ESP: A more robust real-time environment for expression synthesis

Jiayun Guo
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98105, U.S.

September 26, 2010

1 Introduction

The goal of Expression Synthesis Project(ESP) is to create a driving (pedals, wheel and display) interface for generating expressive performances interactively in real time from expressionless music files. The premise of ESP is that driving can serve as an effective metaphor for expressive music performance.[1] As expected, ESP is a complicated real-time interactive virtual environment. As such, the most challenging part of ESP is the organization of, and interaction between the software components carrying out multiple tasks, such as simultaneous processing of graphics, MIDI, music computation, car dynamics and driving control data streams in real-time.[2] The ESP system is designed and implemented using the SAI/MFSM framework which Alexandre R.J. Francois created. SAI(Software Architecture for Immersipresence) is an architectural framework for the design, analysis and implementation of interactive software systems. MFSM(Modular Flow Scheduling Middleware) is an open source, cross-platform architectural middleware, which provides a multi-threaded implementation of SAI's primitives.

2 Problem

To match my interests of software engineering, Prof. Elaine Chew assigns me the Expression Synthesis Project during my stay in USC MuCoaco lab. A prototype of ESP has been implemented by a doctoral student who has since graduated. But there are several problems with this PC-only prototype. Consequently, to make the software more robust and maintainable, I mainly work on optimizing the software and porting it from PC over to the Mac, in order to continue experiments with expressive performance using ESP. The original ESP prototype has the following problems:
1. Overuse of global variables. Global variables should be avoided when unnecessary, because of the following reasons: [3]

- non-locality: global variable can be read or modified by any part of the program which makes it hard to remember or reason about every possible use of it.
- no access control or constraint checking: A global variable can be get or set by any part of the program, and any rules regarding its use can be easily broken or forgotten. By extension, the lack of access control greatly hinders achieving security in situations where you may wish to run untrusted code(such as working with 3rd party plugins).
- concurrency issues: if globals can be accessed by multiple threads of execution, synchronization is necessary(and too-often neglected). When dynamically linking modules with globals, the composed system might not be thread-safe even if the two independent modules tested in dozens of different contexts were safe. In terms of ESP, the third problem is the severest, because the whole project is heavily multi-threaded.

2. Race condition in rendering module.
Once texture mapping is called, in other words when we try to load bmp file, the program will use so many threads of the system and as the program runs, the number of threads will also increase dramatically.
3. Libraries are out of date.
The Fsf creator Alex has updated the libraries from 0.7 to 0.8, but the ESP project is still using the old library.
4. The display module needs to be replaced.

We want to make use of GlutIO to make the project more robust.
5. Not platform-independent.
The original project can only runs in PC, and it is supported by a tremendous number of PC or visual C++ specific libraries.

3 Method

- As I observed, through the whole ESP project, the global variables are primarily used in three different ways: first, they are used to define input and output files; second, they are used in some physics computation in physics module; third, there are several extern global variables that are used in different modules.
  - For the global variables that deal with input and output files, I basically pass the files as parameters to different methods.
  - For the global variables that only appear in one module, especially the physics module, I change them as const.
  - For the global variables that define as extern and are used in different modules, I change them as nodes that can not only talk to each module, but also accessible through source.
- In order to make GlutIO module work in the whole project, there is some modification of source code needed to be done, for example, considering the concept of singleton, I need to replace all direct use of the g_pSystem pointer with a call to CSystem::GetInstance(), which restricts the instantiation of a class to one object.
- We want to build a project that is cross-platform, in order for continuing future experiments with expressive performance. The first problem we meet is the endianness platform difference. We use some different types like unsigned long, unsigned int, unsigned char, etc to store midi message. In order to read the standard midi file successfully, we have to compare and figure out all the size difference between mac and PC, then fix the one that causes problem. It turns out that the problem comes from reading the header file of the midi, we use unsigned long to store the header message, but unsigned long has different size in PC and mac.

4 Results

We have the updated working version of the ESP project on PC by the end of first half of my research, for the second half of my research we updated the new project with the midi module which is compatible in mac, but time limited, we ported the project from PC to mac in vain.

5 Discussion and Future Work

Software engineering is a systematic and disciplined approach to developing software. It applies both computer science and engineering principles and practices to the creation, operation, and maintenance of software systems. Updating and maintaining is a significant part of software engineering, so this summer experience of software engineering gives me a sketch of how the working cycle in industry is and how important it is to get the code correct and clean the first time. Though the ESP project is quite impressive now, we still have a lot of future work.

- First, there is still memory leak in rendering module.
- Second, the joystick can not replace the gamepad module right now, though we implemented joystick support in GlutIOModule.
- Third, the read and write file function needs to connect with the new joystick.

6 References

[1] Elaine Chew, Alexandre Franois, Jie Liu, Aaron Yang. ESP: A Driving Interface for Expression Synthesis. In *the 2005 International Conference on New Interfaces for Musical Expression (NIME05), Vancouver, BC, Canada.*
[2] Jie Liu, Elaine Chew and Alexadre Francois. From Driving to Expressive Music Performance: Ensuring Tempo Smoothness. In *ACE 2006.*
[3] Global Variables are bad. http://c2.com/cgi/wiki?GlobalVariablesAreBad