OpenAL on HPC

What is OpenAL?

OpenAL is a cross-platform 3D audio API that provides a standardized interface for programming 3D audio. It is designed to provide a more realistic spatialization of sound sources in 3D virtual environments, and is commonly used in video games, simulations, and other interactive applications. OpenAL is based on the OpenAL specification, which is an open standard for 3D audio.

Links:

Official Website

API documentation

Versions Available:

The following versions are available on the cluster:

- OpenAL –v1.20.1

How to load OpenAL?

To load OpenAL, use the following commands:

```
#Load the OpenAL module
module load openal/1.20.1
```

To verify if the module loaded correctly, use the following command,

```
# List all the module loaded
module list
```

It should list all the current module loaded. If openAl is loaded on a fresh environment, it should list GCC and openAL.

How to use OpenAL?

To demonstrate the usage of openAl library, copy the following code below:

```
#include <AL/al.h>
#include <AL/alc.h>
#include <iostream>
#include <fstream>
#include <vector>

int main() {
    // Initialize OpenAL
    ALCdevice* device = alcOpenDevice(nullptr);
    ALCcontext* context = alcCreateContext(device, nullptr);
    alcMakeContextCurrent(context);

    // Create a sound source
    ALuint source;
    alGenSources(1, &source);

    // Load the sound file into a buffer
    std::vector<char> bufferData;
    ALenum format;
    ALsizei freq;
    std::ifstream file("example.wav", std::ios::binary);
    file.seekg(0, std::ios::end);
    bufferData.resize(file.tellg());
    file.seekg(0, std::ios::beg);
    file.read(bufferData.data(), bufferData.size());
    file.close();
    ALuint buffer;
    alGenBuffers(1, &buffer);
    alBufferData(buffer, format, bufferData.data(), bufferData.size(),
freq);
```

```
    // Attach the buffer to the source
    alSourcei(source, AL_BUFFER, buffer);

    // Set the source's position
    alSource3f(source, AL_POSITION, 0.0f, 0.0f, 0.0f);

    // Play the sound
    alSourcePlay(source);

    // Wait for the sound to finish playing
    ALint state;
    do {
        alGetSourcei(source, AL_SOURCE_STATE, &state);
    } while (state == AL_PLAYING);

    // Clean up
    alDeleteSources(1, &source);
    alDeleteBuffers(1, &buffer);
    alcDestroyContext(context);
    alcCloseDevice(device);
    return 0;
}
```

This script uses the OpenAL library to play a sound file (in this case, "example.wav") in a Python script. The script first initializes OpenAL, creates a sound source, attaches a sound buffer to the source, sets the source's position, and plays the sound. After the sound finishes playing, the script cleans up the resources by destroying the source and buffer, and closing OpenAL.

To compile it, use the following sbatch script,

```
#!/bin/bash

#SBATCH --job-name=openal-example
#SBATCH --output=openal-example.out
#SBATCH --error=openal-example.err
#SBATCH --nodes=1
#SBATCH --cpus-per-task=12
#SBATCH -p main
#SBATCH --qos main
```

```
# load OpenAL module
module load openal/1.20.1

# Compile the program
g++ -o openal-example openal-example.cpp -lopenal -std=c++11

# Run the program
./openal-example
```

Submit the script to the scheduler,

```
sbatch script.sbatch
```

Where to find help?

If you are confused or need help at any point, please contact OIT at the following address.

https://ua-app01.ua.edu/researchComputingPortal/public/oitHelp