AudioFlow

Flow-influenced sound

Matthias Adorjan, BSc
e0927290@student.tuwien.ac.at

February 12, 2015

1 Introduction

Nowadays flow visualization is used in various application fields to make invisible flow patterns and flow structures visible. The automotive industry, for example, needs flow visualization to evaluate and optimize car designs. Since flow data can contain much more information than the direction of the flow, flow visualization can also be used for failure analysis. It reveals regions of high pressure or turbulant flow, which can indicate a source of error in car engines, for example. Another application area of flow visualization is the field of medical diagnosis. To gain a better understandig or to detect misfunctionalities of the human respiratory system, the motion of lungs during an inhale-exhale cycle can be visualized with the help of flow visualization [LEG+08].

However, we use flow datasets in a very different way. The main goal of our work is to analyze how flow data can influence audio playback. We use values from existing flow datasets to control the playback of an audio stream in our developed music player called *AudioFlow*. This application allows the user to extract flow data values from a loaded dataset with the help of simple drag and drop gestures. A rendered flow visualization makes it possible to select specific flow channel values. The extracted data values are then used to change the volume and playback speed of a background audio stream. Furthermore, it is possible to introduce a fading effect, which repeatedly decreases and increases the volume of the played audio file.

2 Implementation

Since portable devices like smartphones or tablets have become very popular these days, we decided to implement our idea in an Android application. *AudioFlow* targets an Android 4.3 device with OpenGL ES 3.0 support. We choose these minimum requirements, because full floating point texture support is needed to render the flow dataset in a proper way. This support was introduced with OpenGL ES 3.0 in August 2012, which is further available on devices with Android 4.3, or newer, and appropriate graphics hardware and drivers. We developed the program using the *IntelliJ IDEA*, *Android SDK 19* and a *HTC One* with an *Adreno 320* GPU. The user interface is completely realized using OpenGL ES 3.0 (see Section 2.1).

2.1 Interaction concept

In order to use a flow dataset as an input, one must be able to extract specific data from it. The following lines describe how we realized our user interface.

When searching for a suitable interaction method for this selection task, we found a device called Reactable [Rea]. It is an electronic musical instrument with a tangible user interface. There also exists a mobile version of Reactable. Figure 1 depicts the main concept of this instrument. It comes with a marker-based user interface which lets the user create music by placing markers, which can be real tangible or virtual objects, on a table or screen. This way of interaction deals as an inspiring example for our user interface development.

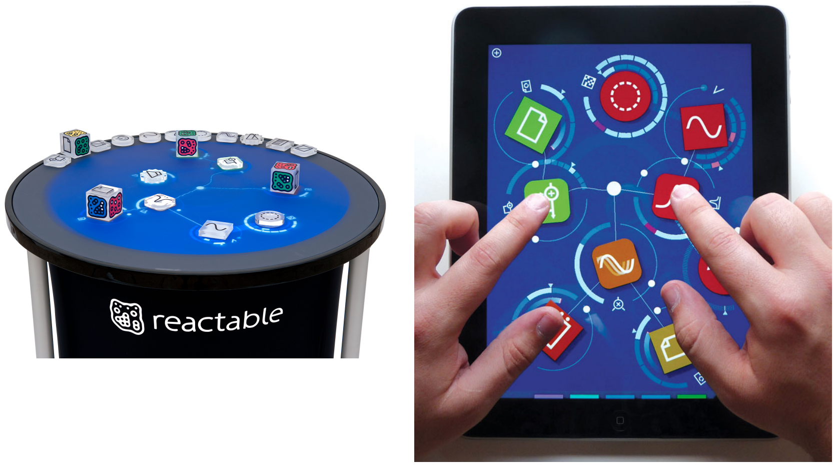

Figure 1: Reactable Live! (left) and Reactable Mobile (right) [Rea].

