Omniverse ACE Audio2Face Plugin

The ACE Audio2Face plugin allows Omniverse Audio2Face and Avatar Cloud Engine (ACE) to stream or burst blendshape coefficients and wave audio into Unreal Engine. The blendshape coefficients are sent to the Live Link interface to drive animation of MetaHuman or other face types. The wave audio data is received and submitted to the ISubmixBufferListener interface to be played simultaneously with the facial blendshape coefficients.

The plugin can operate in two modes when receiving blendshape coefficients: Streaming or Burst.

Basic Instructions in Unreal Engine

- Install the plugin by copying the files to an appropriate folder
  - For a packaged engine (such as one installed from the Epic Games Launcher) copy plugin somewhere under Engine\Plugins\Marketplace. Create the Marketplace folder if one doesn't already exist.
  - For a source engine copy plugin somewhere under Engine\Plugins\Runtime and build your engine.
- Enable the `NVIDIA Omniverse ACE: Audio2Face` plugin
- Open the Window > Virtual Production > Live Link tab
- Create a new `NVIDIA Omniverse LiveLink` Source
- Modify the `AnimBlueprint /Game/MetaHumans/Common/Face/Face_AnimBP` so the `LLink Face Subj` is set to `Audio2Face`
  - This may not be possible until a connection is made to the plugin with the `Audio2Face` subject name
- Open a map with a MetaHuman actor that includes a `LiveLink Skeletal Animation` component
- At this point everything should be setup correctly to receive blendshape and audio data

Testing with Python Script

A test Python script is provided which will stream or burst data to the plugin. It is located in the plugin: `/Test/simple_socket_sender.py` and is fully documented with many different command line arguments using `--help`.

Example:

- `>python simple_socket_sender.py -f 0` - this will BURST all of the data and it will be replayed at 30 FPS by the ACE plugin
- `>python simple_socket_sender.py -f 0.033` - this will stream the blendshape and audio data at 30 FPS, it will be replayed by the ACE plugin as it is received

Prepending Data with Size Verification

Add data sent over the sockets is prepended with an unsigned 64 bit integer (uint64) equal to the byte count of the data within the packet. If the data is a header, it's equal to the size of the following appended data. If the data is an ASCII JSON struct, it's the length of the string. This uint64 size is always *binary* (not *ASCII*).

Here's a Python example demonstrating how to send ANY data over the sockets, whether it's for Audio, Blendshapes, a header, or data:

```python
def send_with_validation(socket, raw_data, ascii_convert):
    # Q format provides a uint64
    verify_size = struct.pack("!Q", len(raw_data))
    send_data = verify_size
    if ascii_convert:
        send_data += bytes(raw_data, "ascii")
    else:
        send_data += raw_data
    socket.send(send_data)
```

Streaming Mode

The Streaming protocol allows an application like Audio2Face to send individual frames at whatever rate it wishes for them to be played back in Unreal. The plugin does no dedicated buffering of blendshape frames and simply submits them to the Live Link interface as soon as it receives them. Because of this the application doesn't need to provide a header for the frame stream, it can just send JSON formatted blendshape frames that look like the following ("Audio2Face" becomes the subject name within Unreal).

A Blendshape Frame:

```
{uint64-FrameByteCount}
{
    "Audio2Face": {
        "Body": {},
        "Facial": {
            "Names": [
                "EyeBlinkLeft",
                ...
                "HeadYaw"
            ],
            "Weights": [
                0.0,
                ...
                0.0
            ]
        }
    }
}
```

Note that the blendshape names conform to the ARKit Blendshape Coefficients

```
|Binary---------------|ASCII------------------|
{uint64-FrameByteCount}JSONFormattedFrame-ASCII
```

Burst Mode

The Burst Mode protocol is intended for a cloud application like ACE so that it can send all of the animation and audio data as soon as it is ready, rather than streaming it frame by frame. Because the intention is for animation and audio data to be synchronized, both streams require headers. The audio and animation playback won't begin until both streams have valid data.

Blendshape Header

The Blendshape header contains the characters "A2F" plus the playback Frames Per Second (FPS). A typical example would be `A2F:30` for a playback rate of 30 frames per second.

```
|Binary-----------|ASCII-----------|
{uint64-HeaderSize}A2F:{PlaybackFPS}
```

Blendshape Data

The blendshape data sent over the socket is the header followed immediately by all of the blendshape frames:

```
 ------------------ --------------------- ----- --------------------------- -----
|Blendshape Header| Blendshape Frame 0 | ... | Blendshape Frame Count-1 | EOS |
 ------------------ --------------------- ----- --------------------------- -----
```

Wave Audio Header

The wave audio contains the characters "WAVE" plus wave header information:

```
|Binary-----------|ASCII---------------------------------------------------------|
{uint64-HeaderSize}WAVE:{SamplesPerSecond}:{ChannelCount}:{BitsPerSample}:{Format}
```

Audio Format

```
1: PCM
3: IEEE Floating Point
```

Wave Audio Data

Wave Audio samples are grouped into separate chunks and sent with a header:

```
 ----------- --------------------- ----- ---------------------------- -----
|Wave Header| Wave Sample Chunk 0 | ... | Wave Sample Chunk Count-1 | EOS |
 ----------- --------------------- ----- ---------------------------- -----
```

Wave Sample Chunk

The wave audio data sent over the socket is the header followed immediately by chunks of wave sample chunks. Wave sample chunks consist of a uint64 size prepended to a number of wave samples.

The uint64 sample chunk size = `sample_count * bytes_per_sample`

```
|Binary----------------------------------------|
{uint64-sampleChunkSize}+sample0+sample1+sampleN
```

Ending the Stream

To help the ACE plugin reset the Audio and Blendshape synchronization the bursting client can send an "End of Stream" (EOS) packet. This is formatted as an ASCII string: `EOS`:

```
|Binary--|ASCII-|
{uint64-3}EOS
```

Socket Information

The connection to the plugin uses TCP/IP sockets. The plugin is listening on these ports:

- Blendshape Port: 12030
- Audio Port: 12031