Piano Protégé

Table of Contents

The Project

Following the popularity of games such as Rock Band and Guitar Hero, we propose a game that is of the same vision but a different flavor. This program is a game using beat precision key strokes with the difference being the instrument being emulated is the piano. The selection of songs will range from beginning level using one hand and three keys to difficult where the player will use both hands and 14 keys to imitate the key strokes of many classical and contemporary songs played on piano. This will all be played either on the keyboard or on a MIDI keyboard.

The Problem

The multi-million dollar idea of simulating instrument playing developed into video games has hit the nation by storm, so why can't it be extended to other popular instruments? Many people have had the urge to play the piano or at least play song that use the piano, but don't have the skill of actually playing because of monetary lapses or lack of connections to a proper teacher. This program allows for actual player and non-players of the piano alike to have the opportunity to emulate playing songs on the piano.

The Impact

This program will allow anyone the opportunity to emulate playing songs written for piano without having to have any real piano playing skills. It will over time develop the player's sense of rhythm and possibly influence them to actually learn to play piano for real. The program will also open the player to a library of songs that use piano in the main melody.

The Motivation

This program was though of out of a mutual enjoyment of playing the games Rock Band and Guitar Hero. We thought that the implementation of this game would be a good way to use the skills we all have with music and programming and find a way to merge the two together. We also wanted to extend our knowledge of MIDI interfaces through java. This will allow for us to have experience dealing with external interface implementation though a java environment.

Gameplay

The gameplay of Piano Protégé will be very similar to the gameplay successfully implemented in games like Guitar Hero and Rock Band. The user will choose a song to play from a list; then the song will begin. While the song is playing, symbols representing notes will scroll down the screen in time to the music. The horizontal position of these symbols determines which note should be played. When a symbol reaches the bottom of the screen, the user should play the correct note at that time. If more than one note appears at the same height on the screen, a chord is represented, and the user should play each of these notes at the same time to receive credit. Every time the user plays a correct note at the right time, the user's score will increase. A combo is created by playing consecutive notes correctly without playing any extra wrong notes. As the combo increases, the user's score will increase at a higher rate. When the song ends, the user will be shown the total score.

Difference from Guitar Hero

Obviously, the main difference between Piano Protégé and Guitar Hero is the particular musical instrument simulated in each game. Guitar Hero uses a custom guitar-like controller to simulate playing a guitar. Piano Protégé will allow the user to use a MIDI keyboard, which provides for a much closer simulation of the piano. Also, for the piano, two sets of notes are needed – one for each hand. Therefore, Piano Protégé will have two sections of notes scrolling down the screen at the same time.

This screenshot from Guitar Hero is similar to what Piano Protégé will look like (minus the background and fireworks graphics). The screenshot shows two sections of notes, which will be seen in Piano Protégé. However, in Guitar Hero, the two sections are meant to be played by two different people, whereas in Piano Protégé, the left section will be played by the user's left hand, and the right section will be played by the same user's right hand.

Explanation of the MIDI protocol

MIDI, or the Musical Instrument Digital Interface, is an industry standard protocol designed to allow music hardware of all types and brands to communicate with one another. Originally, the protocol was only used as a high baud rate, serial protocol (much like RS-232), but eventually, it made its way to the computer and a standard file format evolved. The protocol defines many different messages that relate to musical controls. The messages that we will be focusing on are the note-on and note-off events.

The structure and purpose of these message types are very straight forward. The note-on event tells the event listener when a note on a keyboard is pressed and how hard it was pressed (known as the note's velocity) and the note-off event tells the event listener when that note was released. Let us briefly examine the structure:

