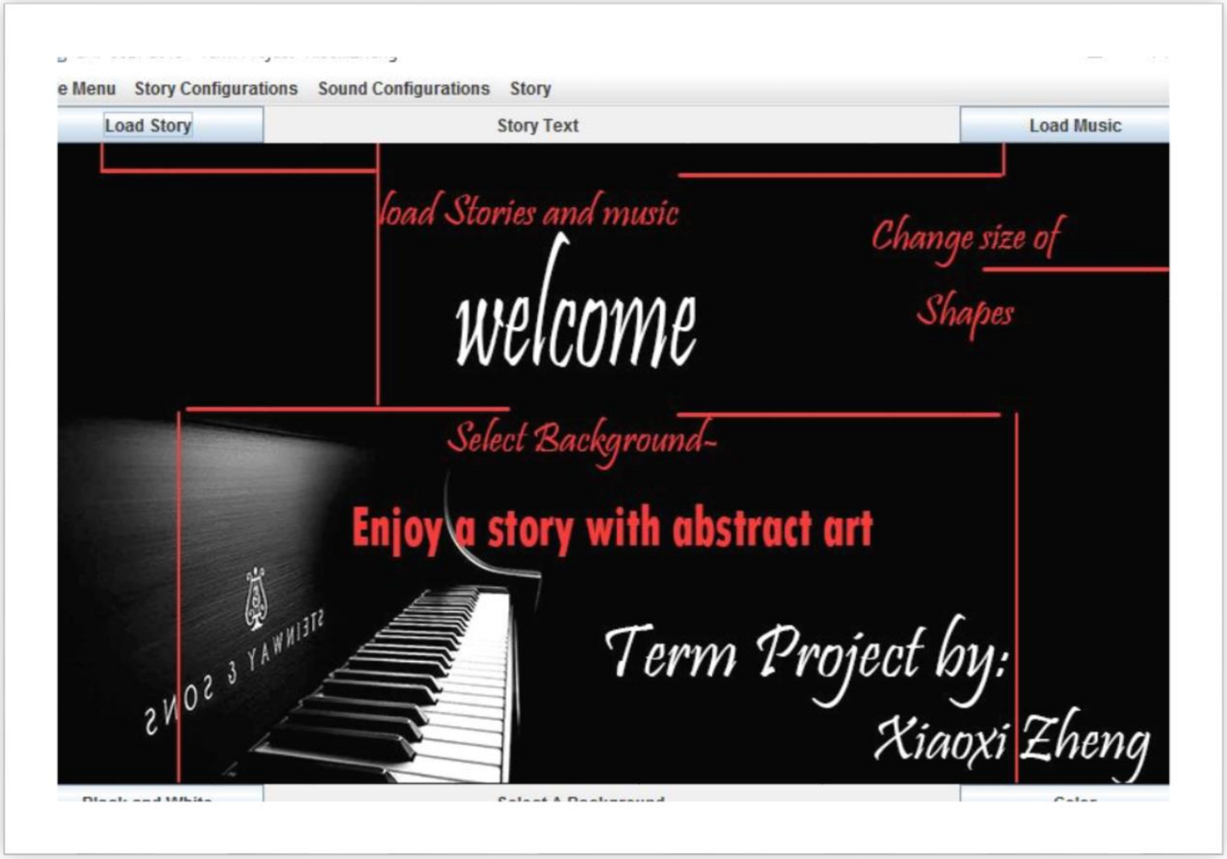


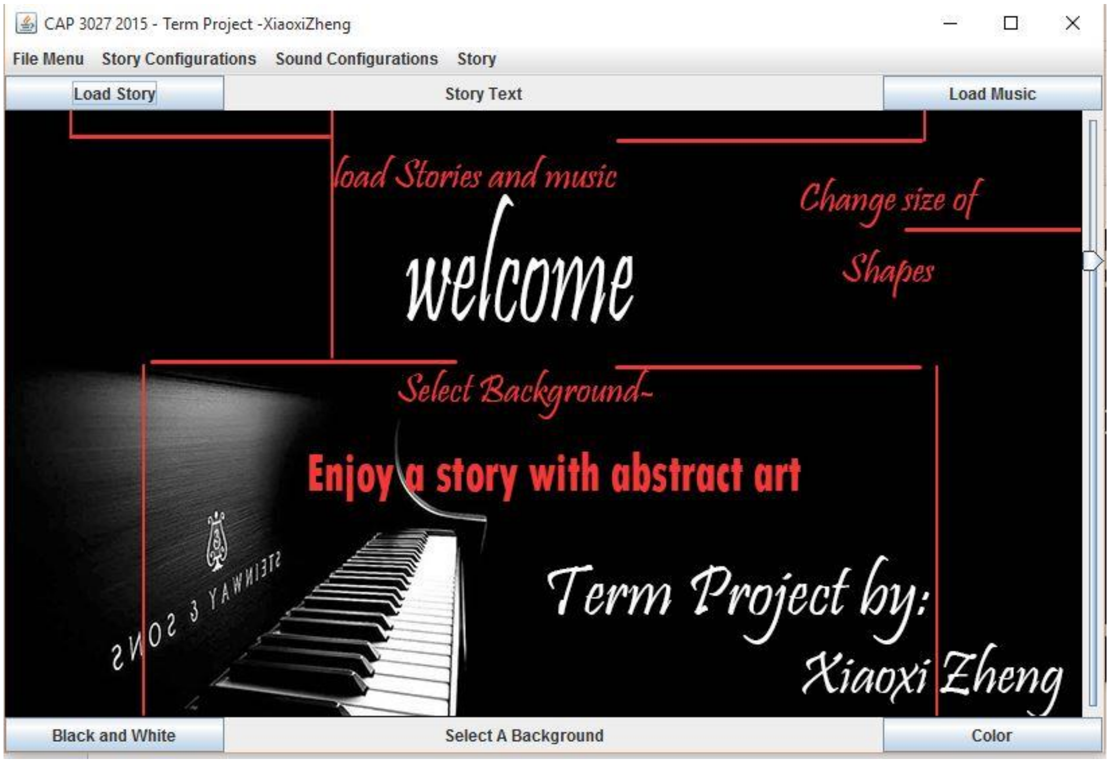
User Manual

ITERACTIVE ART VIEWER

Xiaoxi Zheng | Cap3027 | December 8, 2015

Program Overview

The goals for my term project was to create an interactive abstract art viewer. Like a musical birthday cards, a large part of this project was concentrated on the music. I wanted to create a viewer that users can click on that reflected the corresponding line of text and music. The major functionality of this project was being able to create text stories line by line, and respectively being able to save and reload them back into the program. The same functionality applies to the music aspect of this program. I allowed users to "write" music in text input of music notes, instruments, and the tempo they'd want to use. When all properties are loaded along with the background, the screen will display an array of "Shapes" they could click on that correspond to the number of lines in the story, and upon clicking on those shapes, the story will begin with text, corresponding image assets and some background music.



Compiling and running the program

To compile and run with jFugue(Mandatory):

Compile: javac -cp .:jFugue.jar storyTelling.java

Run: java -cp .;jFugue.jar storytelling

Note: I been compiling jFugue on cmd and can't get jFugue to properly work on Windows PowerShell.

Configuration:

Please follow this configuration so the program can run most smoothly. Please finish setting up all the story line, music, and background before triggering to have all the shapes to display. After everything is set up, trigger the program to draw all the shapes by selecting either [Story]→[Load Black or White Shapes] or [Story]→[Load Color Shapes].

Create, load, and save Story Line as .txt files -under the [Story configuration] on the menu bar, options to create, load and save story lines are available.

1. To [Create] a story line, user is first prompt for the number of lines they wish to input, and then follow by the contents they wish to input sequentially.
2. To [Save] a story line, [Save inputted story] is available and will save whatever user inputted so far as a text file.
3. To [Load] a story line, the program will expect any text file. The program will parse the documents line by line, and know the total number of lines it receive.

Create, load, and save Sound Track using jFugue library as .txt files-under the [Sound configuration] on the menu bar, options to create, load and save music are available.