We implemented a similar marker-based input method. In contrast to the Reactable's interaction technique, which changes the audio output according to the spatial relationship between the virtual objects, we influence the output by analyzing the relationship between the markers and the underlying flow visualization. The flow data value at the marker's position on the currently displayed flow visualization is used to change a specific audio property. By combining different types of markers, each of them influencing another audio property, the audio playback can be manipulated in various ways (see Section 2.2). Figure 2 shows our realization of a marker-based music player. It can be seen that the background layer consists of a rendered flow visualization, and the foreground layer contains the interactive input objects. To select these objects/markers, the user has to press the finger on them and hold for about a second. After selection they can be dragged and dropped onto the visualization to changed the audio playback, its properties, or the used flow dataset/visualization. The menu at the bottom of the screen lists all available markers and is scrollable if the screen does not provide enough space for it. Furthermore, it is possible to switch between the represented flow data channels by simple horizontal-swiping over the currently displayed visualization.

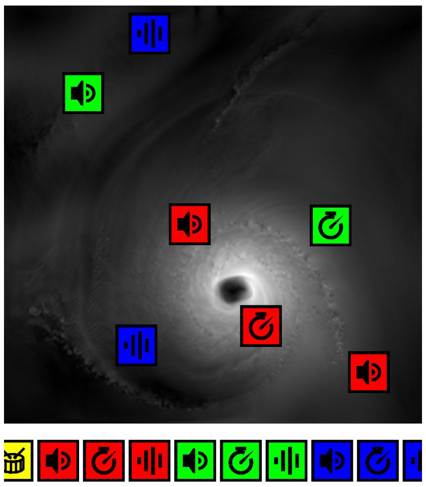

Figure 2: User interface of our created music player called AudioFlow.

2.2 Marker types and audio manipulation

In Section 2.1 the marker-based interaction concept has been introduced. This chapter now gives a detailed description of the different types of markers, which make up the integral part of this interaction concept.

2.2.1 Flow markers

The first type of markers reviewed are the so-called flow markers. They are used to change the currently loaded flow visualization, from which data values can be extracted to affect the audio output. When dropped into the scene (which means out of the menu area) the dragged marker disappears and the flow visualization is updated showing the dataset connected to the used flow marker. Our application comes with two flow datasets:

- the Block dataset created by R.W.C.P. Verstappen and A.E.P. Veldman of the university of Groningen (the Netherlands) [VV98]. It stores flow data representing a flow around a static block. Additionally to the velocity vectors, the dataset contains two scalar channels describing the flow pressure and its vorticity perpendicular to the flow slice.

- the Hurricane dataset originally created at the National Center for Atmospheric Research [Nat]. Its additional datasets contain scalar values describing the temperature in degrees celsius and the amount of liquid cloud water in the cell.

Figure 3 depicts the markers representing these two datasets.

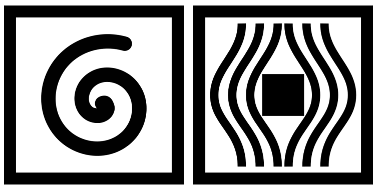

Figure 3: Marker representing the Hurricane dataset (left) and Block dataset (right).

2.2.2 Audio markers

Audio markers are used to change the audio file, whose playback is manipulated with the help of the background flow data. Similarly to flow markers, they have to be dropped into the scene to change the currently streamed sound track. We include two audio tracks in our application:

- a festive christmas sound obtained from `http://www.freesound.org/people/Setuniman/sounds/207637/`
- a drum loop retrieved from `http://www.freesound.org/people/Setuniman/sounds/207637/`

Figure 4 shows the markers representing these two sound tracks. They are colored yellow for easier identification.

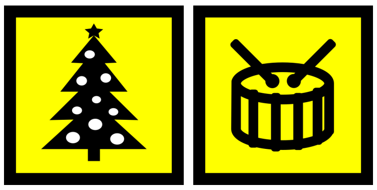

Figure 4: Markers representing a christmas sound (left) and a drum loop (right).

2.2.3 Channel markers

The third and last type of markers used in our application are so-called channel markers. They interact with the underlying flow visualization to change the playback of the currently active sound track. There exist three different channel markers:

- a volume marker affects the playback volume. The normalized flow channel value (between 0.0 and 1.0) is mapped directly onto the normalized volume value.
- a speed marker allows the user to adjust the playback rate of the currently active sound track. The normalized flow channel value is mapped into the range from 0.5 to 2.0 to halve or double playback speed.

- a fading markers allows the user to specify a fading time. The currently played sound track fades in and out repeatedly if one of these markers are placed onto the flow visualization.