Note-on Event

1-0-0-1 X-X-X-X **Byte 1**

0-X-X-X-X-X-X-X **Byte 2**

0-X-X-X-X-X-X-X **Byte 3**

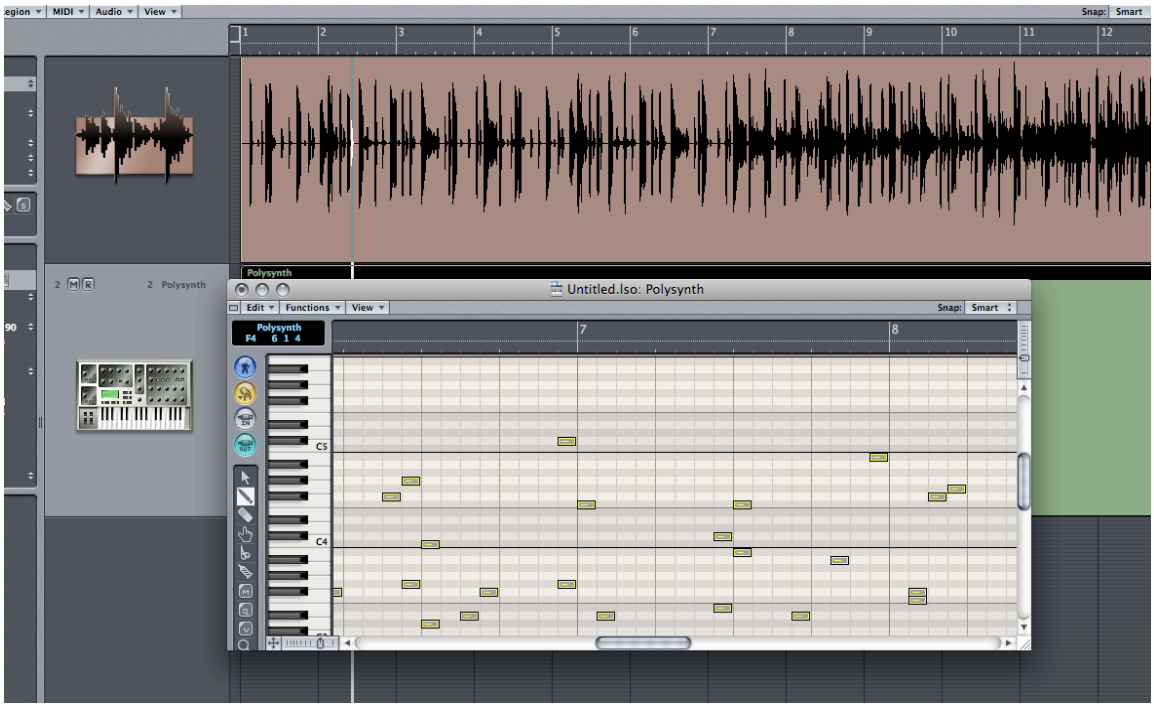
The note-on event consists of three bytes. The most significant half of the first byte is the code 1-0-0-1 which is the code for a note-on event. The least significant half, X-X-X-X here, is the channel number. This value can be arbitrarily chosen by us when we create the MIDI file so an explanation of its use is not important. The important parts are bytes 2 and 3. Byte 2 is the note value. Since we have 7 bits to work with, it is a number between 0 and 127. This value corresponds to the notes of a keyboard with 0 being the lowest pitched note and 127 being the highest pitched note. The value 60 usually corresponds to middle C on the keyboard. Byte 3 represents the velocity of the note. As explained earlier, the velocity is a number between 1 and 127 that tells the event listener how hard the note was hit. For our purposes, we are only concerned with this number being above a certain threshold, for example, only velocity > 60 constitutes a note played event. The note-off event is almost exactly the same as the note-on event except the code is 1-0-0-0 and the

velocity is always 0. The point of the note-off event in our case is to let us know how long the player is holding down a note.

