TJHSST Senior Research Project
Computer Systems Lab
Music Editing/Compsition Software
2006-2007

Patrick Mutchler

January 16, 2007

Abstract

Current music editing software is expensive, complicated, and can be a poor teaching tool. A new, free software designed for amateur composers and music students requires a less powerful editing system and can incorporate learning tools to aid teachers in teaching music theory to students.

Keywords: Editing, teaching

1 Introduction

1.1 Rationale

The two leading compositional softwares, Finale and Sibelius, are programs designed to cater towards professional composers and trained musicians. This target audience leads to significant overcomplexities (512th notes for example) for music theory students who do not require all of the capabilities of profesional software. Another type of compositonal software that exists today focuses less on editing capabilities and more on the attractiveness of computer created scores. No effort has been made to create software that caters towards students who need a simple program to write out music using a computer.

1.2 Purpose

This project is the creation of a music editing software designed for students and amatuer composers who desire a simple and free program to meet their composing needs. The specific, large scale goals for this project are:

1. A working and effective music printing system that prints out music in an appealing fashion.

2. A working and effective graphical user interface for inputing and creating music within the running program.

3. A working and effective system of teaching tools including chord charts and chord patterns.

The research aspect of this project is small due to the nature of my project. I have researched existing editing programs to understand the nessecary components of a good program but the majority of my research is in the abstract field of using software to aid in teaching and learning.

2 Background

2.1 Existing Software

Existing editing software falls into two major catagories: compositional software and copying software with a focus on visual appeal.

Finale and Sibelius are two examples of compositional software. These programs are aids to composers to make writing music easier and nothing more. These programs do not contain any teaching tools and their help menus are all but useless. I have used some of the Sibelius interface as a model, but without so many complexities. For an example of a typical Sibelius interface see figure 1.

Lilypond is a program that focuses on music copying and making computer printed scores as appealing as hand written scores. Lilypond, however, cannot be used to create or edit scores quickly or easily due to a complicated file input system. Lilypond does not relate to my software at all, but is an example of software being developed in this field for other purposes. For an

example of a piece printed using Lilypond see figure 2. Pay attention to the additonal capabilties using this output method despite a multiple page file input that resembles hyroglphics.

3 Development

3.1 Goals

The main goals for this project for it to be deemed sucessful are a working output system that can handle simple music and a working GUI input system that can handle simple melodies. Beyond these goals, I also want to include as many teaching tools as possible into the program but they are not required for a sucessful release.

3.2 Development Method and Procedure

The development model being used for this project is a Staged Delivery. This model provides for multiple releases of software as the project progresses. For example, each quarter I have completed a major release and I have completed a small release every several weeks. This allows me to work on a basic shell of the software that improves as the project progresses. Also, this allows for effective record of the progression of the project.

The development process of this project is being broken down into four sections, one per quarter. Each of these sections will focus on a different aspect of the program until it it is evolved enough to be called a completed project. The goals for each quarter are as follows:

1. 1st Quarter - version 0.x: Create and implement a text based input and a terminal based output system as well as design the basic architecture of the final product.

2. 2nd Quarter - version 1.x: Create and implement a graphical output to replace the text based output. This will be the final output system for the completed product.

3. 3rd Quarter - version 2.x: Create and implement a graphical user input system to replace the text based input system. This will be the final input system for the completed product.

4. 4th Quarter - version 3.x: Create several teaching and learning aids. This is the bulk of the research material but is not completely nessecary for the final release.

3.3 Design

There are three class files used to run this program. The first, named Muse followed by the version number, is the program run by the user that creates the other panels. The second, named Printer followed by the version number, is a panel that deals with the actual printout of the music once the music has been delivered to it in a matrix. The third, named GUI followed by the version number, is a panel that deals with the GUI input of the music and delivers the information to the Printer panel.

3.3.1 Muse

Currently, the only methods within this program deal with the file input of the music. Once the file input has been done away with, this program will change into a pathway for transferring information between GUI and Printer.

3.3.2 Printer

Printer is a much more complicated class at this point. The major methods are described as follows:

1. PrintScore() loops over each measure of music and prints each measure using PrintMeasure() and adjusts the x position of the writing tool.

