Smart Player Piano

Spring 2010 Design Report

Jason Pawlak (Adviser: Dr. Anca Ralescu)

Table of Contents

Project Requirements

Description and Scope

The Smart Player Piano allows users to mix the senses of sight and hearing into an interrelated piece of art. As a computer driven extension to the old player piano, the Smart Player Piano allows users to interact more with the music by adding, deleting, or changing notes of MIDI songs. It also gives the opportunity to create music and visual art from scratch or import images for viewing, hearing and modification. But what really adds the "smart" in the Smart Player Piano is the ability for the application to analyze and modify the image and song such that it follows basic music theory rules of key and chord progression.

a) An application will be developed named Smart Player Piano. It will either read in a MIDI and create a related Image file or read in a Image and create a MIDI file. Another option is to draw your own Image in the application.
b) Jason Pawlak – Student; Anca Ralescu – Advisor
c) The relation between the Image and MIDI will be very close and it will be attempted to create a Image that is pleasing to the eyes and a MIDI that is pleasing to the ears. Basic music theory rules will be applied such that the MIDI is pleasing to the ears, and program intelligence will be used to modify Images as minimal as possible
d) The application will have an output of both image and audio file that are related in ways previously described. It will also have the ability to be used as a simple drawing program. The user will be able to watch the image scroll, as does the paper on a player piano while hearing the instruments play.

Project Perspective

The Smart Player Piano is an independent project that is not related to any other applications available today. The application will be developed from the ground up.

Project Functions

While using the Smart Player Piano application, a user will be able to upload or draw an image that will be slightly modified to generate an audio file. A player piano functions by a scroll of paper with holes running across a bar that has a vacuum. When the vacuum is not covered by paper, the note on the piano (relative left = lower note, right = higher note) is played. The same will happen in this application. When a pixel is drawn, the further left, the lower the note, the further the right, the higher the note will be played. The image will be slightly modified for the reason that the audio will follow music theory rules to generate music that is generally accepted as pleasing to the ears.

User Characteristics

The intended user is anyone that has a computer. The application will be simple enough that any person that has experience with using a mouse will be able to generate music and visual art. For the younger audience, this application could serve as a tool for learning about music and the range of notes. For musicians, the application could be used for fun to see what their music "looks like". For visual artists, the opposite is true and they could see how their art "sounds". For any other user, the application can serve as a fun tool to play with visual and audio art.

Specific Requirements

Inputs: MIDI file, Image file, User in application drawing
Output: MIDI file, PNG file
Once the user gives a valid input, the application is able to generate the rest of the said inputs. Only one is required to display all three. The application will not accept any file types that aren't recognized to be of MIDI or Image file type. After importing the file or drawing their own image, there is an array of operations that the user can do to affect the image and thus the audio as well. The user is able to export a MIDI or PNG file at any point and quit or continue modifying the image.

Definitions, Acronyms and Abbreviations

PNG: Portable Network Graphics; an open protocol for digital images

MIDI: Musical Instrument Digital Interface. A MIDI file doesn't contain actual audio data, but rather contains commands that let MIDI-capable synthesizers re-create a specific musical passage.

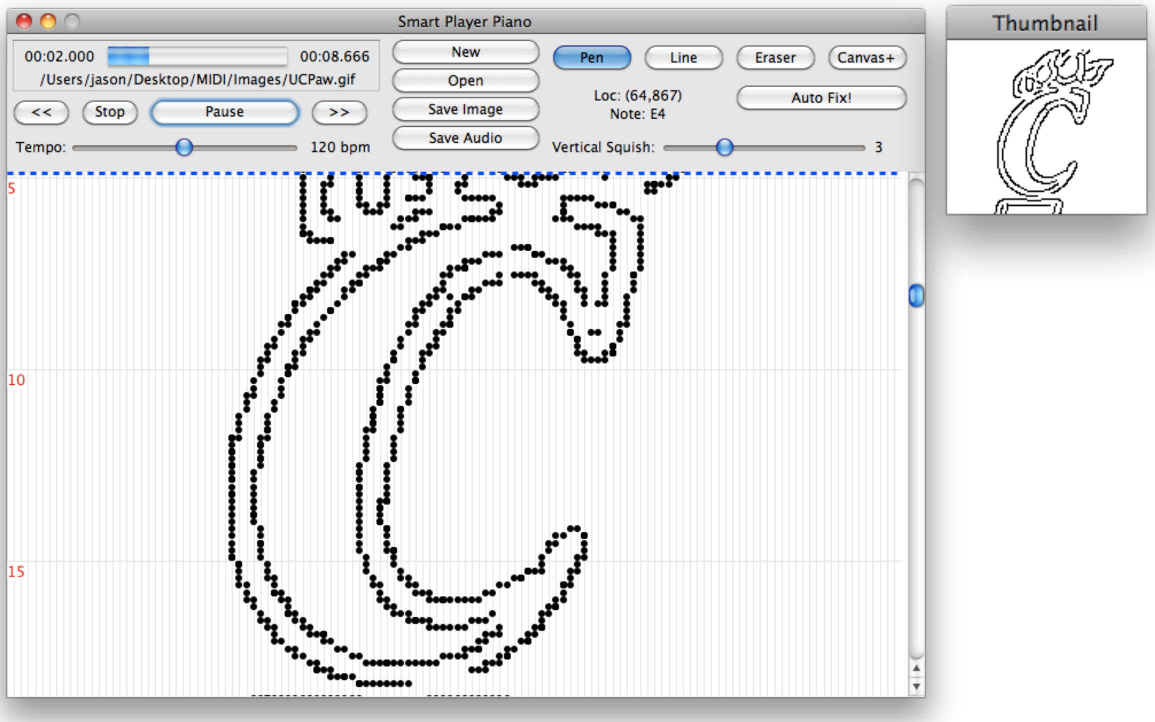
Music Theory: The field of study that deals with how music works. It examines the language and notation of music. It identifies patterns that govern composers' techniques. In a grand sense, music theory distills and analyzes the parameters or elements of music – rhythm, harmony (harmonic function), melody, structure, form, and texture.

Player Piano: self-playing piano, containing a pneumatic or electro-mechanical mechanism that operates the piano action via pre-programmed music perforated paper, or in rare instances, metallic rolls.