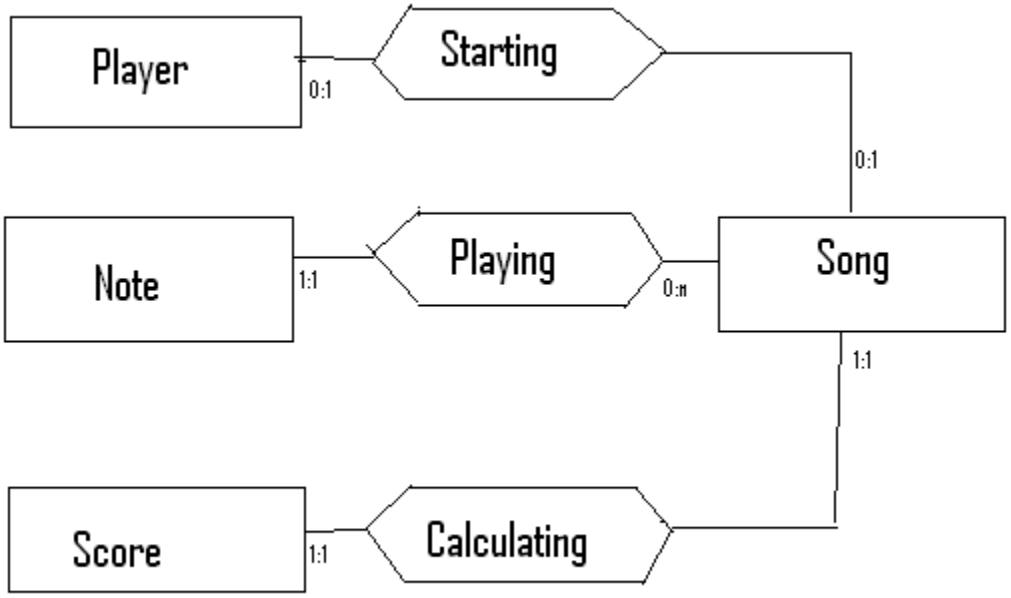
Song Construction

Now that we have explained what MIDI is, we will show how the songs will be constructed. To create the MIDI files, we will use a MIDI editor. The layout of these programs are very straightforward. They consist of a grid of squares where the X-axis corresponds to a musically defined quantization (1/4 of a bar, 1/8th of a bar, etc) and the Y-axis corresponds to the ascending notes of a keyboard (also the ascending note values in the MIDI specification). More specifically, we will be using a MIDI editor inside a multi-track studio. What this will allow us to do is import a song as sound file (in wav, mp3, ogg, etc) and line it up with the MIDI notes that we want the user to play. So basically, a song (as we define it) will consist of a MIDI file and a sound file. Since they are both synced, we can use a single timecode for both files (the sound file playing and the MIDI file being visualized with our custom display). We can create the MIDI notes in a number of ways, we can play the song file and have the multi-track studio record someone playing the keyboard, we can use a program to convert sheet music to MIDI, or we can just put it in by hand with a mouse and the midi editor interface.

Here is a picture of song construction in progress, you can see the actual sound file represented by the sound wave on top, and the MIDI editor right under it:


MIDI
Audio
View
Snap: Smart
Polysynth
Untitled.lso: Polysynth
Edit
Functions
View
Snap: Smart
C5
C4

ERM



Player(Player ID, Player name, HasScore)

Song(Song Id, Title, Composer, NumberofNotes, Length, Tempo, Difficulty)

Note(Note Id, Song Id, Keys, Duration, WaitTime)

Score(Player ID, Song Id, Date Played, Score, Placement, HighestCombo)

Calculation(Song Id, Score, NumberOfCorrectNotes, HighestCombo)

Playing(Song Id, Note Id, boolCorrect)

Starting(Player Id, Song Id, difficulty)

0:1 Player to Starting link:
A player may at any one time have a minimum of 0 songs starting and a maximum of 1 song being started

0:1 Song to Starting link:
A minimum of 0 songs may be starting at any one time to a maximum of 1.

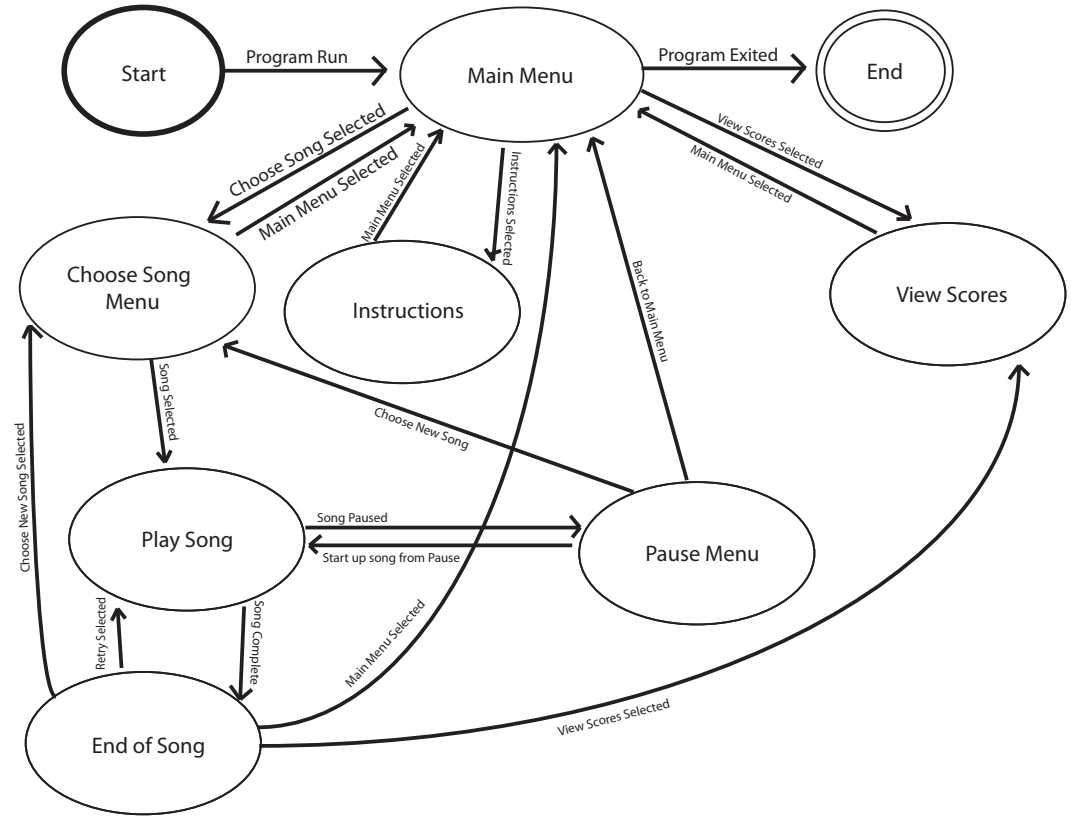
0:n Song to Playing link:
A Song may at any one time be playing a minimum of 0 notes and a maximum of n notes