2. PrintMeasure() loops over each note in the measure and calls ParseNote() on the text describing the note and adjusts the x position of the writing tool.

3. ParseNote() takes a text input describing a note (for example c+, notates a csharp4) and calls PrintNote() on this parsed note and adjusts the x position of the writing tool.

4. PrintNote() takes a parsed note input and actually prints it to the screen based on its x position, its y position determined by its pitch, and its duration.

3.3.3 GUI

Currently, the GUI file does not exists but will be implemented during the third quarter.

4 Quality Assessment

4.1 Testing Characteristics

The most imporant feature to be tested within this project is accuracy and precision. I must be sure that there are no errors in the input/ouput system that will hinder the user. Every input should correspond with a correct output. No mistakes will be allowed here as this is a fundamental part of the program.

In a more qualitative process, the usablilty and intuitiveness of the interface will also be tested. This project is directed towards amatuer composers and students who cannot be expected to deal with complicated interfaces, something I want to avoid.

4.2 Testing Procedure

Because this project does not contain any algorithms based on randomness or any particularly complicated algorithms at all, testing does not need to be particularly rigorous. However, a two part testing method will be used throughout the development process to assure quality.

First, I will be testing various inputs at each stage of development to make sure that the input/output system is fuctioning properly. Any errors within these systems can be devastating to the project if they are not dealt with promptly. With each major change to the project I test anything that I believe can go wrong. For example, when a key signature printer was implemented I tested each possible key signature to make sure that they were all working correctly.

Also, I will be using the other people in this lab as a testing resource. Because my project is aimed towards amateurs it needs to be easy to use and understandable to an untrainer person. With each major change to the input I will call on my peers to review the system based on understandabilty and ease of use.

