Audio Processing in Unity

Maybe not metaverse, but…

- Prototyping ideas
- Simulating data
- Studying perception

Unity is a very popular tool.

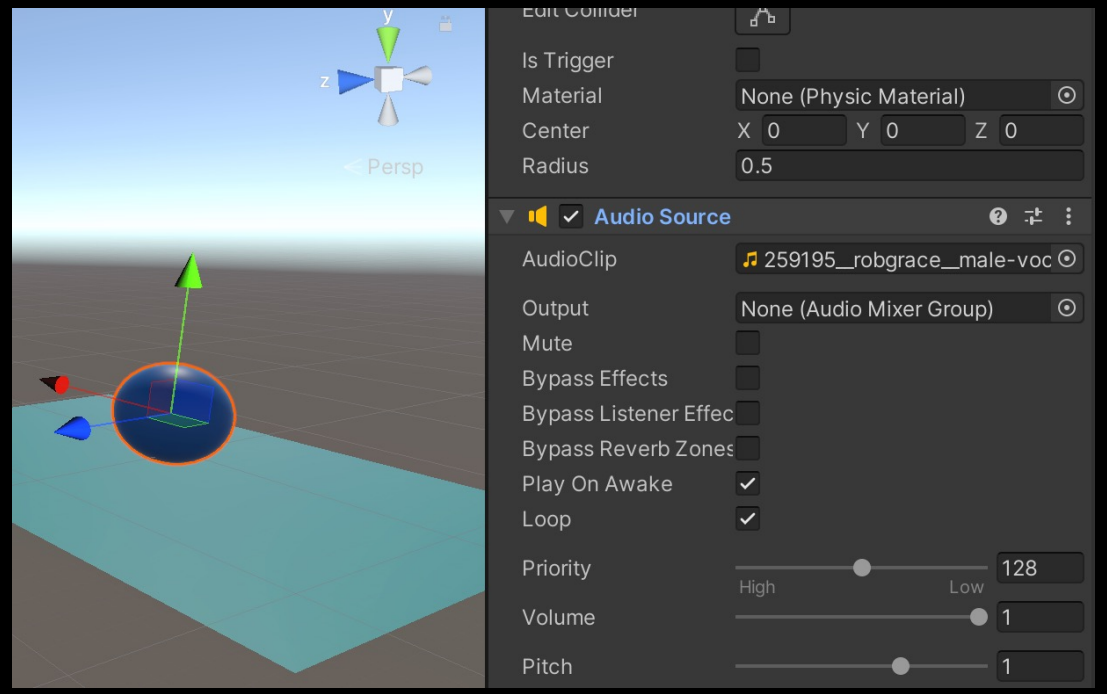
But…most tutorials don’t talk about audio.

Crash course: manipulating sound in Unity

1. Basic audio functionalities
2. OnAudioFilterRead()
3. Native Code
4. Unity Audio Plugins
5. Spatializers
6. ML Inference

1. Basic audio functionalities

- Unity: Object hierarchy, Scene & Game view, Inspector, Project directory, Console
- Adding audio source
- Unity Event Functions (3 main types)
- Scripting API - Making sound the element of the game logic
  - Access to editor parameters, and a bit more
  - https://docs.unity3d.com/ScriptReference/AudioSource.html
- Problem -> No access to actual audio data…



2. OnAudioFilterRead()

- Function to process audio within C# scripts:

```
private void OnAudioFilterRead(float[] data, int channels)
{
    // data -> current audio frame
    // channels -> number of audio input channels
    for (int i = 0; i < data.Length; i++)
    {
        // Example: reduce the volume by half
        data[i] *= 0.5f;
    }
}
```

- Separate audio thread
  - https://docs.unity3d.com/560/Documentation/Manual/ExecutionOrder.html
- For simple things sufficient, but performance not good

3. Native code

- Plugging in C++ code:
  - Write C++ code and use `extern "C"` keyword
  - Compile library (.bundle) and place it in Assets/Plugins folder
  - Use `[DllImport]` to declare external functions in C#

MyNativeAudioPlugin
main

MyNativeAudioPlugin 〉 My Mac (Intel)

C⁺ MyPlugin.cpp Build

MyNativeAudioPlugin 〉 C⁺ MyPlugin.cpp 〉 No Selection

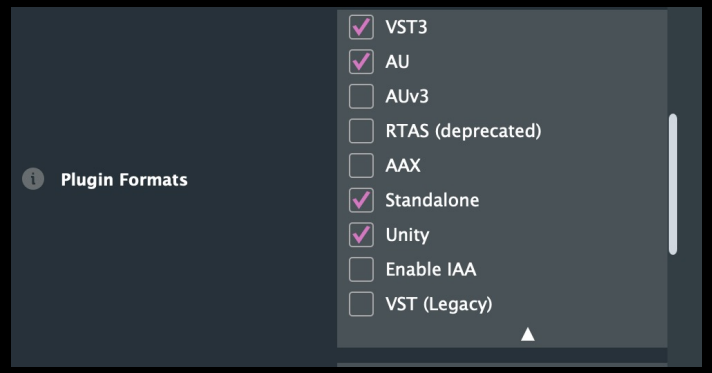
```cpp
extern "C" void applyGain(float* data, int numSamples, float gain)
{
    for (int i = 0; i < numSamples; ++i)
    {
        // Apply gain to each sample
        data[i] *= gain;
    }
}
```