References

- http://www.xiliphone.com/glossary2.html
- http://wordnetweb.princeton.edu/perl/webwn?s=bitmap
- http://en.wikipedia.org/wiki/Music_theory
- http://en.wikipedia.org/wiki/Player_piano

Interface Specifications



File -> New, Open, Save Audio, Save Image, Settings, Exit
Edit -> Undo
Help -> About, Getting Started

Description

This is the application window that the user will view when using the Smart Player Piano Application. This window includes the sheet where the music is drawn in addition to all tools and playback options available to the user. The tempo is set by a slider underneath the playback buttons. When the user draws on the screen, the dots are limited to the vertical lines that specify notes in half step intervals. The different drawing buttons allow for different methods of easier drawing. Each vertical line represents a key on the piano. Lines drawn on the far left will be low notes and lines drawn on the far right will be high notes. The window on the right is a representation of the true image size. One dot on the window on the left is equal to one pixel in the window on the right.

Interface Goal

Allow users to draw images and play music relative to the image or import music (MIDI file) and view a drawing relative to the music.

Test Plan and Results

Overall Test Plan

This test plan is organized in three different levels. An example of this three level system is A.B.C where C is variations on completing task B and where B are lower level tasks to complete higher level task A. The purpose of this numbering scheme is for easy identification of issues that might be associated. During testing, the user conducting the tests should be able to conduct A level tasks independently of one another, unless a prerequisite is noted in the test case description (and for the case that 0.X.X is a prerequisite for all test cases except 0.X.X). In contrast, B level tasks might depend on previous B level tasks. C level tasks may be completed in any order and not all C level tasks, but at minimum one, must be completed before incrementing the B level task.

Test Case Description

0.X.X Initializing Application

All of these tests occur with no instances of the application running

1. TEST ***0.10.10***	**2. Purpose:** To verify that the application initializes		
3. Description: Locate the application icon and double click			
4. Inputs: n/a		**5a. Expected Results:** Application should load. Icon is in dock and toolbar is populated. No canvas.	
		5b. Actual Results: expected	
6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST ***0.10.20***	**2. Purpose:** To verify that the application initializes
3. Description: Locate a file of type .bmp Right click this file and select Open with -> Smart Player Piano	

4. Inputs: File of type .bmp		**5a. Expected Results:** Application should load. Icon is in dock and toolbar is populated. Canvas should have dots drawn.	
		5b. Actual Results: functionality removed	
6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *0.10.30*	**2. Purpose:** To verify that the application initializes		
3. Description: Locate a file of type .mid Right click this file and select Open with -> Smart Player Piano			
4. Inputs: File of type .mid		**5a. Expected Results:** Application should load. Icon is in dock and toolbar is populated. No canvas.	
		5b. Actual Results: functionality removed	
6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

10.X.X Load Image

1. TEST *10.10.10*	**2. Purpose:** To verify that an image loads properly		
3. Description: Click File -> Open and select a file of type .bmp. Select open.			
4. Inputs: file of type .bmp		**5a. Expected Results:**	

		The image should appear on the canvas	
		5b. Actual Results: works as expected	
6. Case indication [✓] Normal ☐ Abnormal ☐ Boundary	**7. Test indication** [✓] Blackbox ☐ Whitebox	**8. Test indication** [✓] Functional ☐ Performance	**9. Test indication** [✓] Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *10.10.20*	**2. Purpose:** To verify that an image loads properly		
3. Description: Locate a file of type .bmp. Click and drag the icon of this file onto the dock icon of Smart Player Piano or an open canvas			
4. Inputs: file of type .bmp		**5a. Expected Results:** The image should appear on the canvas	
		5b. Actual Results: works as expected	
6. Case indication [✓] Normal ☐ Abnormal ☐ Boundary	**7. Test indication** [✓] Blackbox ☐ Whitebox	**8. Test indication** [✓] Functional ☐ Performance	**9. Test indication** [✓] Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

20.X.X Load MIDI

1. TEST *20.10.10*	**2. Purpose:** To verify that an audio file loads properly		
3. Description: Click File -> Open and select a file of type .mid. Select open.			
4. Inputs: file of type .mid		**5a. Expected Results:** The image should appear on the canvas. Click play to verify correct music	
		5b. Actual Results:	

		works as expected	
6. Case indication ✓ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ✓ Blackbox ☐ Whitebox	**8. Test indication** ✓ Functional ☐ Performance	**9. Test indication** ✓ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *10.10.20*	**2. Purpose:** To verify that an audio file loads properly		
3. Description: Locate a file of type .mid. Click and drag the icon of this file onto the dock icon of Smart Player Piano or an open canvas			
4. Inputs: file of type .mid		**5a. Expected Results:** The image should appear on the canvas. Click play to verify correct music	
		5b. Actual Results: works as expected	
6. Case indication ✓ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ✓ Blackbox ☐ Whitebox	**8. Test indication** ✓ Functional ☐ Performance	**9. Test indication** ✓ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

30.X.X Blank Canvas

1. TEST *30.10.10*	**2. Purpose:** To verify that a blank canvas loads properly	
3. Description: Click File -> New.		
4. Inputs: n/a		**5a. Expected Results:** A window with a blank white canvas should appear. Vertical lines mark notes and are part of a blank canvas.
		5b. Actual Results: works as expected

6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

40.X.X Playback

A prerequisite step of 10.X.X or 20.X.X is required for 40.X.X

1. TEST *40.10.10*	**2. Purpose:** To verify that audio playback is correct in both audible and visual forms		
3. Description: Click 'Play' button in toolbar			
4. Inputs: n/a		**5a. Expected Results:** Music should play relative to the image shown. The image should also be 'moving' downward or rather, the canvas should be scrolling from bottom to top	
		5b. Actual Results: works as expected	
6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *40.10.20*	**2. Purpose:** To verify that audio playback is correct in both audible and visual forms	
3. Description: Click Playback -> Play in the menu bar		
4. Inputs: n/a		**5a. Expected Results:** Music should play relative to the image shown. The image should also be 'moving' downward or rather, the canvas should be scrolling from bottom to top
		5b. Actual Results:

		works as expected	
6. Case indication ✓ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ✓ Blackbox ☐ Whitebox	**8. Test indication** ✓ Functional ☐ Performance	**9. Test indication** ✓ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *40.10.30*	**2. Purpose:** To verify that audio playback is correct in both audible and visual forms		
3. Description: Modify the tempo by moving the tempo meter			
4. Inputs: n/a		**5a. Expected Results:** Music should play relative to the image shown. The image should also be 'moving' downward or rather, the canvas should be scrolling from bottom to top. If a higher tempo, music should play faster. If a lower tempo, music should play slower	
		5b. Actual Results: works as expected	
6. Case indication ✓ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ✓ Blackbox ☐ Whitebox	**8. Test indication** ✓ Functional ☐ Performance	**9. Test indication** ✓ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