1. To [Create] a music line, user is prompt for the following inputs:
    a. <u>Instrument type</u>, the program will accept string representation of instruments in the jFugue library.
        i. Sample input can be [Piano, 0 (int representation of piano), flute, guitar, rain and etc…please see the attachment to the end of this document for full list of possible inputs.
    b. Tempo
        i. Expects an int that represents the beat per minute you wish the music to be played.
    c. Music notes
        i. First expects the number of lines they wish to input, and then follow by the contents they wish to input sequentially.
            1. Note: the number of lines here should equal to the number of lines inputted when creating a story (So 1 line of music corresponds to 1 line of story text, if not - the less of the 2 number is picked to avoid crashing.)
            2. Sample music explained:
                a. Sample string: "Eq Ch. | Eq Ch. | Dq Eq Dq Cq   V1 I[Flute] Rw | Rw | GmajQQQ CmajQ"
                b. '|' (pipe) for indicating measures (optional). To specify durations, use 'q' for quarter duration, 'qqq' for three quarter notes (multiple durations can be listed together), and 'h' for half, 'w' for whole, and '.' for a dotted duration; 'R' for rest, and the chords G-Major and C-Major. Whitespace is not significant and can be used for visually pleasing or helpful spacing.
2. To [Save] a User inputted Music, [Save inputted story] is available and will save whatever user inputted so far as a text file.

3. To [Load] a music, the program will expect any text file. The program will parse the documents line by line, and know the total number of lines it receives.

Select background to paint Black and white or Color

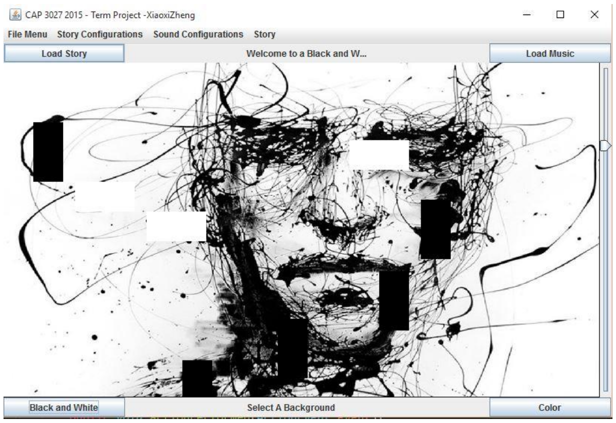
1. I have 2 pre-defined background images or the Abstract Art Pieces to be displayed when the program is at rest for receiving a mouseEvent. This needs to be set before "loading black or colored shapes".

Change folder name to look for image assets

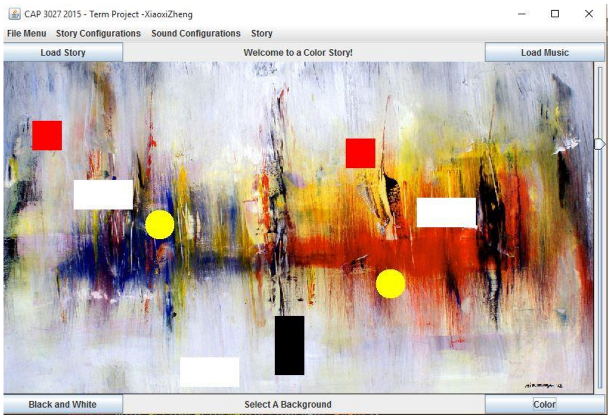
1 . This prompts users for the directory the program should go look in for the image assets triggered by mouseEvents. Please save the images assets in [num].jpg, where num is a placeholder for int starting at 0. If this never got selected, the default path is [xmas] inside the termProject folder.

Run

After everything is set up, the trigger to view the interactive part of this program is by going to either [Story]→[Load Black or White Shapes] or [Story]→[Load Color Shapes]. Then, you can click on the individual shapes to reflect the corresponding storyline and music.



[Story]→[Load Black or White Shapes]