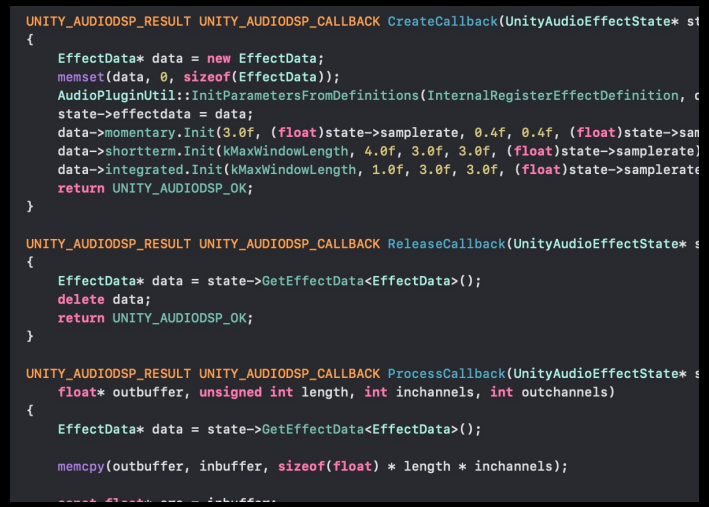
```csharp
[DllImport("MyNativeAudioPlugin")]
public static extern void applyGain(float[] data, int numSamples, float gain);
```

  - Use as a function in C#

4. Unity Audio Plugins

- Unity Mixer and Plugins
- Default Plugins
- “Demo” Plugins – Unity Audio SDK Library
- Possible to create your own plugins.
- Also possible – create plugins using JUCE and export to Unity format.





```
UNITY_AUDIODSP_RESULT UNITY_AUDIODSP_CALLBACK CreateCallback(UnityAudioEffectState* st
{
    EffectData* data = new EffectData;
    memset(data, 0, sizeof(EffectData));
    AudioPluginUtil::InitParametersFromDefinitions(InternalRegisterEffectDefinition, d
    state->effectdata = data;
    data->momentary.Init(3.0f, (float)state->samplerate, 0.4f, 0.4f, (float)state->sam
    data->shortterm.Init(kMaxWindowLength, 4.0f, 3.0f, 3.0f, (float)state->samplerate)
    data->integrated.Init(kMaxWindowLength, 1.0f, 3.0f, 3.0f, (float)state->samplerate
    return UNITY_AUDIODSP_OK;
}

UNITY_AUDIODSP_RESULT UNITY_AUDIODSP_CALLBACK ReleaseCallback(UnityAudioEffectState* s
{
    EffectData* data = state->GetEffectData<EffectData>();
    delete data;
    return UNITY_AUDIODSP_OK;
}

UNITY_AUDIODSP_RESULT UNITY_AUDIODSP_CALLBACK ProcessCallback(UnityAudioEffectState* s
    float* outbuffer, unsigned int length, int inchannels, int outchannels)
{
    EffectData* data = state->GetEffectData<EffectData>();

    memcpy(outbuffer, inbuffer, sizeof(float) * length * inchannels);
```

5. Spatializers

- Apart from `AudioSources` -> 1 `AudioListener` Object
  - Player control script
- Simple spatialization available by default (Spatial blend =1)
- Simple level decay curve also available
- Need HRTF-based spatialization -> Spatializer plugins
  - Steam Audio (https://valvesoftware.github.io/steam-audio/downloads.html)
  - Oculus Spatializer (https://developer.oculus.com/documentation/unity/audio-osp-unity-req-setup/)
  - 3DTI (https://github.com/3DTune-In/3dti_AudioToolkit_UnityWrapper)
  - …and more -> Overview

5. Spatializers

- Import custom package -> Project settings -> Audio -> Spatializer and Ambisonics decoder
  - Package = exported unity project
- Audio Source Editor -> Spatialize, Spatialize post effects
- Ambisonics audio source -> Import (Spatialize tag off)
- Controller script for point sources (directivity pattern, reflections, occlusion) and for ambisonics source.
- Custom HRTFs :
  - Place .sofa file in the Assets/StreamingAssets folder
  - Go to SteamAudio -> Settings and type sofa names
  - In play mode -> choose sofa files
  - Acoustic geometry -> tag meshes that should be used for the computation -> export active scene…

6. ML in Unity

- ML Agents (reinforcement learning and probably much more)
- Barracuda inference engine:
  - Window -> Package manager -> Add by name: com.unity.barracuda”
  - Train model/ download from internet
  - Tranfsorm pytorch/tensorflow format to ONNX format
  - Load model from Assets
  - Instantiate inference engine (worker)
  - Execute model

Closing remarks

- Overview about Unity audio functionalities
- I havent shown how to deploy the environments on Quest
- For basic stuff Unity is very approachable but for more complext things its hard to find information (being a developer helps)
- Troubleshooting is a part of the work