50.X.X Manual image modification

A prerequisite of step 10.X.X or 20.X.X or 30.X.X is required for 50.X.X

1. TEST *50.10.10*	**2. Purpose:** To verify that the canvas can be modified with addition of dots	
3. Description: Use tools to add dots to the canvas		
4. Inputs: n/a		**5a. Expected Results:**

		Dots should stay on canvas in whatever form the user chooses	
		5b. Actual Results: works as expected	
6. Case indication [✓ Normal] ☐ Abnormal ☐ Boundary	**7. Test indication** [✓ Blackbox] ☐ Whitebox	**8. Test indication** [✓ Functional] ☐ Performance	**9. Test indication** [✓ Unit] ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *50.20.10*	**2. Purpose:** To verify that the canvas modifications have been saved		
3. Description: Press play for playback			
4. Inputs: n/a		**5a. Expected Results:** User should hear the proper dots being played once crossing threshold	
		5b. Actual Results: works as expected	
6. Case indication [✓ Normal] ☐ Abnormal ☐ Boundary	**7. Test indication** [✓ Blackbox] ☐ Whitebox	**8. Test indication** [✓ Functional] ☐ Performance	**9. Test indication** [✓ Unit] ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *50.20.20*	**2. Purpose:** To verify that the canvas modifications have been saved		
3. Description: File -> Save Image			
4. Inputs: n/a		**5a. Expected Results:** The saved image should look like the canvas	
		5b. Actual Results: works as expected	
6. Case indication	**7. Test indication**	**8. Test indication**	**9. Test indication**

[✓ Normal] ☐ Abnormal ☐ Boundary	[✓ Blackbox] ☐ Whitebox	[✓ Functional] ☐ Performance	[✓ Unit] ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *50.20.30*	**2. Purpose:** To verify that the canvas modifications have been saved		
3. Description: File -> Save Audio			
4. Inputs: n/a		**5a. Expected Results:** The saved audio file should sound like the playback	
		5b. Actual Results: works as expected	
6. Case indication [✓ Normal] ☐ Abnormal ☐ Boundary	**7. Test indication** [✓ Blackbox] ☐ Whitebox	**8. Test indication** [✓ Functional] ☐ Performance	**9. Test indication** [✓ Unit] ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *50.30.10*	**2. Purpose:** To verify that the canvas can be modified with deletion		
3. Description: Use the eraser tool to erase some dots that are on the canvas			
4. Inputs: n/a		**5a. Expected Results:** The canvas should no show dots that the user removes with the eraser	
		5b. Actual Results: works as expected	
6. Case indication [✓ Normal] ☐ Abnormal ☐ Boundary	**7. Test indication** [✓ Blackbox] ☐ Whitebox	**8. Test indication** [✓ Functional] ☐ Performance	**9. Test indication** [✓ Unit] ☐ Integration
10. Comments:			**11. Outcome**

	☐ Pass ☐ Fail

1. TEST *50.40.10*	**2. Purpose:** To verify that the canvas modifications have been saved		
3. Description: Press play for playback			
4. Inputs: n/a		**5a. Expected Results:** User should hear the proper dots being played once crossing threshold	
		5b. Actual Results: works as expected	
6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *50.40.20*	**2. Purpose:** To verify that the canvas modifications have been saved		
3. Description: File -> Save Image			
4. Inputs: n/a		**5a. Expected Results:** The saved image should look like the canvas	
		5b. Actual Results: works as expected	
6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *50.40.30*	**2. Purpose:** To verify that the canvas modifications have been saved
3. Description: File -> Save Audio	

4. Inputs: n/a		**5a. Expected Results:** The saved audio file should sound like the playback	
		5b. Actual Results: works as expected	
6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

60.X.X Autofix

A prerequisite of 30.X.X is required for 60.X.X

1. TEST *60.10.10*	**2. Purpose:** To verify that Autofix functionality is working correctly		
3. Description: Add at least 50 dots randomly to the canvas			
4. Inputs: n/a		**5a. Expected Results:** Dots will be displayed on the canvas	
		5b. Actual Results: works as expected	
6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *60.20.10*	**2. Purpose:** To verify that Autofix functionality is working correctly
3. Description: Play audio and make note of musicality (how ‘normal’ the music sounds)	

4. Inputs: n/a		**5a. Expected Results:** Music should sound 'bad'	
		5b. Actual Results: works as expected	
6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *60.20.20*	**2. Purpose:** To verify that Autofix functionality is working correctly		
3. Description: Autofix -> Generate Report			
4. Inputs: n/a		**5a. Expected Results:** The report should show a low musicality score with notes in all sorts of keys	
		5b. Actual Results: works as expected	
6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *60.30.10*	**2. Purpose:** To verify that Autofix functionality is working correctly		
3. Description: Click the Autofix button in the toolbar			
4. Inputs: n/a		**5a. Expected Results:** The image on the canvas should appear to change slightly	
		5b. Actual Results: works as expected	

6. Case indication [✓ Normal] ☐ Abnormal ☐ Boundary	**7. Test indication** [✓ Blackbox] ☐ Whitebox	**8. Test indication** [✓ Functional] ☐ Performance	**9. Test indication** [✓ Unit] ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *60.30.20*	**2. Purpose:** To verify that Autofix functionality is working correctly		
3. Description: Autofix -> Run Autofix			
4. Inputs: n/a		**5a. Expected Results:** The image on the canvas should appear to change slightly	
		5b. Actual Results: works as expected	
6. Case indication [✓ Normal] ☐ Abnormal ☐ Boundary	**7. Test indication** [✓ Blackbox] ☐ Whitebox	**8. Test indication** [✓ Functional] ☐ Performance	**9. Test indication** [✓ Unit] ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *60.40.10*	**2. Purpose:** To verify that Autofix functionality is working correctly		
3. Description: Play audio and make note of musicality (how ‘normal’ the music sounds)			
4. Inputs: n/a		**5a. Expected Results:** Music should sound ‘good’	
		5b. Actual Results: works as expected	
6. Case indication [✓ Normal] ☐ Abnormal ☐ Boundary	**7. Test indication** [✓ Blackbox] ☐ Whitebox	**8. Test indication** [✓ Functional] ☐ Performance	**9. Test indication** [✓ Unit] ☐ Integration
10. Comments:			**11. Outcome**