1:1 Note to Playing link:
At any one time a note will be playing for only one song

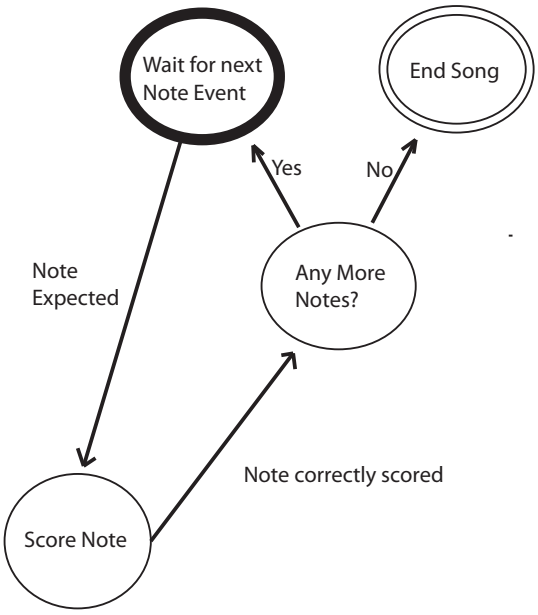
1:1 Song to Calculating link:
A song may at any one time have 1 score being calculated at a time

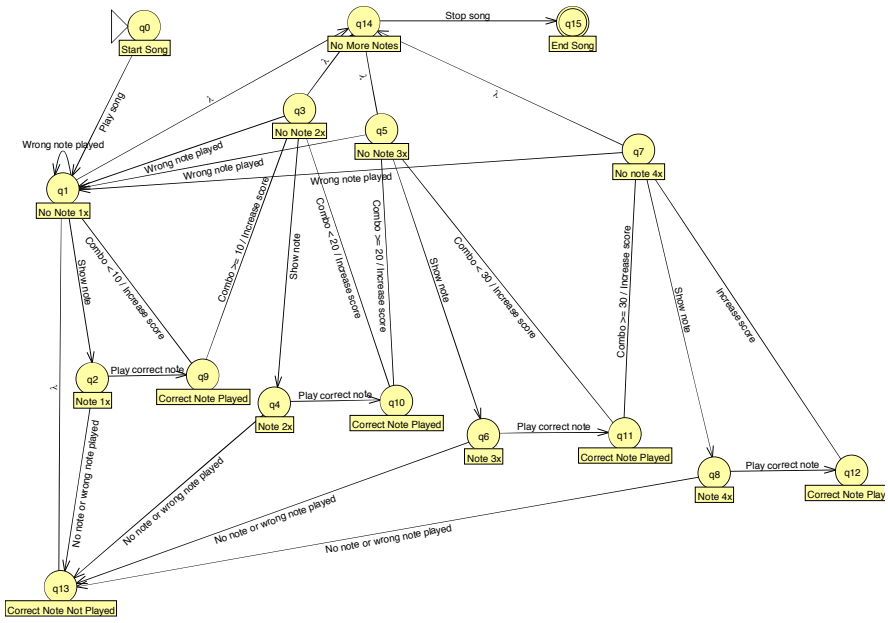
1:1 Score to Calculating link:
A Score may at any one time be calculated for only one song at a time