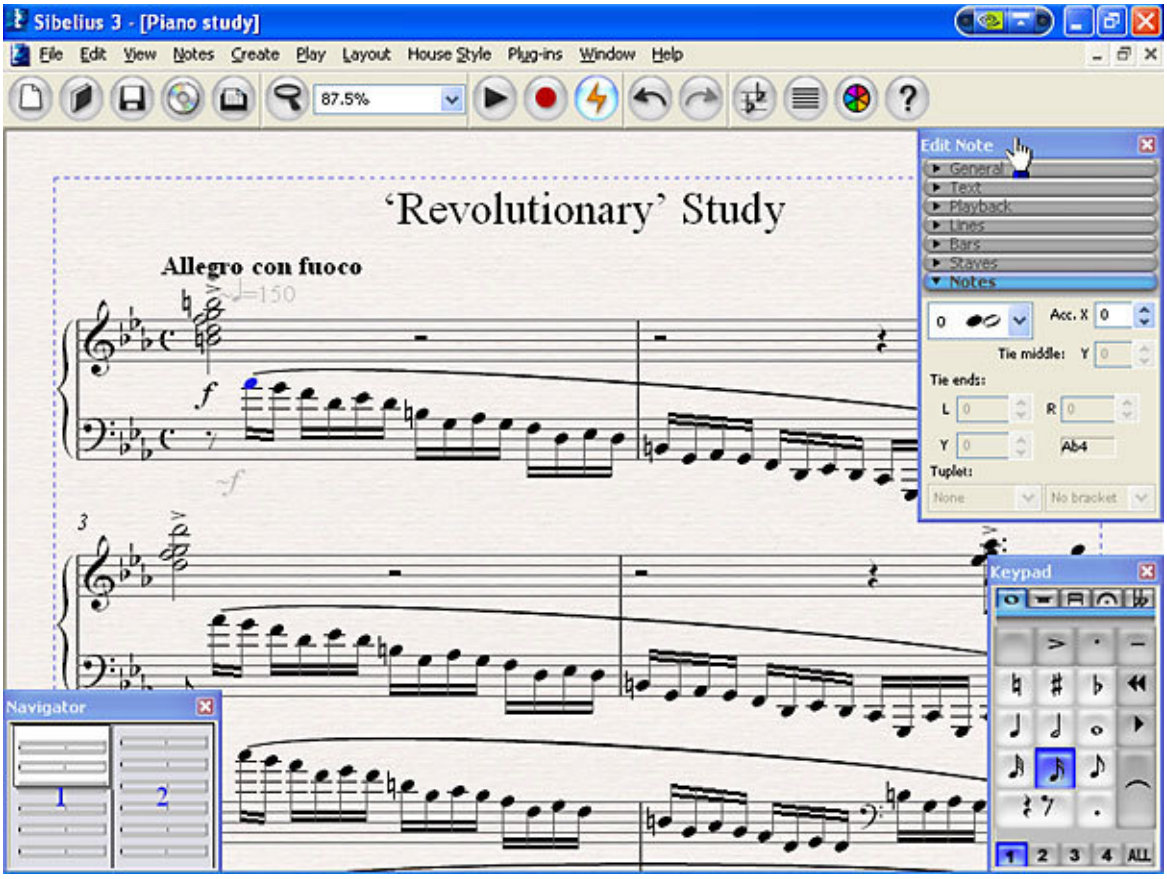

Figure 1: Sibelius

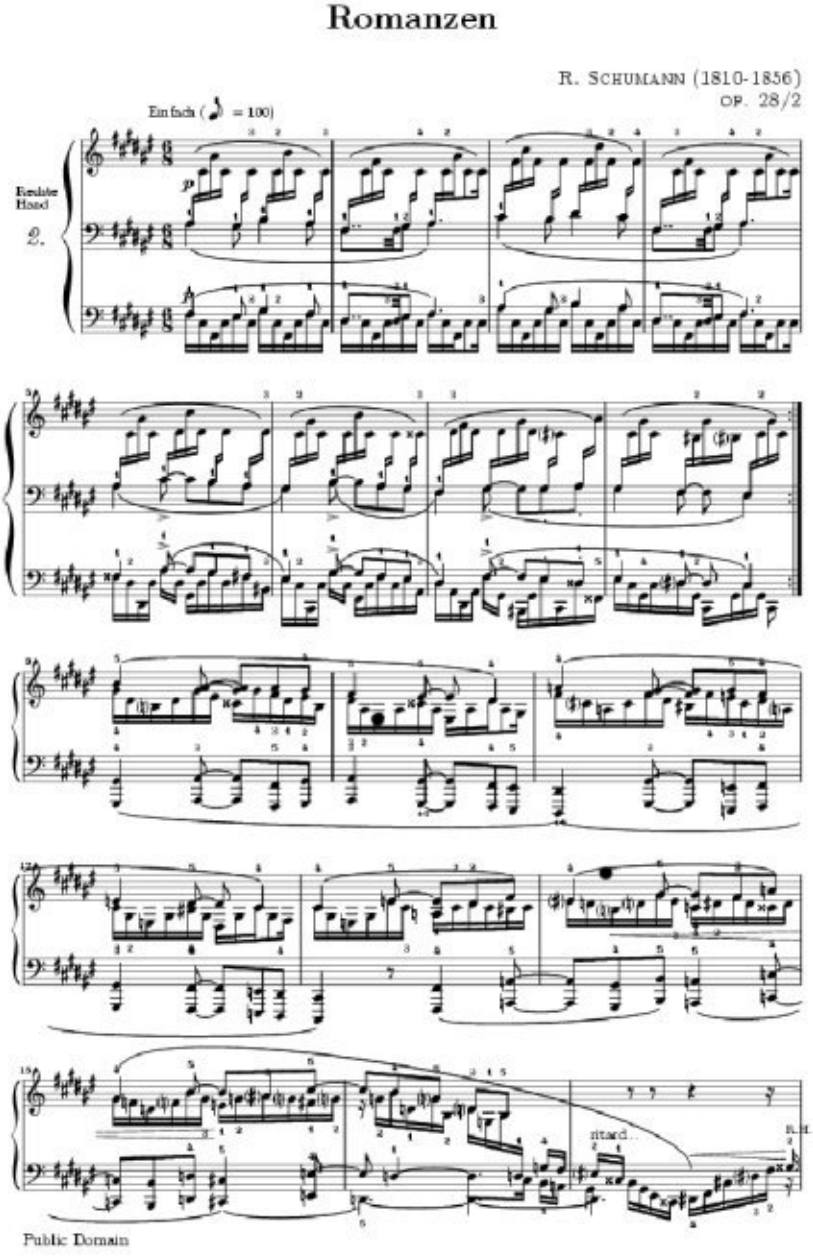