	Pass Fail

1. TEST *60.40.20*	**2. Purpose:** To verify that Autofix functionality is working correctly		
3. Description: Autofix -> Generate Report			
4. Inputs: n/a		**5a. Expected Results:** The report should show a high musicality score with most notes in one single key	
		5b. Actual Results: works as expected	
6. Case indication [✓ Normal] Abnormal Boundary	**7. Test indication** [✓ Blackbox] Whitebox	**8. Test indication** [✓ Functional] Performance	**9. Test indication** [✓ Unit] Integration
10. Comments:			**11. Outcome** Pass Fail

70.XX Help

1. TEST *70.10.10*	**2. Purpose:** To verify that the help section of the application is functioning properly		
3. Description: Help -> Getting Started			
4. Inputs: n/a		**5a. Expected Results:** The window that appears should give a easy to understand and general overview of how to get started using the Smart Player Piano in all the areas covered in this test plan	
		5b. Actual Results: works as expected	
6. Case indication [✓ Normal] Abnormal Boundary	**7. Test indication** [✓ Blackbox] Whitebox	**8. Test indication** [✓ Functional] Performance	**9. Test indication** [✓ Unit] Integration
10. Comments:			**11. Outcome** Pass

	☐ Fail

1. TEST *70.10.20*	**2. Purpose:** To verify that the help section of the application is functioning properly		
3. Description: Help -> Search (for menu items such as 'Generate Report') and help topics like 'Playback'			
4. Inputs: n/a		**5a. Expected Results:** When searching for menu items, the menu item should be highlighted for easy clicking. For articles, helpful information should be displayed relative to search	
		5b. Actual Results: works as expected	
6. Case indication ☑ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ☑ Blackbox ☐ Whitebox	**8. Test indication** ☑ Functional ☐ Performance	**9. Test indication** ☑ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

80.X.X Terminate application

1. TEST *80.10.10*	**2. Purpose:** To verify that the application can be terminated properly		
3. Description: Smart Player Piano -> Quit			
4. Inputs: n/a		**5a. Expected Results:** All application windows should go away and icon be removed from the dock. Check processes for any remaining processes related to the application (TODO: update with what process names are)	
		5b. Actual Results: works as expected	
6. Case indication ☑ Normal ☐ Abnormal	**7. Test indication** Blackbox Whitebox	**8. Test indication** Functional Performance	**9. Test indication** Unit Integration

☐ Boundary			
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

1. TEST *80.10.10*	**2. Purpose:** To verify that the application can be terminated properly		
3. Description: Command + Q			
4. Inputs: n/a		**5a. Expected Results:** All application windows should go away and icon be removed from the dock. Check processes for any remaining processes related to the application (TODO: update with what process names are)	
		5b. Actual Results: works as expected	
6. Case indication ✓ Normal ☐ Abnormal ☐ Boundary	**7. Test indication** ✓ Blackbox ☐ Whitebox	**8. Test indication** ✓ Functional ☐ Performance	**9. Test indication** ✓ Unit ☐ Integration
10. Comments:			**11. Outcome** ☐ Pass ☐ Fail

Test Case Matrix

0.X.X - Initializing Application				
0.10.10	Normal	Blackbox	Functional	Unit
0.10.20	Normal	Blackbox	Functional	Unit
0.10.30	Normal	Blackbox	Functional	Unit
10.X.X – Load Image				
10.10.10	Normal	Blackbox	Functional	Unit
10.10.20	Normal	Blackbox	Functional	Unit
20.X.X – Load MIDI				
20.10.10	Normal	Blackbox	Functional	Unit
20.10.20	Normal	Blackbox	Functional	Unit
30.X.X – Blank Sheet				
30.10.10	Normal	Blackbox	Functional	Unit

40.X.X – Playback				
40.10.10	Normal	Blackbox	Functional	Unit
40.10.20	Normal	Blackbox	Functional	Unit
40.10.30	Normal	Blackbox	Functional	Unit
50.X.X – Manual image modification				
50.10.10	Normal	Blackbox	Functional	Unit
50.20.10	Normal	Blackbox	Functional	Unit
50.20.20	Normal	Blackbox	Functional	Unit
50.20.30	Normal	Blackbox	Functional	Unit
50.30.10	Normal	Blackbox	Functional	Unit
50.40.10	Normal	Blackbox	Functional	Unit
50.40.20	Normal	Blackbox	Functional	Unit
50.40.30	Normal	Blackbox	Functional	Unit
60.X.X – Autofix				
60.10.10	Normal	Blackbox	Functional	Unit
60.20.10	Normal	Blackbox	Functional	Unit
60.20.20	Normal	Blackbox	Functional	Unit
60.30.10	Normal	Blackbox	Functional	Unit
60.30.20	Normal	Blackbox	Functional	Unit
60.40.10	Normal	Blackbox	Functional	Unit
60.40.20	Normal	Blackbox	Functional	Unit
70.X.X – Help				
70.10.10	Normal	Blackbox	Functional	Unit
70.10.20	Normal	Blackbox	Functional	Unit
80.X.X – Terminate application				
80.10.10	Normal	Blackbox	Functional	Unit
80.10.20	Normal	Blackbox	Functional	Unit

User Manual

Getting Started

You can launch the Smart Player Piano by double clicking the application icon.

Upon launching the Smart Player Piano you will be welcomed with a blank page.

