Interactive music visualization

Ondřej Kubelka
xkubelka@cslab.felk.cvut.cz

Department of Computer Science and Engineering
Czech Technical University,
Prague / Czech Republic

Abstract

This experimental work presents an application for music visualization. There are several ways how to solve this task. We can find two basic approaches to music visualization: real-time animation (Winamp, Cubic player - plugins), and preprocessed animation (3D Max studio, Softimage). Real-time animations are often composed simple objects modified in dependency of music. Preprocessed animations are prepared by the user according to his/her own notions about a specific music. Our application combines both ways described above. Implementation details and user interface are discussed in this paper.

Keyword: music

1. Introduction

When thinking on music visualization we deal with different characteristics like volume, mood, melody, instruments, tempo etc. that can be extracted from the audio file. Such features have to be mapped to visual properties of target rendered scene. The scene typically consists of objects with various colors, positions and other attributes. One of the main problems of the music visualization is how to map music characteristics to the parameters of visual objects. In this paper we introduce one application solving this problem.
Our application consists of three parts - sound analyzer, visualization module, and scene editor (Fig.1). Sound analyzer extracts the music parameters from audio files. Visualization module offers two modes of the scene rendering (real-time mode and full-rendering mode). Scene editor allows the user to create a scene composed of different objects. Sound analyzer works separately and prepares data in internal data format for the visualization module.
The basic information on the sound analyzer is presented in Chapter 2, the visualization module is described in Chapter 3 and the scene editor is presented in Chapter 4. Some implementation details are given in Chapter 5. Chapter 6 concludes the paper and brings several ideas for future extensions.

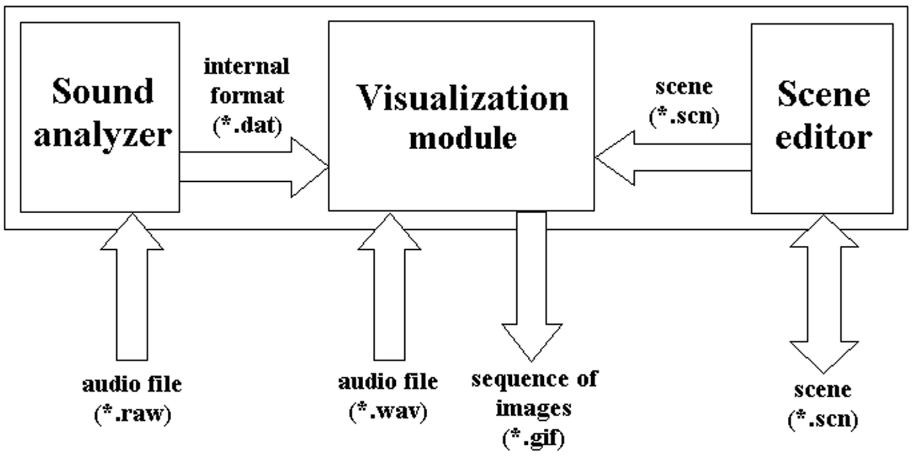

Figure 1: Basic scheme of the music visualization application

2. Sound analyzer

Audio files input to the sound analyzer (*.raw) where different features are extracted. The output consists of a set of 5 byte size records. Each record characterizes a music interval of 1/25 sec. duration. Each byte represents one music parameter. The meaning is as follows:

- volume - current music volume
- balance - expresses the current music stereo position (from left to right)
- dominant (solo) instrument position (melody)
- composition rate (tempo)
- estimation of the music feeling (mood)

Currently just volume and balance parameters are evaluated, while the analysis of the remaining three is under preparation. This is a task for other student group.

3. Visualization module

The visualization module processes the scene data and the sound data in internal data format. There appears a problem with the rendering speed of each single animation phase. It depends on the technical equipment and on the complexity of the visual scene. The optimal speed is to render one animation frame in 1/25 sec., i.e. in real-time. In most cases it is not possible to reach this speed for each frame in complex scenes. For this reason the visualization module works in two basic modes:

- real time mode (preview mode),
- full-render mode.

Using the real time visualization mode the scene is rendered simultaneously with the playback of the sound record from the audio file (*.wav). In the case that it is not possible to render the whole range of 25 frames per second some of the frames are left out. This mode serves as a preview and helps the user to estimate the design of the resulting animation quite well.

In the full-render mode, all the animation is rendered frame by frame and resulted images are saved on a disc. The final sequence can be further edited and combined with the original music in suitable applications (Adobe Premiere, Media Studio).

4. Scene Editor

The scene must be defined before the visualization is started. The scene is composed by specific author requirements and is changed according to the data created by the sound analyzer. Few basic scene objects were currently implemented in our application. Design of the objects was intended both on the appearance of the objects and the possibility of an easy modification of their parameters. Basic geometric objects (e.g. sphere, cube) were used in the first phase. These objects react to the change of music in such a way that they change their size, position, orientation, and color. It is quite complicated to create complex and nice looking scenes just from these simple objects. Due to these deficiencies other objects have been investigated. It was necessary to find objects complicated enough, but still controllable by limited number of parameters. Objects' characteristics should also offer the user adequate possibilities to express feelings from the music. We have found that one of suitable objects is the particle system. We have then extended the set of 3D objects not only by particles but also by coloured background and textured planar faces.

4.1 Animated objects

All the objects inside the scene are subjects of animation. Some of the objects can be present in the scene during the whole visualization some objects can dynamically appear only for a given time period. Therefore every single object contains a time interval in which is rendered and in which its parameters are modified. The following objects were implemented in the program:

- background
- textured face
- fountain (a particle system)

Background
Background object allows the user to change continuously the background color. The speed of the change from one color to other depends on the specified time interval. The background object is suitable for expressing lightning (short time interval), dawn or dusk.

