TJHSST Senior Research Project Computer Systems Lab Composition Software With a Focus on Teaching 2006-2007

Patrick Mutchler

June 5, 2007

Abstract

With the expansion of the Internet, more people are learning new subjects without the aid of a human teacher. The target audience of existing compositon software is trained musicians rather than students. This project is the development of a Compositon program that teaches students the basics of composing without the need of a teacher. The primary scope of this project bridges the studies of music, programming, and teaching.

Keywords: Internet, teaching, composition, students

Introduction

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 or students who wish to learn music theory without the aid of a teacher.

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 as well as providing tools to teach the user basic composition techniques. A professional program, Sibelius, will be used as a model for a simplified, learning based program. 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.
4. A series of lessons designed to walk the user through the basics of composing.

The research element of this project is transforming an existing, professional program into a learning tool for students. I will research the advances being made in internet based learning without a teacher and do my best to incorportate these ideas into a method of teaching composition. These programs are aids to composers to help make writing music easier, rather than teaching.

Background

Existing Composition Software

Existing editing software falls into two major catagories: compositional software and copying software

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 contain few tools for untrained users and do not help a student learn to compose without a teacher or textbook. 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 hyroglyphics.

eLearning

With the expansion of interent and self learning, there is a greater need for integrated functional software and teaching tools. This project will be able to be used by students studying composition without a teacher or textbook. The project will marry an effective compositional software with effective teaching tools.

In a presentation given by four professors at the Florida Gulf Coast University, three major requirements are set up for effective online learning programs: Instructional Design, Course Management, and Visual Design. Instructional Design means that the program must be designed to teach a student, setting up progressive goals and assignments for the student. Course Management builds off this idea, making each task large enough in scope to teach the student, but small enough that something can be accomplished in one session. Finally, Visual Design requires that the student be able to navigate the program easily and intuitively, rather than feel lost within the program.

Development

Exercising your rights can never be used as evidence to justify a search or detention. In

Goals

The main goals for this project for it to be deemed successful are a working output system that can handle simple music, a working GUI input system that can handle simple melodies, and a series of lessons and teaching tools that help the user learn composing as they use the program.

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 quarter contains the bulk of the research material.

Design

The original goal of the project was to create a program that could be used by amatuer composers to learn basic compositional techniques without the aid of a teacher.

There are four 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. The fourth, Teacher, is a frame containing all of the learning tools and lessons available to the user. See figure 3 for a outline of the design.

Muse

The purpose of Muse is simply to provide an interaction between Printer and GUI as well as the creation of these two objects.

Printer

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

1. addNote() takes a string input from the GUI class and adds it to the score matrix. This allows the GUI class to update the data that is to be printed.

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

3. 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.

4. 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.

5. 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.

GUI

1. createButtons() creates all the nessecary buttons on the GUI panel.
2. createListeners() creates the methods that will be called when each button is pressed.
3. callPrinter() takes a string, manipulated by the user through the GUI, and passes it to the Printer class to be added to the score matrix and eventually displayed.

Teacher

1. createButtons() creates all the nessecary buttons on the Teacher panel for the user to select lessons.
2. createListeners() creates the methods that will be called when each button is pressed. These buttons print out the appropriate lesson when the user presses the button.

Quality Assessment

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.

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. I ask each tester to grade the program in several categories, as well as providing suggestions for improving the program.

Results and Discussion

The original goal of the project was to create a program that could be used by amatuer composers to learn basic compositional techniques without the aid of a teacher.

Results

The completed project was successful at teaching an untrained user, my sister, to write a simple melody based on several basic ideas and rules. The lesson system alllowed her to first learn the interface and then the nessecary rules for composition.

What I Have Learned

Unfortunately, creating a functional composition program and creating learning tools is simply too vast of a task for a single student working five hours a week to complete. Given more time and a larger team, this could become a very useful piece of software for the general public.

As a consequence of taking on a large coding project, I have learned many lessons about consistancy of style, commenting, readable and understandable code, and design. These lessons will be extremely useful to me in the future even if learning them may have hurt the overall success of the software.