These channel markers receive their input from three different flow channels (velocity magnitude, pressure/temperature, vorticity/cloud water amount). Therefore *AudioFlow* creates nine channel markers, three for each loaded flow channel. When a marker is dragged onto the flow visualization, the value of its corresponding flow channel at the marker's position on the visualization is extracted and used to change the audio playback according to the marker's audio manipulation property. To increase the app's usability, the flow visualization shows the channel which is used by the currently selected flow channel marker.

Figure 5 shows the available channel markers. The red markers use velocity magnitude values, blue markers use the pressure/temperature values and green markers use vorticity/cloud water amout measurements to adapt the audio playback.

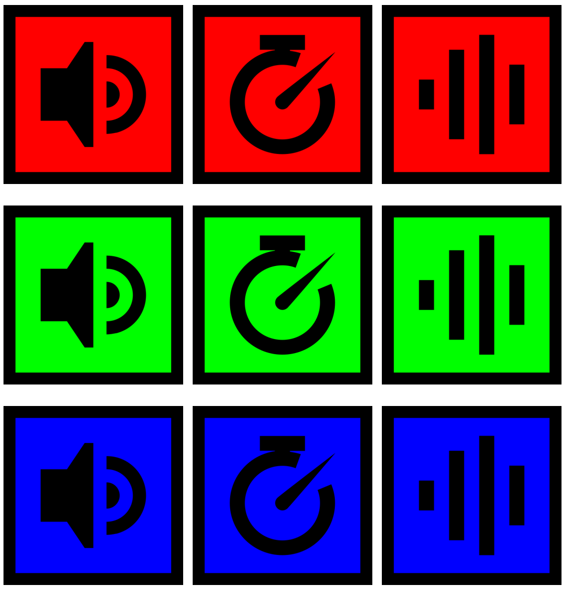

Figure 5: Markers used to manipulate the playback volume (left column), playback speed (middle column) and fading time (right column). The colors indicate different flow channels from which the input values are gathered.

2.3 Flow visualization

Additionally to the input markers, our implemented user interface contains a flow visualization in the background. It allows the user to explore the loaded flow dataset in order to pick specific information out of it. This section briefly explains details about the implementation of the flow dataset loading and the flow visualization rendering.

2.3.1 File format and loading

Our application supports flow datasets that are stored within two files. A file with a .gri extension stores the geometry of the dataset. It contains a description of the rectilinear grid on which the flow data is given. The first 40 bytes are the ascii header of the grid file. It describes the spatial dimension of the dataset, the number of stored flow channels and the number of timesteps. We ignore the number timesteps when loading the dataset, because we only need one timestep for our application.

The actual flow data is stored in a separate file with a .dat extension. The flow channels are stored interleaved, which means that each connected data block contains all flow data values for a given grid point defined in the grid file. For further information visit `https://www.cg.tuwien.ac.at/courses/Visualisierung/Angaben/Bsp2.html#Datensätze`.

To prepare the channel data for rendering, it has to be deinterleaved such that connected data blocks belong to the same flow channel. In order to generate correct textures out of the flow channel data, it has to be resampled from the rectilinear input grid onto a cartesian grid. Furthermore, a velocity magnitude channel is computed from the loaded velocity vectors to obtain additional visualizable information. The data loading and preprocessing is a time consuming task when done on a current smartphone like the HTC One. Therefore we decided to develop a small tool called *FlowResampler*, which does the preprocessing task in an offline process to further allow fast switching between different flow visualizations in our Android application. Table 1 compares the computation time needed to load the preprocessed files and the original files on the HTC One. It can be easily seen that we could reduce the loading time dramatically.

Dataset	**Resolution**	**State**	**.gri-file**	**.dat-file**
Hurricane	512x512	Original	0.05 seconds	28.68 seconds
		Preprocessed	0.05 seconds	1.75 seconds
Block	314x538	Original	0.02 seconds	20.74 seconds
		Preprocessed	0.02 seconds	1.19 seconds

Table 1: Loading time of the flow datasets Hurricane and Block.