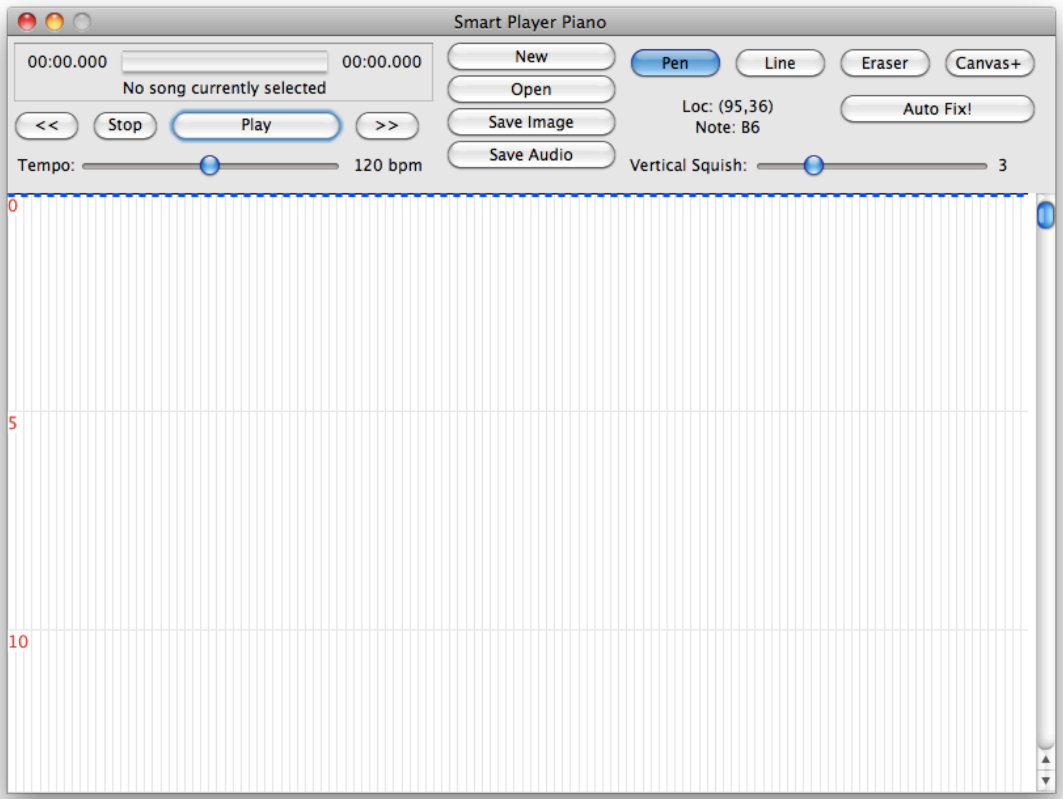


Figure 1: Main Smart Player Piano window

From this window you have song playback options on the left side (for use after a song is loaded), you have all the drawing tools on the right side that can be used at anytime, and you have the file options down the middle for creating new, saving, or loading songs. To start out, you will either Open a file or start drawing a new song with the drawing tools.

Figure 2: Smart Player Piano Thumbnail window
Another window is to the right of the main Smart Player Piano window that is much smaller. This window shows a preview of what your image on the main window would look like if it were saved as a file.

Loading Songs

You can load a variety of different file types into the Smart Player Piano, both images and audio files. The accepted image files are: JPEG, GIF, PNG, TIFF, and BMP. The only accepted audio file is a MIDI file. You can load these files by clicking the "Open" button on the Smart Player Piano main window or by going to the "File" menu and going to the "Open" option.

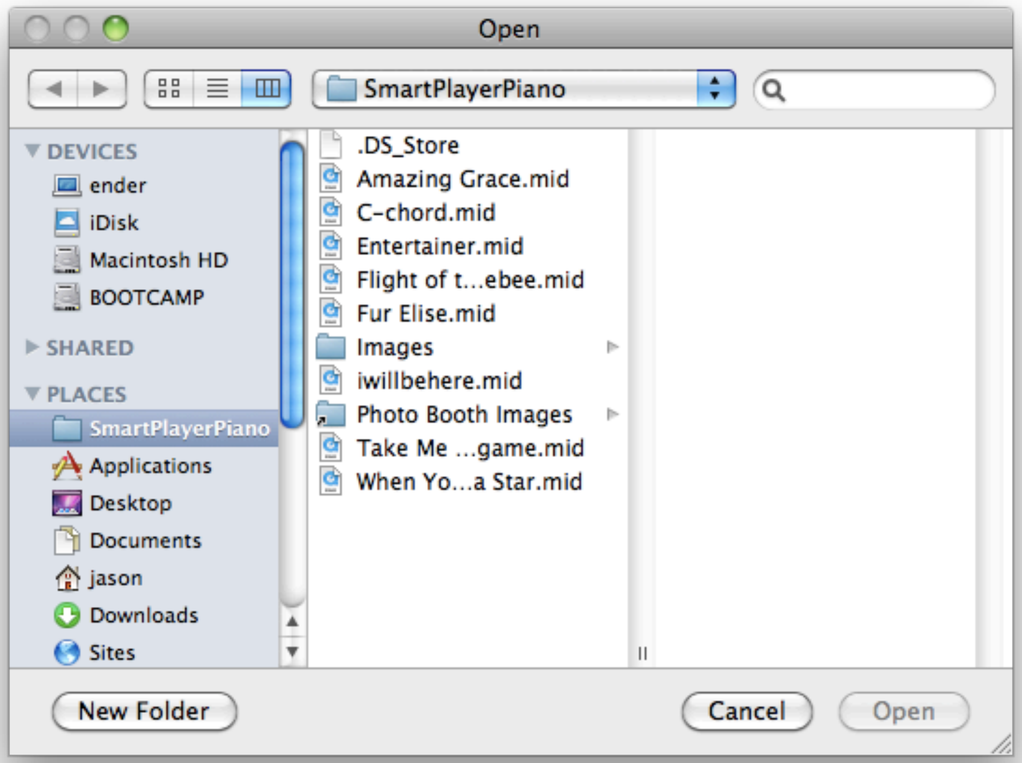


Figure 3: Open Dialog for opening image and audio files

Once you find a file that you'd like to open, click the open button and the image will be displayed on both the main window and the thumbnail preview.

