Readme

CAO Chunxiao

2016.12.15

BST_Release is a sound transport demo with an implementation of the Bidirection Sound Transport algorithm. The implementation is not intended for practical use, and the code quality is very low. For any practical application, we recommend the programmer to implement his/her own sound renderer according to the algorithm description in our paper.

1 Building Instructions

1.1 System Requirements

For compilation:

- The project files are for Visual Studio 2013, and it should work for higher version of VS.
- This project uses `NtCreateKeyedEvent()`, `NtWaitForKeyedEvent()` and `NtReleaseKeyedEvent()` for thread synchronization. These are undocumented windows APIs. One would need `ntdll.lib` to link these functions successfully. `ntdll.lib` can be found in Windows Driver Kit(WDK) or Windows Driver Device Kit(DDK).

For program execution:

- Use Windows Vista or higher version of Windows.
- The graphic card must support OpenGL 4.0 or above.

The demo is originally written in Visual Studio 2013 on a 64-bit Windows 7 machine, which is the only environment we have tested.

1.2 Directory Structure

- `$(root_directory)`: GUI, graphic rendering and sound output.
  - `\BDPT`: Core algorithm of sound transport.
  - `\Corelibs`: Low-layer functions and access to other libraries. We used many third-part libraries in our program, including embree, FFTW, FreeImage, FreeType, GLEW, libogg and libvorbis. When updating these libraries, one should put related files into corresponding directories listed below:
    - `\env`: Runtime environment with required `.dll` files. A precompiled demo (for 64-bit Windows 7, using AVX instruction set) can be found in this directory.
    - `\libsrc`: `.lib` files for program linking.
  - `\Data`: Example scenes.

1.3 Buliding Options

All the options and parameters can be found in `main.hpp`, as well as their explanation.

2 User Guide

2.1 Scene Definition

The definition of a sound scene is a list of instructions. Each instruction may be followed by a parameter that is either a symbol, a file path, or a JSON object(one must follow the JSON format strictly or the program may crash during or after loading the scene). The supported instructions are listed below:

- `snd_stream` { Load an audio file.
  - `"name":` The name symbol of audio file.
  - `"path":` The path of audio file. Only `.wav` and `.ogg` files are supported. Note that the sample rate of the audio must correspond to the settings in `main.hpp`.
  - `"volume":` (Optional) volume multiplier.

  }
- `lobj [file path]` Load a 3D scene in Wavefront OBJ format. Object materials (including the sound materials) are defined in the `.mtl` file associated to the `.obj` file. The sound material is marked with the keyword `Kd_snd`, followed by the energy reflectivity in eight octave bands ranged from 62.5-8000Hz, like the example below:
  `Kd_snd 0.8 0.8 0.88 0.94 0.96 0.93 0.9 0.95`
- `cobj [name]` Create an object in the loaded `.obj` file.
- `cobj_all` Create all objects in the loaded `.obj` file.
- `lanim [file path]` Load an animation file. An animation file contains the definition of one or more paths that specifies the position and the direction of an object.
- `clight` { Create a light in the scene.
  - `"type": "point"` for point lights, `"ambient"` for ambient lights.
  - `"intensity":` Light color.
  - `"position":` Light position for point light.
  - `"UseAO":` (Optional, `true` by default) use ambient occlusion for ambient lights.

  }
- `snd_src` { Create a sound source in the scene.
  - `"stream":` The audio file used for this source.
  - `"position":` Source position.
  - `"path":` (Optional) Animation path name for this source.

  }
- `listener` { Listener configuration.
  - `"position":` Listener position.
  - `"path":` (Optional) Animation path name for listener.
  - `"volume":` (Optional) volume multiplier for listener.

  }

One could refer to the example scenes for a better understanding.

2.2 User Interface

Our program uses a GUI interface. The function of each control are explained below:

- "open": Open a new scene definition file.
- "play": Turn on/off real-time sound simulation. The user can hear the simulated result while exploring the scene.
- "record(RT)": Turn on/off real-time recording. The result will be saved into "record.wav" in the current working directory.
- "record(Path)": Move objects along the animation path and record the simulation result until all animations have ended. If there is no animation in the currect scene, the program will not stop recording. The user can force the program to stop recording by clicking the button again.
- "Frame Test": Render several consecutive frames to test the sound simulation. The number of consecutive frames is defined in `main.hpp`.
- "Show Impulse Response": When checked, the program will show the impulse response (green lines) and the sample distribution between bounces (green lines) of the selected source.
- "Record Energy Response": When checked, the program will save the energy response of every source after sound transport in each frame.

The user can change the camera direction by dragging the mouse over the window and move the camera around with the arrow key and mouse wheel.
If option `PERFORMANCE_TEST` is defined, the program will print the performance statistics in the console window. Not all these data are related to sound transport. One should look at `SecPerFrame` for the frame cost.