Intermediate report: Synthesis of sound with nodes

***Abstract*—This report provides an intermediate overview of an ongoing project focused on visual sound synthesis using nodes. The project aims to develop an intuitive and interactive tool that allows users to generate and manipulate sound waves visually, bypassing the need for traditional programming approaches. Each node in the system serves a specific purpose: sound generation, manipulation, or output to an audio device. This report discusses the project's goals, reviews related work in the field, describes progress made in application development, and outlines future directions for the project.**

I. Goals

The primary objective of this project is to design and implement a tool for computer sound synthesis based on a node-based interface. The tool seeks to simplify the process of learning and experimenting with sound synthesis by eliminating the need for coding skills. The key node categories, which provide the following features, are:

- sound generation Nodes: nodes capable of producing audio signals, such as sine wave generators or file input nodes;
- sound manipulation Nodes: tools for modifying audio signals, such as adders, multipliers, pass-through nodes, and various filters;
- components for directing audio to outputs, such as speakers, file writers, or other audio devices.

Users will be able to arrange nodes on a virtual canvas and connect them with wires, creating pathways for data flow between nodes. The application emphasizes usability and interactivity to improve the learning process and enable creativity in sound synthesis without requiring programming expertise.

II. Related work

A similar project is a project *AudioNodes* [1], which is used to create sounds, add effects, and compose original creations with a modular interface. This project aims to design and implement a similar tool, where users will be able to arrange and connect nodes to synthesize sound. An example graph, made with *AudioNodes*, is shown in figure 1.

AudioNodes is a digital audio workstation based on interconnectable modules, called nodes. Each AudioNode can:

- produce audio on its own and output it through its output port;
- take audio input and manipulate it (change gain or add reverb) and then output the modified audio;
- nodes that output the audio to the system or to an audio file when exporting.

The nodes in *AudioNodes* are connected with *audio connections* (blue) carrying a sampled audio signal [1]. More information about the tool can be found on the tool's documentation page [2]

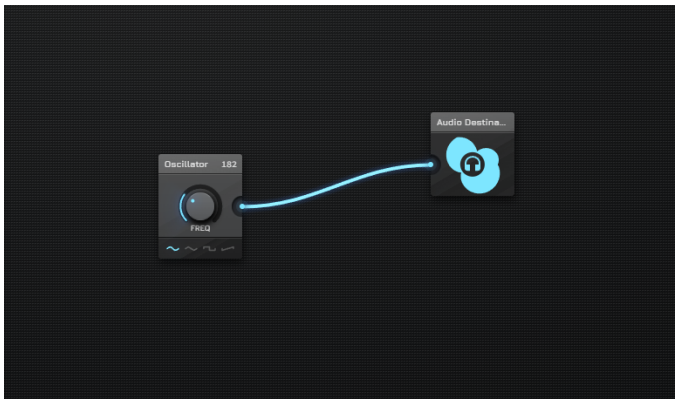

Fig. 1. Example graph, made with the tool *AudioNodes* [1]. The graph generates a sine wave and outputs it to the default audio output device.

III. Progress

The development process began by setting up the project environment for building and managing dependencies. I chose the *CMake* build system [3], a tool for managing and configuring C++ projects. To implement the graphical user interface (GUI), several external libraries were integrated into the project:

- *OpenGL* [4] for rendering the GUI. *Glad* [5] was utilized to provide the necessary bindings for *OpenGL*;
- *GLFW* [6] for creating windows and receiving input and events;
- *ImGui* [7] and an *ImGui* extension *Imgui-Node-Editor* [8] for GUI implementation for the node-based graph editor;
- *RtAudio* [9] for handling audio input and output, providing the necessary functionality to stream and manipulate sound in real time.

All dependencies were chosen to keep the application cross-platform and as lightweight as possible. Each dependency was installed and included in the project, and the functionalities were tested to make sure they work correctly.

After that, the project was organized into logical independent components, which will be developed independently. The codebase was separated into two parts:

- `AudioEditor` for editing and constructing graphs;
- `AudioEngine` for implementing nodes and generating, playing, and manipulating sound.

First, groundwork for the `AudioEditor` was implemented, where we created an empty window and initialized and rendered to it with *ImGui* [7]. Then, the *Imgui-Node-Editor* [8] library was initialized on that window, and an empty workspace was implemented, where the graph would be constructed. A simple node was drawn on the workspace to test *Imgui-Node-Editor* [8]. We also prepared the necessary callbacks for handling input and errors. The current window state is shown in figure 2.

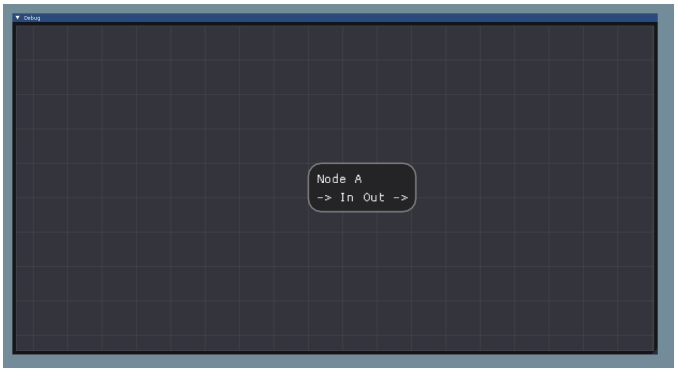

Fig. 2. Workspace for graph construction, implemented in *Imgui-Node-Editor* [8] with a simple example node drawn.

Then, we started designing and implementing the application programming interface (API) for *AudioNodes*. We defined the following classes:

- *AudioPin* for defining inputs and outputs;
- *AudioLink* for linking *AudioPins* together;
- *AudioNode* for sound generation, manipulation, and streaming.

After that, we began implementing the first two nodes:

- *SineOscillatorNode* for generating a sine wave;
- *LineOutNode* for outputting audio to the default device.

Finally, we tested its functionality by linking the two nodes together in code and producing a simple sound, shown in figure 3

```
void AudioEngine::initialize()
{
    SineOscilatorNode* sine = new SineOscilatorNode();
    LineOutNode* lineOut = new LineOutNode();
    audioNodes[0] = sine;
    audioNodes[1] = lineOut;

    AudioLink<AudioData*>* link = new AudioLink<AudioData*>();
    link->startPin = sine->audioOut;
    lineOut->audioIn->Link = link;
}
```

Fig. 3. Code used to test the API for the *AudioEngine* part of the application. It creates two *AudioNodes* and connects them with an *AudioLink* to generate a simple sound.

IV. Future work

The future work on this project will involve improving the *AudioEngine* API and making it easier to define new *AudioNodes* for different purposes. The *AudioEditor* will need to be implemented to support the construction and editing of the graph, and finally, more nodes will be implemented for more advanced sound generation and manipulation.

References

[1] Audio Nodes https://www.audionodes.com/
[2] Audio Nodes https://www.audionodes.com/docs/
[3] CMake https://cmake.org/
[4] OpenGL https://www.opengl.org/
[5] Glad https://github.com/Dav1dde/glad
[6] GLFW https://www.glfw.org/
[7] ImGui: API for GUI in C++ https://github.com/ocornut/imgui
[8] Imgui Node Editor: Implementation of a node editor using ImGui https://github.com/thedmd/imgui-node-editor
[9] RtAudio: API for realtime audio input/output in C++ https://www.music.mcgill.ca/ gary/rtaudio/