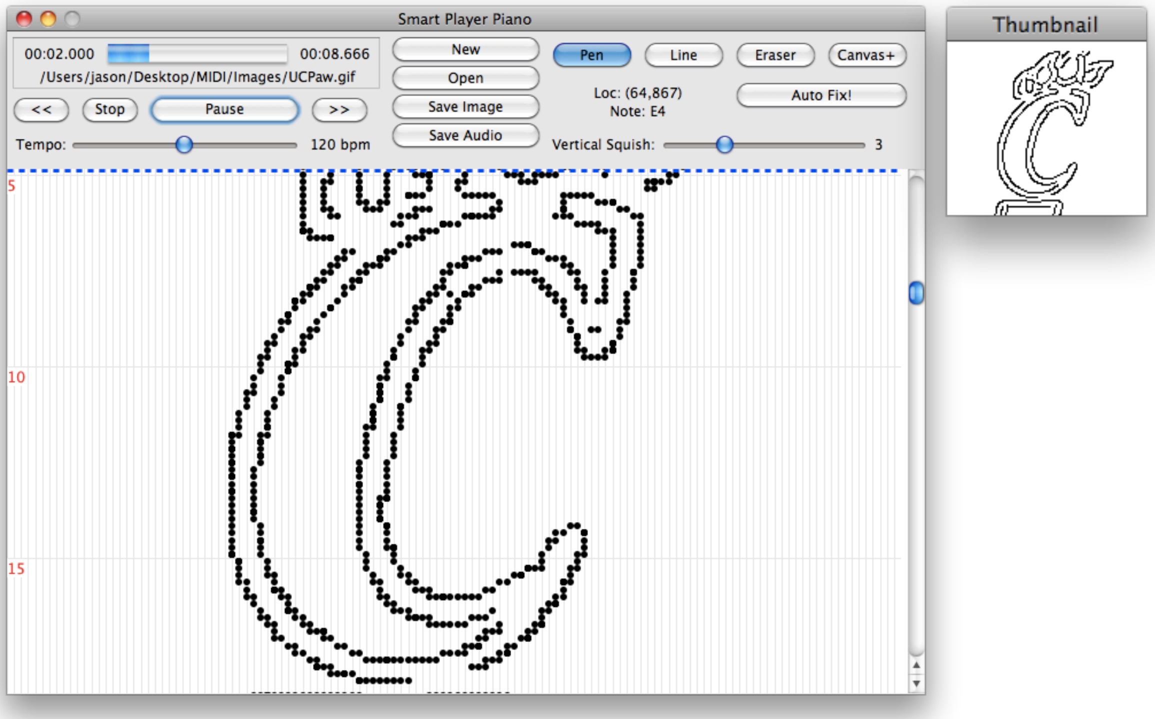


Figure 4: View of application after an image has been loaded

Saving Songs

Once a song has been drawn, you can then save it as an image and/or an audio file. Remember, the thumbnail window is what the image will look like after it has been saved to a file.

To save the image, you can click the “Save Image” button on the Smart Player Piano main window or can use the “File” menu and select “Save Image”

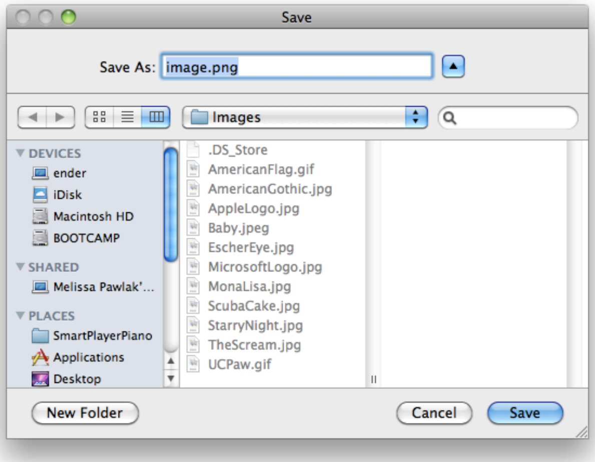


Figure 5: Save dialog for saving an image
Find the location and name the file anything you'd like before clicking the save button. After clicking the save button, the image will be saved as a file.

To save the image, you can click the "Save Audio" button on the Smart Player Piano main window or can use the "File" menu and select "Save Audio"

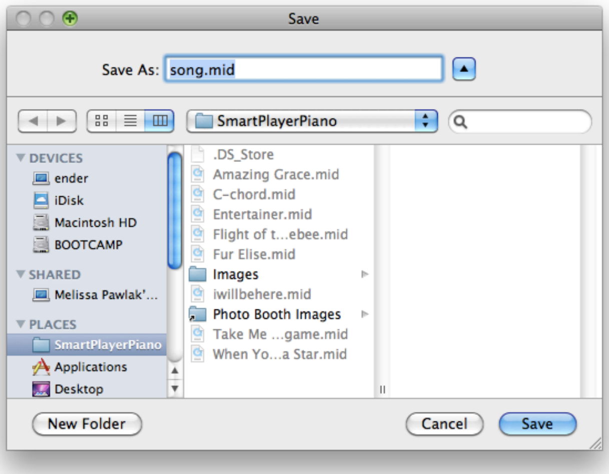


Figure 6: Save dialog for saving an audio file
Find the location and name the file anything you'd like before clicking the save button. After clicking the save button, the audio will be saved as a file.

Modifying Songs

Modifying a song or creating a song from scratch is a lot like using any normal paint program. On the top right of the main window (see Figure 1) the drawing tools are made available. The options are for a Pen, Line, and Eraser. All that you need to do is select which tool you'd like to use on the canvas and then click or click and drag to see the results of that tool.

Pen

Draw individual dots that are like a quick strike of a piano key. Click once to draw a single dot, click and drag to draw a series of dots.

Line

Draw a line by clicking where you'd like the note to begin, and then drag to where you'd like the note to end. A line will sound like someone pushes down a piano key and holds it down. Lines can only be drawn vertical, as a sustained note cannot change pitches.

Eraser

The eraser acts exactly like the pen, except that it takes away notes instead of draws them.

Undo
You can access the undo tool through the Edit menu. Using this tool will undo in reverse order all the other drawing commands (pen, line, eraser) you have given.

Canvas+
This tool adds more canvas to the image for drawing. Another way to get more canvas is to just draw down past the edge of the canvas. More canvas will automatically be added to fit all the notes you draw.

Listening to Songs
You can listen to a song that has been loaded or drawn at any time. The playback controls are located on the top left of the main window (see Figure 1). With these playback controls you can either (from left to right) rewind 5 seconds, stop playback of the song, start playback of the song (note that this button turns to pause during playback), or skip ahead 5 seconds of the song.

Above the playback controls, the song file path is displayed as well as the current time placement of the song and the length of the song. In addition, a blue bar is displayed to show the progress through the song.

Below the playback controls is a slider labeled as Tempo. Moving this slider to the left will make the tempo go down (play the song slower) and moving the slider to the right will make the tempo go up (play the song faster). The tempo can be modified before the song has been played or during playback. When saving an audio file, the tempo of the audio file will be set to whatever the tempo slider is set to when saving the song.

AutoFix
The AutoFix is what makes the Smart Player Piano, smart. This tool is used when you click the AutoFix button near the drawing tools (see Figure 1). When this tool is clicked, the application will analyze the image and audio, modifying dots and lines slightly by moving them to the right or the left such that played with other dots, the music will follow basic music theory rules.

Final Results
The Smart Player Piano was fully and successfully implemented for all major milestones. The end result was an application that proved to be fun, intuitive, and enlightening for many age groups. At the Technology Exposition, a number of people, including many children, experienced the Smart Player Piano during demonstration and hands on.

The resulting application is in a state that could be released to the public for both code editing and/or use. There are a few minor bugs that could use some cleaning, however, the application is fairly stable.

The three main methods of using the Smart Player Piano was to either import a song for viewing and listening, import a picture for viewing and listening, or start from scratch in drawing an image for viewing and listening. All three methods were implemented fully and completely.

