Project Club - A Music Visualizer Utilizing Ray Tracing

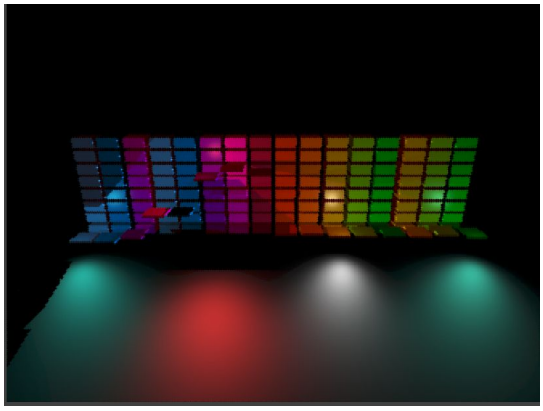
Authors:
Brian Mansfield
Alberto Scicali

Project Description:
This program is a music visualizer, using ray-tracing to render a live dynamic scene. Our objectives with this project were to build a ray tracer in Unity, integrate it with a dynamic music visualizer using Unity's Audio Analysis functionality, and optimize it for the best performance.

Approach
We used a standard ray tracing approach to achieve our results. Starting with a standard .mp3 file, our program plays the music live in the scene, and returns frequency data that is then used to change the objects in the scene. The scene is then ray-traced.

Results

We feel that we have hit our goal in some respects for this program. We took a dynamic scene and applied a series of optimizations to get decent performance from a live ray-traced rendering.
The scene itself can look quite good, as the screenshot below shows. At



A screenshot of our equalizer in action

Future Enhancements
In the future, we would like to extend the functionality of our ray tracer by improving the speed of it's output, preferably without sacrificing scene complexity. This would involve looking more in-depth at possible alternatives to our current ray-tracing algorithms, or possibly changing how our scene is set up. Due to how dynamic our scene is, future enhancements would most likely take the form of spatial bounding, separating the scene into areas of high- and low-importance to shave as many ray casting calls as possible.

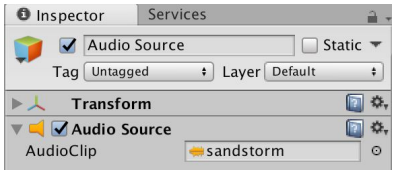
Users Guide

Program Description:

Our program is a live-render ray-traced music visualizer. The program uses Unity's built-in audio player and audio analysis functionality, combined with a custom built ray-tracer, to render a live scene. The scene consists of 14 channels/bands that represent a given group of frequencies supplied by the audio analysis module. The bands are supplemented by moving objects, coupled with several light sources moving with them to create a dynamic scene, which is quite a challenge to render correctly. Our program is an exploration into how best to optimize our ray tracer to achieve the best results.

Input

If the user wishes to run the application, then there is no input needed, and the application can be run as a standalone. If the user wishes to use their own music, or change the scene in any way, then the program will need to be opened in Unity, and the music file dropped into the project's hierarchy as an asset. Then the music file can be attached to the AudioSource object, by using the Audio Clip file explorer (see below)



This is all that is needed to add a custom song to the program.

Normal Output

If the program executes properly, it should produce a live ray-traced scene, with music playing in the background.

Exception Reports

Out of Bounds Exceptions - These can appear if the user may remove the dynamic objects from the scene, causing the music analyzer to try to access an empty array element.

Program Limitations

Currently, the major limitations on the program are resolution size and scene complexity. These can be adjusted by the user, but with caution. A resolution higher than 600x400, for example, could render the scene nearly unplayable. The program should generally not be run any higher than 300x200. The user is free to adjust the scene as well, but be warned that removing any objects may have adverse effects on the ability of the music analyzer to properly output frequency data.

Command Sequence

If the user is using the application form of the program, one can simply run the MacOS application. A window will appear asking for performance and screen size - for the best performance choose the smallest resolution and the "Fastest" option. Then click run. If the user

has decided to add their own music, or change the program in any way, then the program can be rebuilt using Unity's build process. The lowest resolution and maximum speed are recommended for best performance.

Author Information

Name: Brian Mansfield
Address: 47 Wilmer Street, Rochester NY
Telephone: 585-794-9998

Technical Documentation

Project Club

Program Description

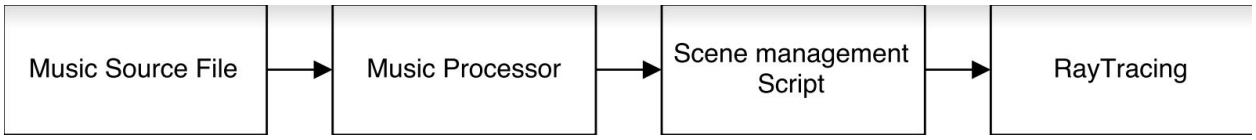
A ray tracer based EQ visualizer that takes in a music files as a source, processes it into a frequency spectrum, and then uses that data to manipulate objects in the scene along with the music.

Historical Development of Program and Current Status

Authors: Alberto Scicali, Brian Mansfield

RayTracer functionality: Alberto Scicali
Music spectrum analyzer and scene transformation: Brian Mansfield
Scene Staging: Alberto Scicali & Brian Mansfield

Overall System Structure



Description of Each Module

Music Processor Script
Author: Brian Mansfield, completed 5/1/2017
Calling Sequence: First, Continually
Purpose: Analyze the music input for frequency data
Data Items Modified: EQ - An array of frequency values

Description of Key Data Structures

- **2DTexture**

- To hold pixel information
- Similar to a matrix or 2D array (Holds color data for each pixel)
- 2D array
- The RayTracer.cs script accesses this data structure
- Declared like 15, instantiated line 23 in RayTracer.cs