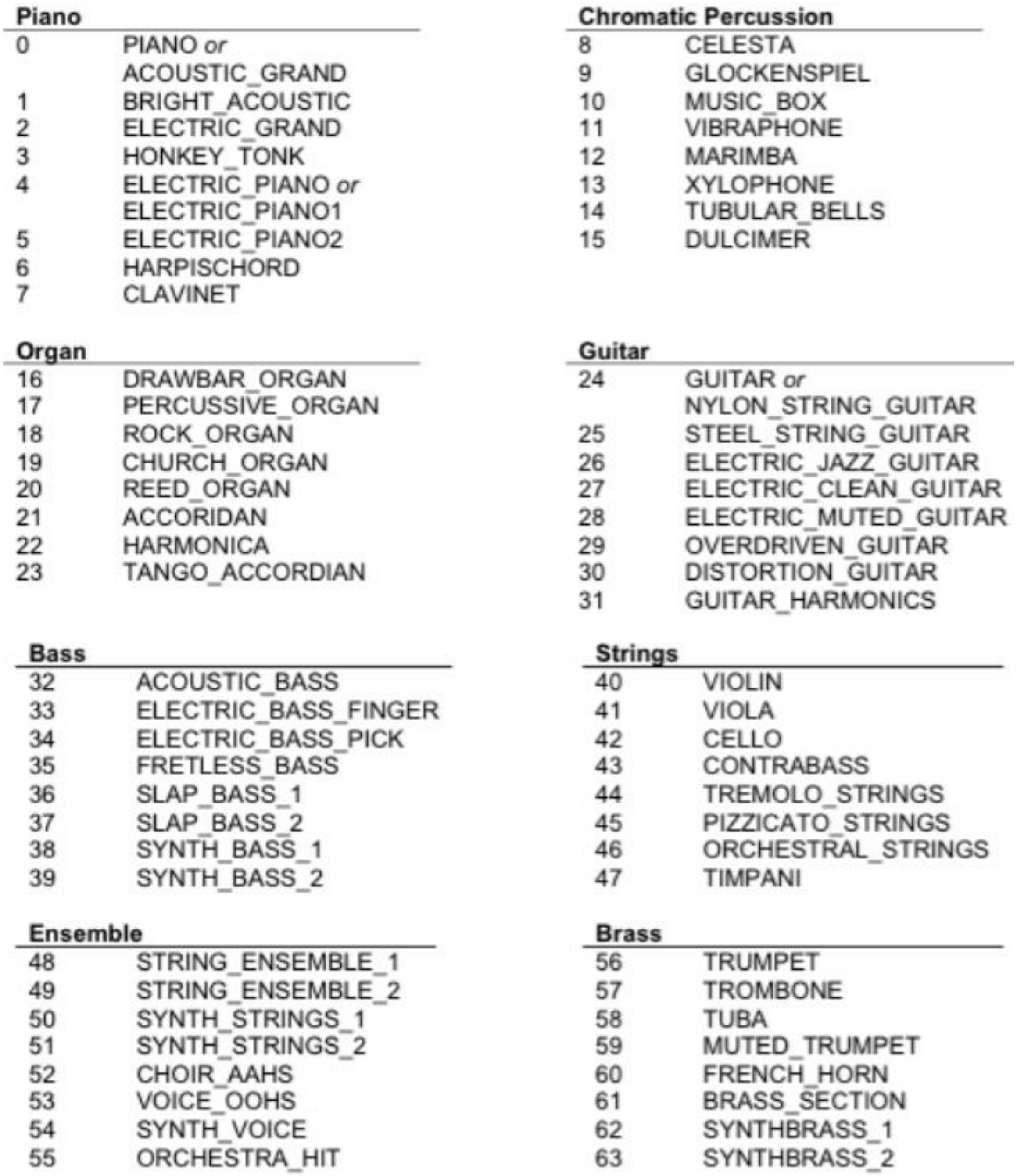


[Story]→[Load Color Shapes].

These two are the default art pieces I decided to use. I purposely made the shapes drawn on the canvas a bit hard to distinguish from the background to give the illusion that it's actually part of the piece. So now, from left to right, simply just click on the different shapes in sequence to view and hear back what you previously configured.

Attachments

Instruments allowed:



Piano	
0	PIANO *or* ACOUSTIC_GRAND
1	BRIGHT_ACOUSTIC
2	ELECTRIC_GRAND
3	HONKEY_TONK
4	ELECTRIC_PIANO *or* ELECTRIC_PIANO1
5	ELECTRIC_PIANO2
6	HARPISCHORD
7	CLAVINET

Chromatic Percussion	
8	CELESTA
9	GLOCKENSPIEL
10	MUSIC_BOX
11	VIBRAPHONE
12	MARIMBA
13	XYLOPHONE
14	TUBULAR_BELLS
15	DULCIMER