2.3.2 Rendering

After loading the preprocessed data from the input files, the deinterleaved and resampled flow channels can be used to create 16bit half-float textures. We use half-float textures because they are filterable, whereas 32bit full-float textures are supported, but not filterable. The flow visualization is implemented using color coding. The maximum data value within a channel is assigned the color white, while the minimum value gets the color black. The colors for the values inbetween are calculated by using linear interpolation. To use the screen's space ideally we defined an orthogonal projection matrix which maps the flow visualization texture onto the screen, such that it fully covers the screen's width. The aspect ratio of the texture depends on the size of the loaded flow dataset. Figure 6 shows the visualization of two datasets, which are included in the application.

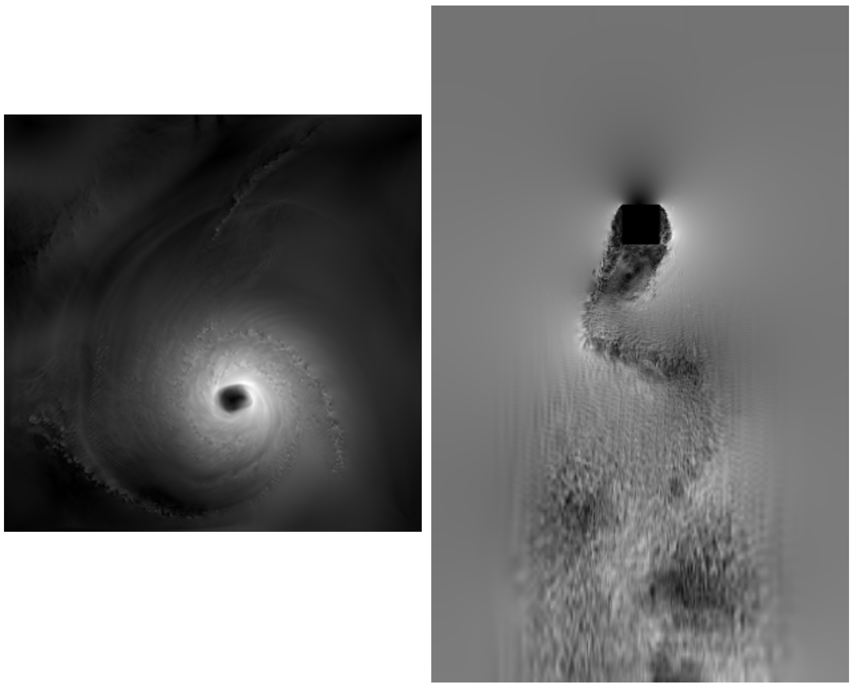

Figure 6: Visualization of the Hurricane dataset (left) and the Block dataset (right).

3 Conclusion and future work

The application field of flow visualization is wide spread. We decided to use this type of data visualization in a very different way. With our developed application, we show that flow data values can be mapped onto audio properties in various ways.

Further types of markers can be added to manipulate other audio properties or to introduce sound effects like the Doppler effect or reverberation. It is also imaginable to load multiple sound channels and play them simultaneously. The influencing flow data value then decides how many sound channels are actually played back. This can result in some sort of orchestral music, if each sound channel represents a different music instrument. To make the application look more pleasing, the flow visualization rendering can be tuned too, for example by using configurable transfer functions for color coding, or by implementing streamlines or glyph rendering.

References

[LEG⁺08] Robert S Laramee, Gordon Erlebacher, Christoph Garth, Tobias Schafhitzel, Holger Theisel, Xavier Tricoche, Tino Weinkauf, and Daniel Weiskopf. Applications of texture-based flow visualization. *Engineering Applications of Computational Fluid Mechanics*, 2(3):264–274, 2008.

[Nat] National Center for Atmospheric Research. Hurricane dataset. Retrieved February 03, 2015, from `https://www.cg.tuwien.ac.at/courses/`

Visualisierung/Angaben/Bsp2.html#Datens%C3%A4tze.

[Rea] Reactable Systems. Reactable. Retrieved January 30, 2015, from http://reactable.com/products/.

[VV98] RWCP Verstappen and AEP Veldman. Spectro-consistent discretization of Navier-Stokes: a challenge to RANS and LES. In *Floating, Flowing, Flying*, pages 163–179. Springer, 1998.