Menus and Interfaces FSM



Note FSM


Wait for next
Note Event
End Song
Yes
No
Note
Expected
Any More
Notes?
Note correctly scored
Score Note

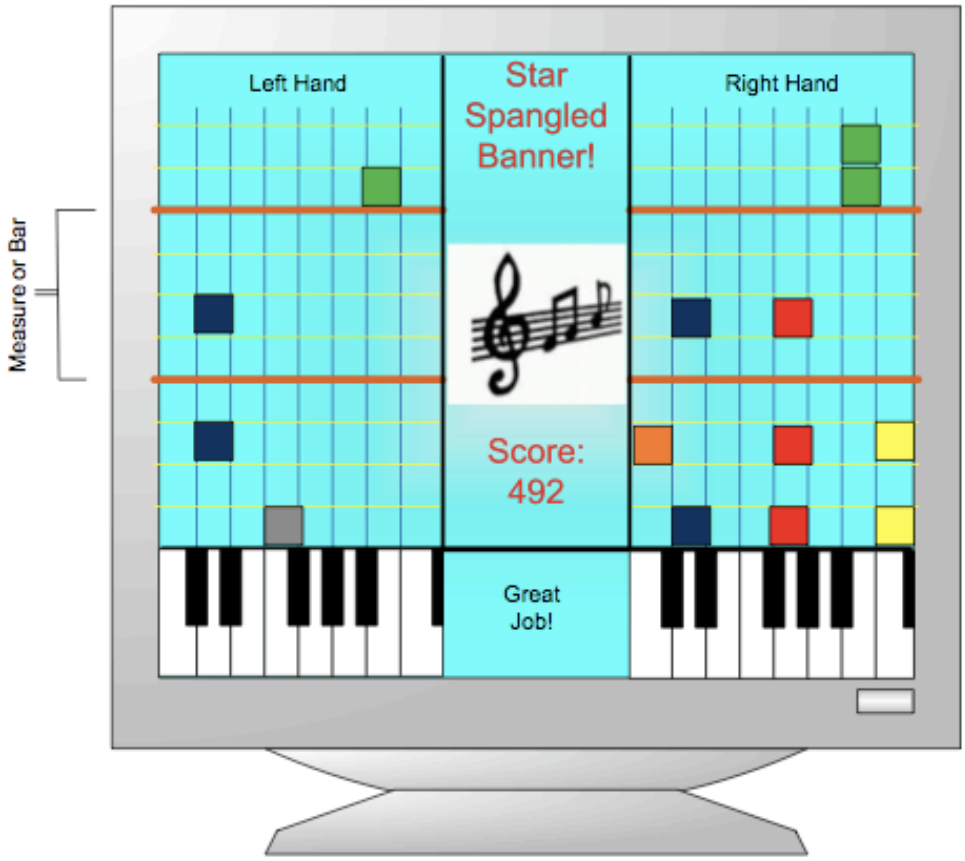

q0
Start Song
Play song
q14
No More Notes
Stop song
q15
End Song
λ
q3
No Note 2x
q5
No Note 3x
q7
No note 4x
Wrong note played
q1
No Note 1x
Show note
Combo < 10 / Increase score
Combo >= 10 / Increase score
Show note
Combo < 20 / Increase score
Combo >= 20 / Increase score
Show note
Combo < 30 / Increase score
Combo >= 30 / Increase score
Show note
Increase score
q2
Note 1x
Play correct note
q9
Correct Note Played
q4
Note 2x
Play correct note
q10
Correct Note Played
q6
Note 3x
Play correct note
q11
Correct Note Played
q8
Note 4x
Play correct note
q12
Correct Note Play
No note or wrong note played
q13
Correct Note Not Played

Screenshots


Main Menu
- Choose Song
- Instructions
- View Scores
End Game


Choose A Song!
Easy
- Mary Had a Little Lamb
- Twinkle, Twinkle, Little Star
Not So Easy
- Star Spangled Banner
- Yankee Doodle
Harder than Not So Easy
- Like a Rolling Stone
- What's Going On
Back to Main Menu

"Play"


Measure or Bar
Left Hand
Star Spangled Banner!
Right Hand
Score: 492
Great Job!


Song Completed!
Score: 1024
- Retry Song
- Choose Another Song
- View High Scores
- Back to Main Menu


Main Menu
- Choose Song
- Instructions
- View Scores
End Game


THANKS FOR
PLAYING
PIANO PROTÉGÉ!