Organ	
16	DRAWBAR_ORGAN
17	PERCUSSIVE_ORGAN
18	ROCK_ORGAN
19	CHURCH_ORGAN
20	REED_ORGAN
21	ACCORIDAN
22	HARMONICA
23	TANGO_ACCORDIAN

Guitar	
24	GUITAR *or* NYLON_STRING_GUITAR
25	STEEL_STRING_GUITAR
26	ELECTRIC_JAZZ_GUITAR
27	ELECTRIC_CLEAN_GUITAR
28	ELECTRIC_MUTED_GUITAR
29	OVERDRIVEN_GUITAR
30	DISTORTION_GUITAR
31	GUITAR_HARMONICS

Bass	
32	ACOUSTIC_BASS
33	ELECTRIC_BASS_FINGER
34	ELECTRIC_BASS_PICK
35	FRETLESS_BASS
36	SLAP_BASS_1
37	SLAP_BASS_2
38	SYNTH_BASS_1
39	SYNTH_BASS_2

Strings	
40	VIOLIN
41	VIOLA
42	CELLO
43	CONTRABASS
44	TREMOLO_STRINGS
45	PIZZICATO_STRINGS
46	ORCHESTRAL_STRINGS
47	TIMPANI

Ensemble	
48	STRING_ENSEMBLE_1
49	STRING_ENSEMBLE_2
50	SYNTH_STRINGS_1
51	SYNTH_STRINGS_2
52	CHOIR_AAHS
53	VOICE_OOHS
54	SYNTH_VOICE
55	ORCHESTRA_HIT

Brass	
56	TRUMPET
57	TROMBONE
58	TUBA
59	MUTED_TRUMPET
60	FRENCH_HORN
61	BRASS_SECTION
62	SYNTHBRASS_1
63	SYNTHBRASS_2

Synth Effects

96	FX_RAIN *OR* RAIN
97	FX_SOUNDTRACK *or* SOUNDTRACK
98	FX_CRYSTAL *or* CRYSTAL
99	FX_ATMOSPHERE *or* ATMOSPHERE
100	FX_BRIGHTNESS *or* BRIGHTNESS
101	FX_GOBLINS *or* GOBLINS
102	FX_ECHOES *or* ECHOES
103	FX_SCI-FI *or* SCI-FI

Ethnic

104	SITAR
105	BANJO
106	SHAMISEN
107	KOTO
108	KALIMBA
109	BAGPIPE
110	FIDDLE
111	SHANAI

Percussive

112	TINKLE_BELL
113	AGOGO
114	STEEL_DRUMS
115	WOODBLOCK
116	TAIKO_DRUM
117	MELODIC_TOM
118	SYNTH_DRUM
119	REVERSE_CYMBAL

Sound Effects

120	GUITAR_FRET_NOISE
121	BREATH_NOISE
122	SEASHORE
123	BIRD_TWEET
124	TELEPHONE_RING
125	HELICOPTER
126	APPLAUSE
127	GUNSHOT

Reed

64	SOPRANO_SAX
65	ALTO_SAX
66	TENOR_SAX
67	BARITONE_SAX
68	OBOE
69	ENGLISH_HORN
70	BASSOON
71	CLARINET

Pipe

72	PICCOLO
73	FLUTE
74	RECORDER
75	PAN_FLUTE
76	BLOWN_BOTTLE
77	SKAKUHACHI
78	WHISTLE
79	OCARINA

Synth Lead

80	LEAD_SQUARE *or* SQUARE
81	LEAD_SAWTOOTH *or* SAWTOOTH
82	LEAD_CALLIOPE *or* CALLIOPE
83	LEAD_CHIFF *or* CHIFF
84	LEAD_CHARANG *or* CHARANG
85	LEAD_VOICE *or* VOICE
86	LEAD_FIFTHS *or* FIFTHS
87	LEAD_BASSLEAD *or* BASSLEAD

Synth Pad

88	PAD_NEW_AGE *or* NEW_AGE
89	PAD_WARM *or* WARM
90	PAD_POLYSYNTH *or* POLYSYNTH
91	PAD_CHOIR *or* CHOIR
92	PAD_BOWED *or* BOWED
93	PAD_METALLIC *or* METALLIC
94	PAD_HALO *or* HALO
95	PAD_SWEEP *or* SWEEP