CULMINATING EXPERIENCE
FINAL DOCUMENTATION

M.M. (Music Technology Innovation)

Austin Har 763777

CONTENTS

1. Introduction

"*The enemy of art is the absence of limitations*," Orson Welles.

At Berklee, I found myself surrounded with performers, composers and technologists from around the world. Each had their unique experiences and perspectives towards music, its cultural value and future trajectories amidst new and groundbreaking technologies. It became clear to me that I wanted my Culminating Experience to contribute to this creative zeitgeist by exploring the innovative possibilities of music technology as I became aware of the importance of capturing and reliving our emotional journeys and inspirations. Thus, as I conversed with my fellow colleagues, I was inspired by their abundance of creative energy and I wanted to create something that would be beneficial to our artistic endeavors.

As I learned new software for my coursework, I began to ponder how the songwriting process had evolved over time with new technologies. What tools did songwriters use to write music today? How did they transfer ideas and files from one place to another? How do we capture that moment of inspiration that would eventually lead to the creation of a work of art? These were the questions that I found myself seeking the answers to as I conceptualized my project. Throughout my investigation, I found room for innovation and meaningful contribution to the field of music technology in the project that I decided to commit my efforts to.

2. Description of Culminating Experience

"For creativity on the go, capture your inspiration with FLO."

'FLO' is an iOS app for the everyday songwriter who needs a simple interface for organizing their daily creativity and capturing that moment of inspiration. Professional and amateur songwriters, composers and music creators will be able to use 'FLO' on their iPhone or iPad to record, store and transfer their everyday ideas online as MIDI files. Whether you are taking a stroll in the park or sitting at home watching TV, songwriters can capture their moment of inspiration and transfer it online for further development.

(Working logo for 'FLO')

'FLO' contains 3 essential tools for capturing your ideas:

1. Drum machine
2. Synthesizer
3. Recorder (via the iPhone / iPad microphone)

These three features were designed with the utmost consideration for the modern songwriter's needs and intuitions during their creative process. The various capabilities of each feature are listed below:

1. **Drum Machine:** - 32 step drum sequencer for kick, snare, hi hat and other samples.
   - Beat quantization patch to the nearest 8th note value for real time recordings.
   - MIDI export and online upload onto Parse server for further development on other DAWs such as Ableton and Pro Tools.

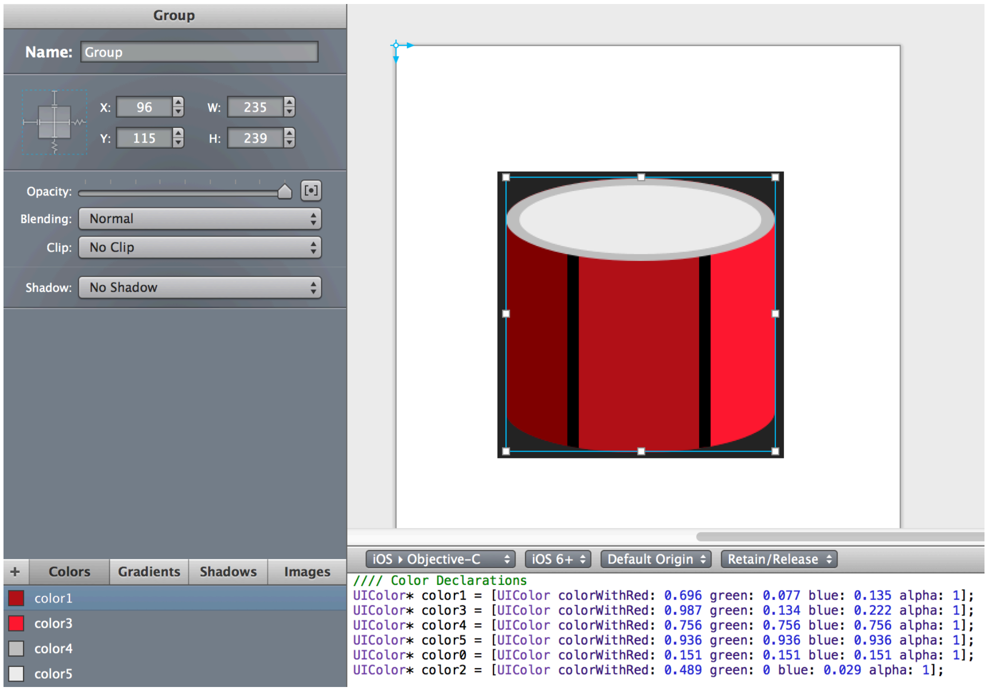
2. **Synthesizer:** - Integration into Pure Data patch for real time synthesis.
   - Beat quantization patch to the nearest 8th note value for real time recordings.

- MIDI export and online upload onto Parse server for further development on other DAWs such as Ableton and Pro Tools.

3. **Recorder:**
   - Integration into Pure Data patch for Audio to MIDI conversion.
   - MIDI export and online upload onto Parse server for further development on other DAWs such as Ableton and Pro Tools.

UI Design

The UI design of 'FLO' began from early prototypes in MAX MSP. As I prototyped the final designs with the standard Xcode storyboard, I discovered a very useful OSX application called 'PaintCode'.



