: Correct: 0, Incorrect: 0
layer04.png: Correct: 0, Incorrect: 0
layer05.png: Correct: 0, Incorrect: 0
layer06.png: Correct: 0, Incorrect: 0
screenshotofAlphaMask.png: Correct: 0, Incorrect: 0
workflow1.jpg: Correct: 0, Incorrect: 0
workflow2.jpg: Correct: 0, Incorrect: 0
```

Key Components:

- **Image File Path Folder:** The script identifies image files (JPG, JPEG, PNG, GIF, BMP) within a specified directory (folder_path).
- **OSC Communication:** The script utilizes the python-osc library to send Open Sound Control (OSC) messages. The script retrieves an IP address (osc_ip) and port (osc_port) to connect to a program like QLab.
- **User Interaction:** The script prompts the user to select an image by entering a number corresponding to an image in the list. The images are put into an array list.
- **OSC Trigger:** Selecting an image triggers an OSC message sent to a specific address to initiate an event (cue).

Description:

1. **Directory Listing:**
    - The script defines a specific directory path (folder_path) containing image files.
    - It lists all files within the directory and filters for image files with supported extensions (JPG, JPEG, PNG, GIF, BMP).
2. **User Image Selection:**
    - The user sees a numbered list of image filenames.
    - The user enters a number corresponding to their desired image.
3. **OSC Message:**
    - Upon image selection, an OSC message is sent to trigger an event (e.g., a cue).
    - The message includes a pre-defined cue number (default: 1) associated with the selected image.
4. **Repetition & Exit:**
    - After triggering the cue, the user can choose to select another image or exit the program.
    - This process repeats until the user enters "n" to exit.

5. **Error Handling:**
   - ○ The script catches errors for invalid user input (numbers outside the list range) and prompts for a valid selection. Any other exceptions are caught as error messages.

Artificial Intelligence API: Choices for QLab:

API	COST	IMPORT FORMATS	EXPORT FORMATS	FEATURES
Open AI (DALL·E/ChatGPT)	Free & w/Payment Plan (ChatGPT $20/month, DALL·E $0.02/images(	Text	Text (ChatGPT) Image (DALL·E | PNG JPEG)	Text & Image Generation. No sound but can be paired with a sound system.
Google Cloud Vision	Free (First 1000 units) & Payment ($1.50 per 1001-5,000,000 units)	JPG/JPEG, PNG, GIF, etc. (Image files to OCR on text for images)	JSON (objects and text but no sound)	Image analysis
RunwayML	Free & Payment ($12/month)	Images, Video, Text	JPG/JPEG, PNG, MP4, JSON, CSV	Image & Video Generation, AI toolsets. Some models support audio processing.
Clarifai	Free & Payment ($25/month)	JPG/JPEG, PNG, MP4, etc.	JSON	Text analysis, Object analysis. Can process audio files for text-to-speech and event detectors.

Analysis:

The following artificial intelligence APIs are efficient and compatible with QLab for the shadow puppet show project: OpenAI (including DALL·E and ChatGPT), Google Cloud Vision, RunwayML, and Clarifai. These APIs are particularly well-suited for the project because they support real-time image and video processing, Also, AI content generation with an automated media management when integrated with a system like QLab. These tools could be scalable to allow real-time data analysis and creating automated responses based on user input(s). ChatGPT would be the most accessible and reliable for other students to use. This is because ChatGPT is widely known and has been discussed by other students in my courses. Making it easier for others to familiarize themselves with it.

Dependencies:

- python-osc: Library for OSC communication.
- os: Library for directory operations (folder path).

Instructions to Run:

1. Choose images/videos to provide feedback to the "AI" and start the program.
2. Modify the folder_path variable to point to your image directory.
3. Update osc_ip and osc_port to match your OSC server configuration (From the QLab settings).
4. Run the script.
5. Exit the program when done.

In Conclusion:

- In conclusion, there needs to be further AI refinement to better predict which images are most likely to be successful in triggering a cue. The integration of more complex user feedback, like ratings or behavioral data analysis. Possible inclusion of different quality of life features such as video support. I was unable to figure out how to program the code to trigger the 'GO1' function in the software, but I hope this feature can be implemented, if possible.

Libraries:

- OSC - This Python library implements Open Sound Control (OSC) server and client functionality, enabling seamless communication between applications. It provides various features for both server and client sides, allowing developers to integrate OSC messaging into their projects.
- OS - The OS module in Python facilitates interaction with the operating system, allowing for file operations and system management. It is a standard library in Python by providing essential tools for file handling and directory manipulation.
- Artificial Intelligence: The recommended artificial intelligent APIs are OpenAI (ChatGPT/DALL-E), Google Vision Cloud, RunawayML, and Clarifai. Which all offer there on as seen in the documentation.