Textured face
Currently just rectangular area with RGBA texture has been implemented. The object is included into the scene by adjusting its local origin situated in the middle of the plane. The alpha channel allows preparing nice effects when the several faces overlap.

Fountain
The name of this object is based on the model that inspired us. It is a particle system with the following parameters (see also Fig.2):

- middle value of the particles velocity vector
- particle source position
- gravitation vector
- bounding box for each individual particle
- parameters for color distribution of the particle system

The user can specify all of these. Each particle of the system has its own subset of parameters derived from the global one. Individual particle parameters are:

- current velocity vector
- current position
- (static) color

All of the particles are emitted from the same initial source position but with different initial velocity vector. The Gaussian distribution controls initial velocity vector of particles. The middle value of this distribution is the middle value of the particle system velocity vector. This value is influenced by the music parameters. The particle position is recomputed in dependence on the current velocity vector. This vector is modified by the gravitation. The color of a particle is computed from the parameters of Gaussian distribution of the whole particle system color.
The fountain is able to express the dynamics and the mood of the music. A dynamics is expressed by the motion of particles, while the mood is expressed by their colors. Several different object can be modeled by the fountain, e.g. fountains, waterfalls, falling leaves, fire etc.

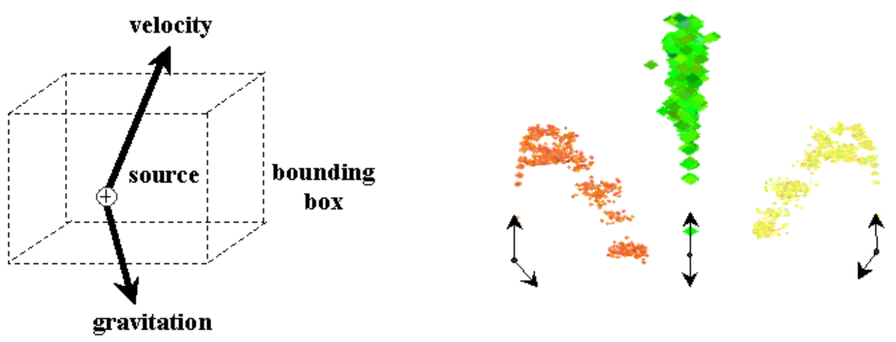

Figure 2: Left - Wire model of the fountain object
Right - Fountain objects with different parameters

4.2 GUI of the scene editor

The scene is created with the use of a graphic editor. The user interface (Fig.3) consists of the modeling space and the graphical representation of music parameters course. Through the editing the objects are temporarily substituted with wire model. The edited object is highlighted and its time interval is displayed in music parameters time course. The object parameters can be entered either by filling the dialogues or interactively using the mouse.

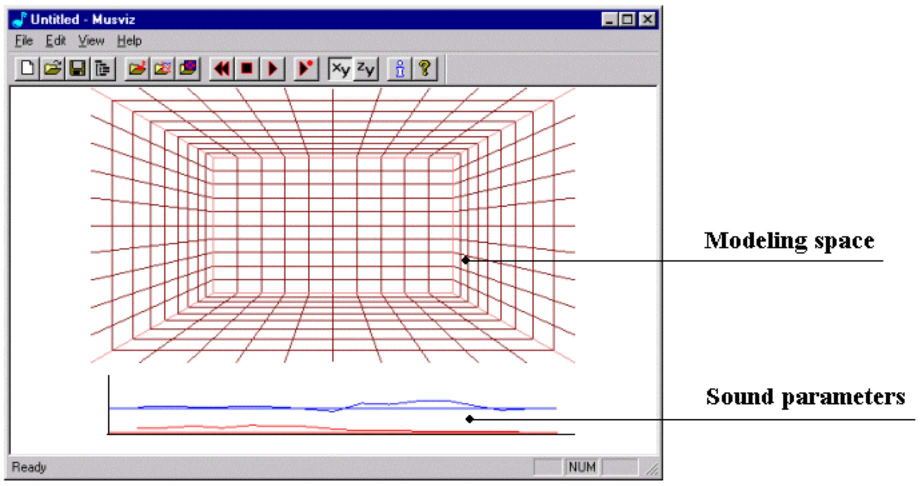

Figure 3: GUI of scene editor

5. Implementation

Implementation takes advantages of object oriented programming. There was created one generic class for all objects, which can be insert in the scene. Derived classes use virtual methods for own parameters adjusting, for parameters recount during the animation and for the object rendering. Due to this it is possible to insert new types of object in the system without any larger changes in the original source code. All objects of the scene are stored in dynamically linked list. Music parameters are stored in an array. The content of this array modifies the objects parameters. The parameters are modified only when the time of the animation is in the time interval of the object.
The system has been implemented for Windows NT platform in programming language C++. The standard graphical library OpenGL has been used for rendering.

6. Conclusion and future work

The particle systems proved to be good for the music visualization. They make possible to express plenty of objects that can be found in a nature. These dynamic objects were supplemented with objects that express the environments of the scene. The small number of objects offers to the user quiet a big range of expressing possibilities. Test animation (2-3min) has verified the system functions.
Other possibilities how to improve this application are:

- to add dynamics to static pictures (textured faces),
- to extract others music parameters and to link them with music parameters (there is an interesting research on using neuron networks for the mood extraction at our Department),
- to improve and to develop the enhanced fountain model (multi-source fountain, square sources), and to create the library of the fountain objects (object fire, object waterfall),
- to use of various image filters

The presented work has been completed by two students during one semester. The project continues and two new students will improve the program and extend its functionality.