Goals vs. Accomplishments

The big picture goals of the Smart Player Piano were fairly concrete throughout the entire process of designing and developing the application. The big three goals, or uses, for the application are as stated above in the "Final Results" section, import a song for viewing and listening, import a picture for viewing and listening, or start from scratch in drawing an image for viewing and listening. All three of these goals were implemented fully and completely.

While the previously stated goals were more on the implementation side, another goal of the Smart Player Piano was to be an intuitive, fun, and enlightening experience for all that became users. The biggest test of these three goals was with school age children that attended the Technology Exposition and wanted to draw or play with images on the application. The kids seemed to catch on to the concept very quickly and appeared to thoroughly enjoy the experience.

While the big picture goals of the project remained unchanged for the majority of the design and implementation process, a lot of the smaller goals changed slightly throughout the development process. Since there have been no previous implementations of the Smart Player Piano concept, a lot of the algorithm for "AutoFixing" the image had to be modified on the fly. One example of this is with the option to either move pixels that don't fit the musical key or to delete pixels that don't fit the musical key. The original design was for the notes/pixels to be moved to the right and left, but after some experimentation, the result of removing the notes turned out better. This result was also dependent on the picture that was being fixed. So in order to better the results for the user, a settings window was added to the design so that the user could specify to the application at most how far a note should be moved if at all.

Another example of a small goal that was changed was during the moment in the "AutoFix" when the image transforms from original image to modified image. During the development process I would watch the image when it transformed to the modified image to see if what was in the picture was still recognizable. To my disappointment, it was challenging at times to see the results related to the original image due to the size of the image. To correct this, a preview window was added to the side of the main window. In the preview window, the image is shown how it would be saved to the file system. This is where one dot on the canvas is equal to one pixel in the saved image. Now when the image is modified, you can watch the preview image and see better the image once it has been transformed, less pixilated.

Overall, however, the design and goals of the Smart Player Piano remained rather unchanged from beginning to end.

Self-Assessments

Initial Self-Assessment

The project that I am working on is bringing together visuals and audio to meld cross-medium art. The software that will be developed will be easy and intuitive to use such that the created art is not only available to those that are tech or music savvy.

Through the courses that I have taken at both University of Cincinnati and Indiana University, I have learned a lot in the areas that this project requires. At IU, I took various music theory and keyboarding classes that are integral to the musical knowledge that will be required to make the program make music and not just noise. In my time at UC, I have taken a heavy load of computer programming such as Computer Science I and II, Network Systems Programming, Java Development among other classes that aren't centered around programming, but incorporate the skills heavily. Through these classes, I have learned the development life of an application and plan to apply this to this project. The language of choice for this application will be Java because of its cross platform functionality. I have learned Java in a class offered at UC, but the majority of that knowledge is from co-op experience.

During co-op opportunities while attending University of Cincinnati I had much experience coding. During my time with The Kroger Co, I programmed web application in Java, which is where most of my Java experience comes from. From the time that I worked at Apple, Inc, I learned a lot of developing within a framework that will be useful in handling MIDI files and images. While I do not plan on this application being a web application, the basics of Java that I learned and enhanced while at Kroger will be very beneficial. Along the same lines, while at Apple, I did not develop any software that was quite like the application that I plan to develop, but the experience of working with the third party libraries will help great amounts.

Through the experiences in the previous paragraphs, I feel that I am well prepared to take on the project that was described above. It will pose challenges in specific software areas that I have not had much experience, such as audio and music files, while also enhancing the knowledge that I already have in topics such as data structures.

Final Self-Assessment

The final results of the Smart Player Piano were better than I ever expected them to be. As stated in the "Initial Self-Assessment", the courses that I had taken at both

Indiana University in music and University of Cincinnati in computer science were equally important for the resulting Smart Player Piano application.

The Smart Player Piano was developed from the ground up and all code was original to me with the exception of the edge detection algorithm which is a public class written for Java. There was a lot of legwork needed to get the code up and running in a standalone application environment. Designing the graphical user interface was one of the largest challenges as my experience in that area was fairly limited. Through this experience there was a lot of learning going on. A lot of the Java language was new to me, especially the areas that dealt with the Java Sound Framework. After the project was completed, however, I feel much more confident and competent in my understanding of MIDI files and the Java Sound Framework.

In addition to learning more about the specifics of the Java language, I learned a lot by the experience of taking the Smart Player Piano from idea to implementation. Beyond the technical details, explaining the project to others who had no musical and/or computer programming experience was a very big hurdle. I quickly realized that explaining what the application is going to do so that others will understand is half the battle to have others back your idea. Throughout many of these conversations with others I kept on saying, "it'll make complete sense when you see it work." In the end, my statement was true, but it would have been nice to be able to help others understand my project without the actual project sitting in front of them. This was by far the most difficult part of the project.

Overall, I feel that my experience and final result was a complete success.

Summary of Hours and Billing

Fall

24 hours
Fall quarter had a lot of design and thought process hours. There wasn't too much work yet on the actual implementation of the Smart Player Piano as there wasn't a solid concept yet.

Winter

47 hours
During the Winter quarter most work was done in the research area of topics that needed to be addressed for the variety of areas in the Smart Player Piano. The largest area of research needed was in the Java Sound Framework. It was this framework that implemented the MIDI files and allowed for my own custom class to be built around the MIDI file format. It was during this time that the application started to be developed by designing and building an application that acted as a music player for MIDI files.

Spring

54 hours

Throughout Spring quarter, many hours were spent working on the Smart Player Piano application. Where in previous quarters a lot of time was spent on design, the Sprint quarter was packed with a lot of hands on coding