References

[1] Kevin Kruse, Gagne's Nine Events of Instruction: An Introduction, http://www.e-learningguru.com/articles/art3_3.htm

[2] Harriett G. Bohannon, Peggy Bradley, Terry Dugas, Joan Glacken, Design Principles for Online Instruction, http://library.fgcu.edu/Conferences/infostrategies00/presentations/Design%20Principles.htm

[3] Professional eLearning Designer's Association, http://www.peldaglobal.com/site/resources/index.php

[4] ClassLeader, www.classleader.com

[5] M. David Merrill, First Principles of Instruction, www.indiana.edu

[6] Robert Gagne, Nine Events of Instruction, http://ide.ed.psu.edu/idde/9events.htm

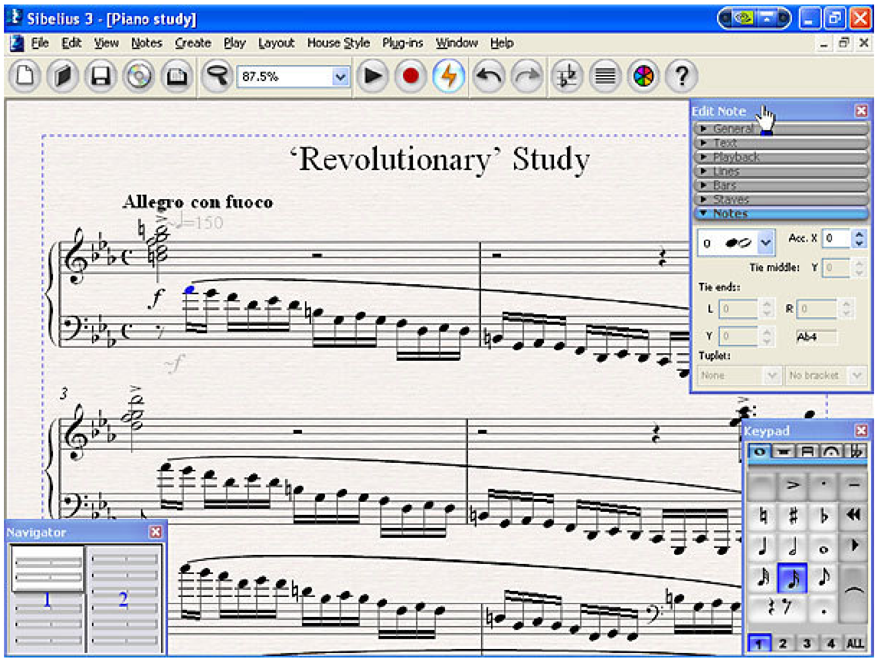


Figure 1: Sibelius

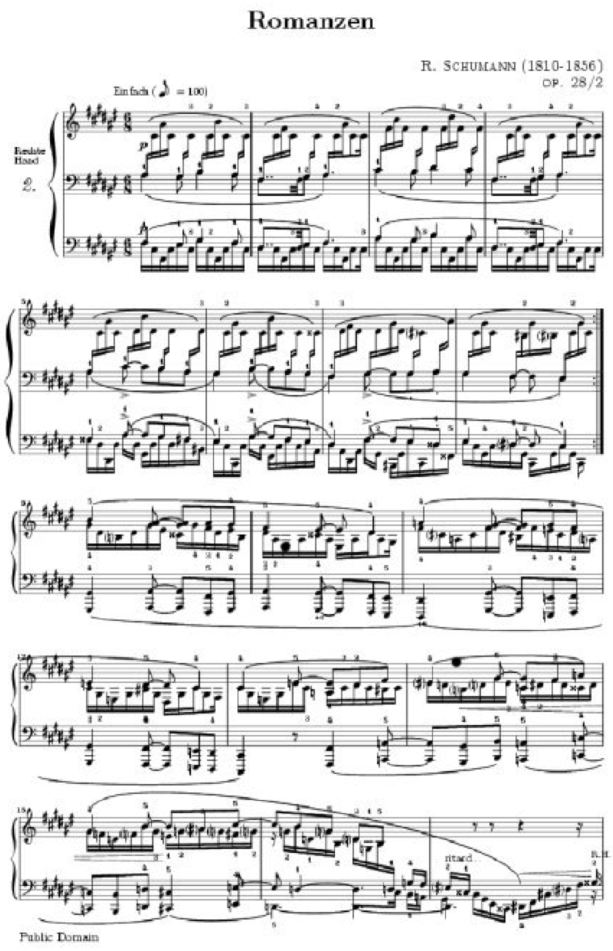


Figure 2: Lilypond

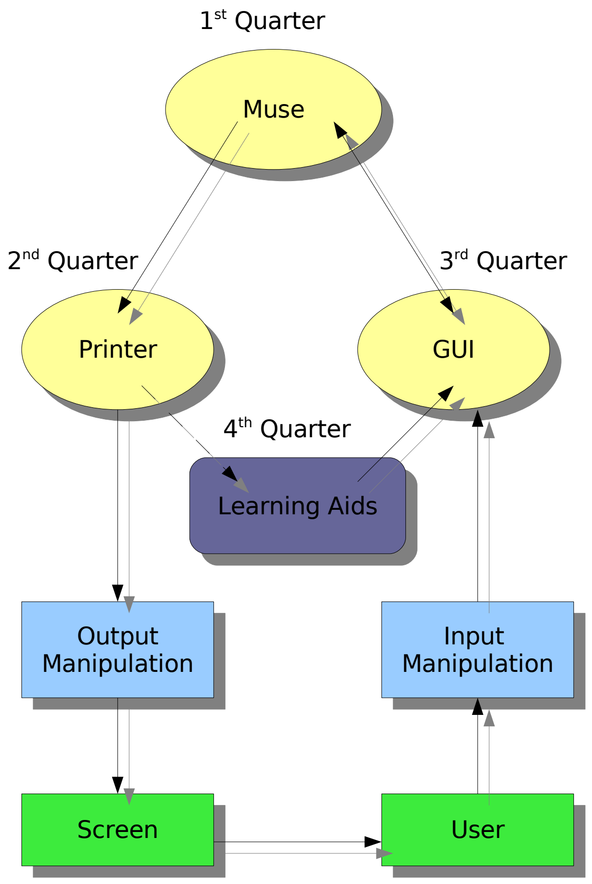
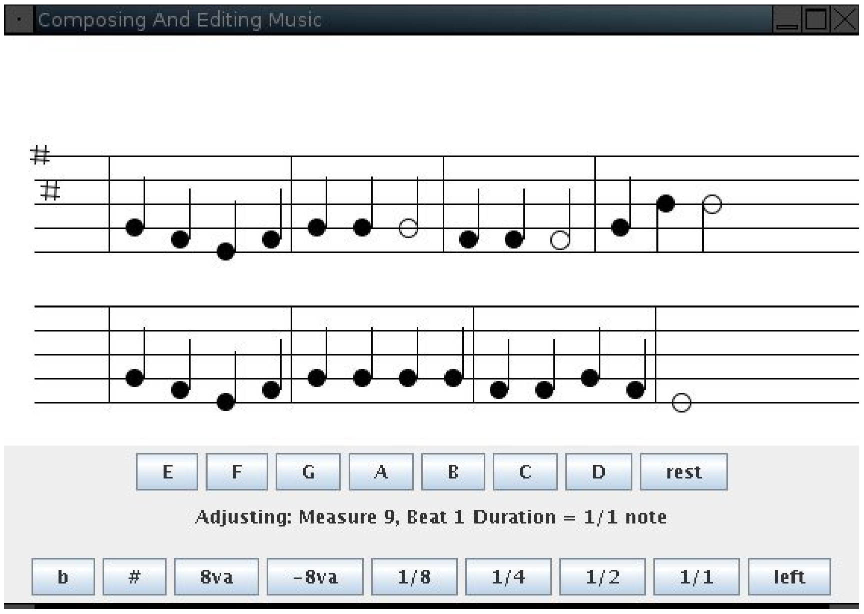
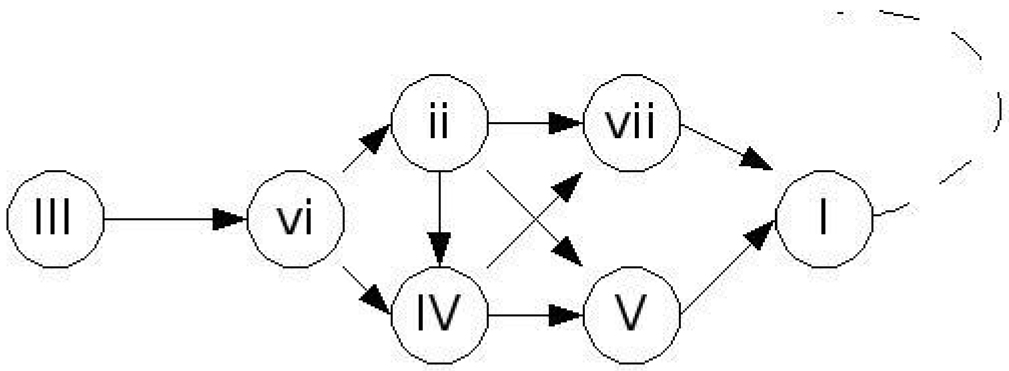


Figure 3: Architecture System



Adjusting: Measure 9, Beat 1 Duration = 1/1 note

Figure 4: Sample Input and Output


III
vi
ii
IV
vii
V
I