Figure 2: Lilypond

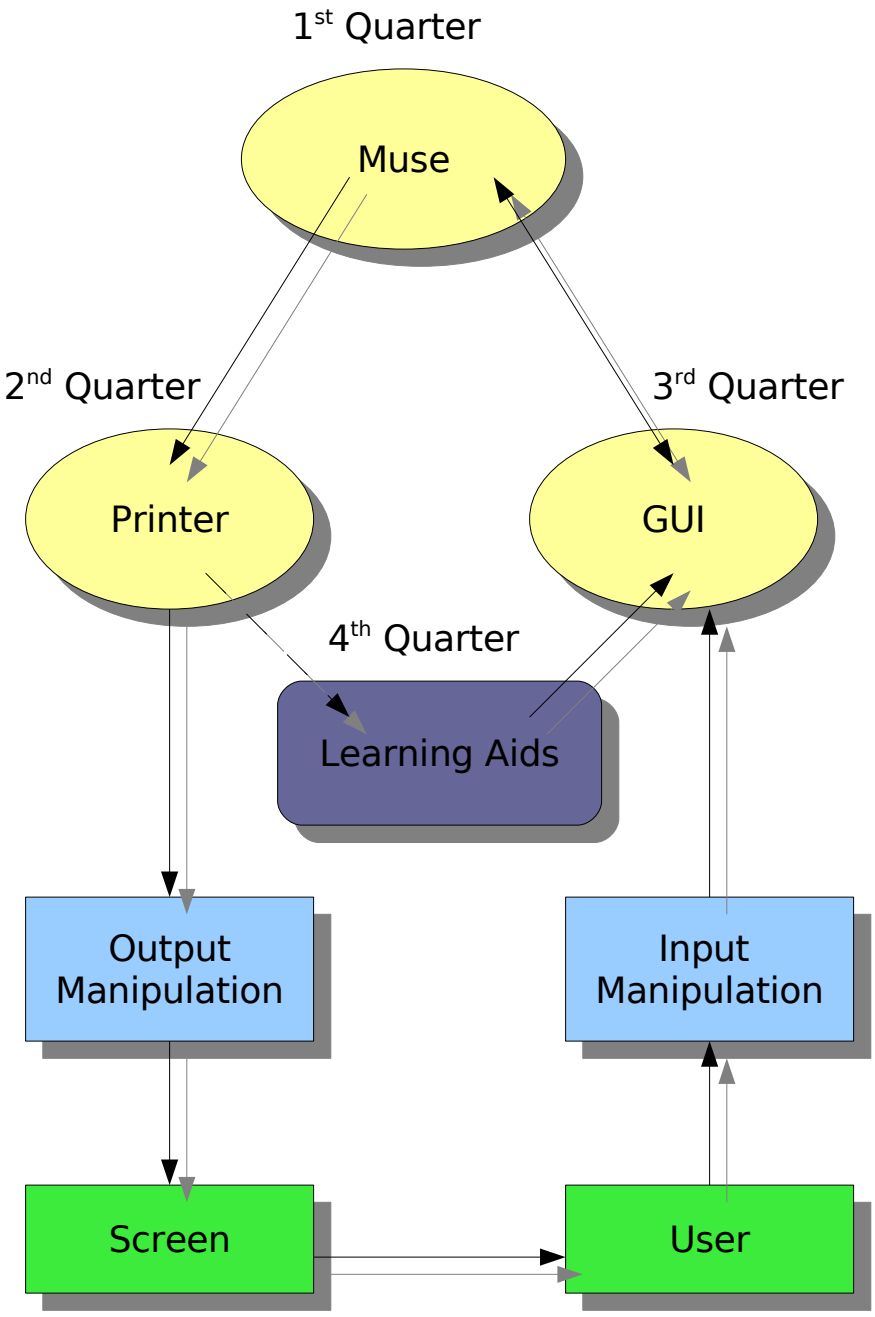

Figure 3: Architecture System