Total

125 hours

Details

Key: Fall Quarter; Winter Quarter; Spring Quarter

Date	Hours	Comments
9/30/2009	3	See http://groups.google.com/group/smart-player-piano
10/7/2009	1	See http://groups.google.com/group/smart-player-piano
10/14/2009	0.5	See http://groups.google.com/group/smart-player-piano
10/21/2009	1.5	See http://groups.google.com/group/smart-player-piano
10/28/2009	3	See http://groups.google.com/group/smart-player-piano
11/13/2009	3	See http://groups.google.com/group/smart-player-piano
11/18/2009	3	See http://groups.google.com/group/smart-player-piano
11/25/2009	2.5	See http://groups.google.com/group/smart-player-piano
12/3/2009	6.5	See http://groups.google.com/group/smart-player-piano
1/15/2010	10	See http://groups.google.com/group/smart-player-piano
1/19/2010	5	See http://groups.google.com/group/smart-player-piano
1/23/2010	5	See http://groups.google.com/group/smart-player-piano
1/24/2010	6.5	Finished GUI MIDI Music Player, see Milestone 2 page in group. Also completely re-made my timeline
1/25/2010	4.5	Added tracks to mute to music player. Researched patch lists too. Put together design review document
2/4/2010	1	Attended group meeting with Dr. Ralescu's graduate students to discuss my project and take suggestions
2/22/2010	6	Worked on and completed my test plan for the application. I believe I have a good format, but not sure it is in depth enough
2/23/2010	1	Reviewed my design review peer comments. I am taking actions to correct and enhance areas that pose a variety of concerns to multiple people
2/25/2010	3	Finally able to continue working with code and breaking down the contents of

		the MIDI file. Have a 'print out' of notes played in midi
3/1/2010	2	Worked on updating report and notebook
3/2/2010	3	Worked on updating report and notebook
4/21/2010	3	Worked on the poster for show
4/26/2010	3	Made significant progress on the abstract data types of the MIDI song. At the point where I will be putting the data type into the image.
4/27/2010	3	Worked a lot on reorganizing code into better classes so that the data types can be more easily utilized for input. Also drew the canvas
4/28/2010	9	Made great progress with the visual part of the application. The player piano now works just as a player piano does
4/29/2010	5	Made more progress with solidifying the player piano aspect. Also added in the ability to save midi files and image files. User can also draw on the canvas now and the player piano will read the music
4/30/2010	8	This was a big day for tightening up the code and general bug fixes. I was able to implement an Undo menu and spent a long time working on the delete tool.
5/1/2010	7	Implemented the image import feature. I used a public class that implements Canny Edge Detection. It works and looks great!
5/2/2010	8	Got the SMART part of the smart player piano done today!!!!
5/3/2010	8	Tech Expo!

Summary of Expenses

There is really no numeric budget for this project. It is completely software related and is being developed from the ground up. There are, however, a few references that have been very helpful in providing code snippets that have helped learn the Java Sound library and SWT.

- http://eclipse.org/swt/
- http://www.jsresources.org/faq_midi.html
- http://java.sun.com/products/java-media/sound/

The one area of the Smart Player Piano that was not developed by me was the implementation of the Canny Edge Detection algorithm for Java. Tom Gibara (http://www.tomgibara.com/computer-vision/canny-edge-detector) wrote an efficient implementation that he released to the Java Public Domain.

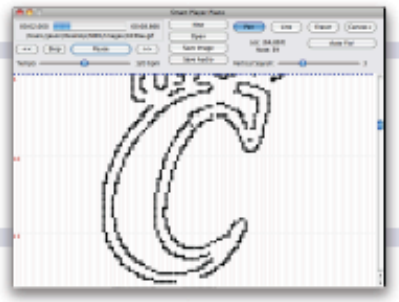
Technology Exposition

Handout

Smart Player Piano

Jason Pawlak (jason@pwlk.net)
Advisor: Dr. Anca Ralescu
University of Cincinnati

-- An amalgam of visual and audio arts --

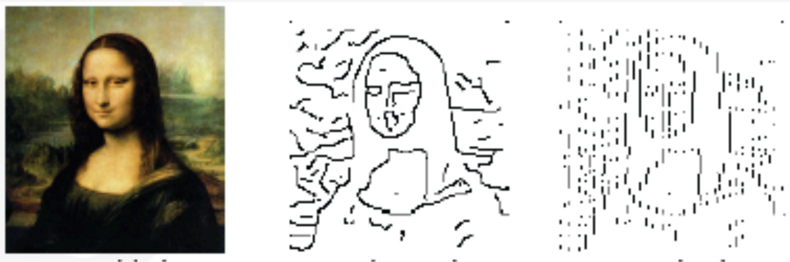


Purpose: To create an intuitive, fun and enlightening application that combines both visual and audible arts

Inspiration: Carlos Amorales' "Psicofonias" at the Cincinnati Contemporary Art 2008/2009 and a player piano my Grandmother had when I was little.

Description: The Smart Player Piano allows users to mix the senses of sight and hearing into an interrelated piece of art. As a computer driven extension to the old player piano, the Smart Player Piano allows users to interact more with the music by adding, deleting, or changing notes of MIDI songs. It also gives the opportunity to create music and visual art from scratch or import images for viewing, hearing and modification. But what really adds the "smart" in the Smart Player Piano is the ability for the application to analyze and modify the image and song such that it follows basic music theory rules of key and chord progression.

Transcription to Smart Player Piano music



Poster

Smart Player Piano

JASON PAWLAK

Advisor:
Dr. Anca Ralescu

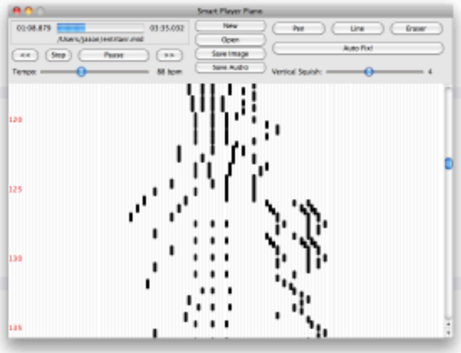
Description:

An application that combines visuals and audios into an amalgam of art for both the eyes and ears.

Goals:

1. Create an intuitive application for a user of any age
2. Allow users to import MIDI files to listen and view 2D representation
3. Allow users to import BMP files to listen and view
4. Allow user to create a new piece of music/ piece of 2D art and view it and listen to it

Results!



Player Piano?

- Powered by person pushing pedals!
- Vacuum detects where holes on scroll are located
- Holes further to the right are higher pitch than holes further to the left

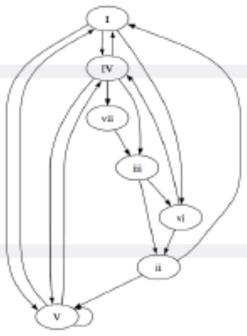
MIDI?

Stands for 'Musical Instrument Digital Interface.' A file that contains data for a digital synthesizer to play a song

How music theory works…

- Set of 7 notes make up a key
- Certain notes when played together sound better than other notes played together
- The distance between two notes is called an interval
- Three notes at specific intervals from each other make a chord
- Specific chords sound best played after other specific chords

Intervals (second to octave)



Smart?

The application will analyze an image that has been imported or created and make minor adjustments based on basic music theory to make the music sound 'pleasant' to the ear. Otherwise a random drawing could sound 'garbled.'