Name: Group
X: 96 W: 235
Y: 115 H: 239
Opacity:
Blending: Normal
Clip: No Clip
Shadow: No Shadow

```
//// Color Declarations
UIColor* color1 = [UIColor colorWithRed: 0.696 green: 0.077 blue: 0.135 alpha: 1];
UIColor* color3 = [UIColor colorWithRed: 0.987 green: 0.134 blue: 0.222 alpha: 1];
UIColor* color4 = [UIColor colorWithRed: 0.756 green: 0.756 blue: 0.756 alpha: 1];
UIColor* color5 = [UIColor colorWithRed: 0.936 green: 0.936 blue: 0.936 alpha: 1];
UIColor* color0 = [UIColor colorWithRed: 0.151 green: 0.151 blue: 0.151 alpha: 1];
UIColor* color2 = [UIColor colorWithRed: 0.489 green: 0 blue: 0.029 alpha: 1];
```

(My drum icon design in 'PaintCode')

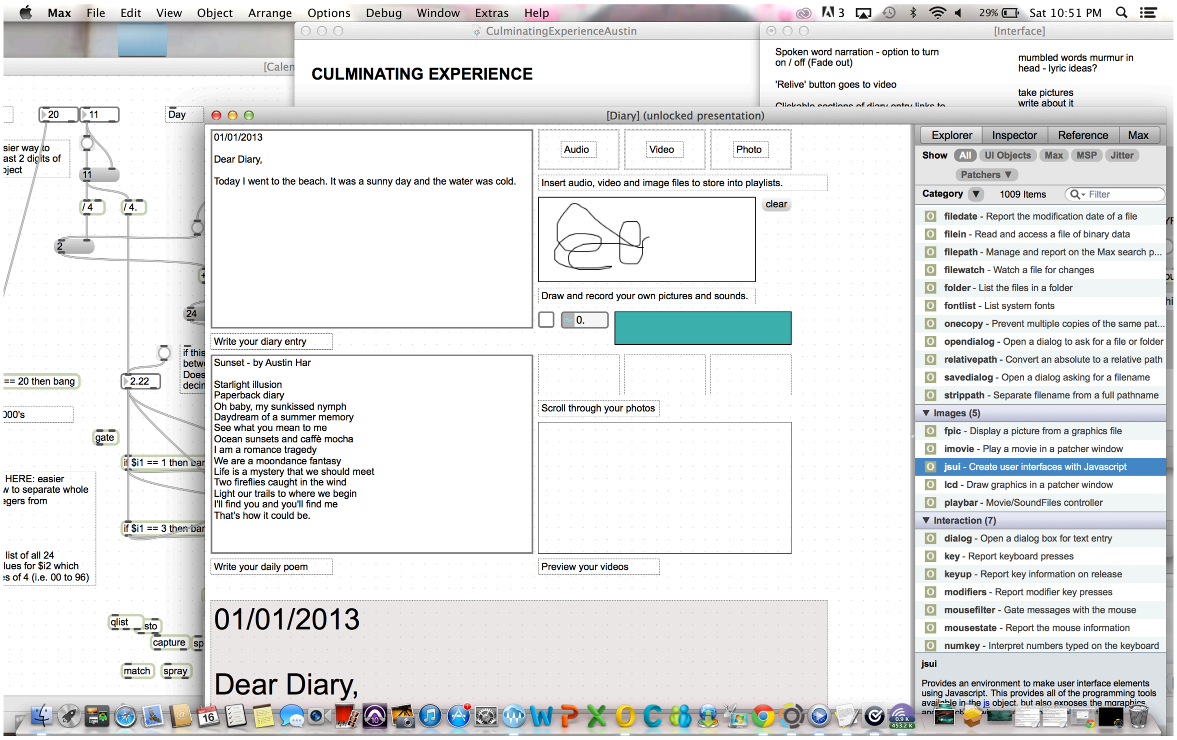
'PaintCode' is a powerful application that allows users to create intricate icons, buttons or interfaces, or alternatively to import svg and Photoshop files, that are automatically converted into usable Objective C code.

By drawing the icons, buttons and UI as Objective C code, rather than using conventional .png files, this saves a significant amount of storage space in the app.

Programming Languages Used

For the significant programming portion of the app, I used three primary languages:

1. **MAX MSP:** - Object oriented music technology programming language.
   - Original working prototypes of calendar and diary storage functions.



CULMINATING EXPERIENCE

Spoken word narration - option to turn on / off (Fade out)

'Relive' button goes to video

Clickable sections of diary entry links to

mumbled words murmur in head - lyric ideas?

take pictures
write about it

sier way to
ast 2 digits of
oject

if this
betw
Does
decin

HERE: easier
w to separate whole
egers from

list of all 24
lues for $i2 which
es of 4 (i.e. 00 to 96)

01/01/2013

Dear Diary,

Today I went to the beach. It was a sunny day and the water was cold.

Write your diary entry

Sunset - by Austin Har

Starlight illusion
Paperback diary
Oh baby, my sunkissed nymph
Daydream of a summer memory
See what you mean to me
Ocean sunsets and caffè mocha
I am a romance tragedy
We are a moondance fantasy
Life is a mystery that we should meet
Two fireflies caught in the wind
Light our trails to where we begin
I'll find you and you'll find me
That's how it could be.

Write your daily poem

Insert audio, video and image files to store into playlists.

Draw and record your own pictures and sounds.

Scroll through your photos

Preview your videos

01/01/2013

Dear Diary,

filedate - Report the modification date of a file
filein - Read and access a file of binary data
filepath - Manage and report on the Max search p...
filewatch - Watch a file for changes
folder - List the files in a folder
fontlist - List system fonts
onecopy - Prevent multiple copies of the same pat...
opendialog - Open a dialog to ask for a file or folder
relativepath - Convert an absolute to a relative path
savedialog - Open a dialog asking for a filename
strippath - Separate filename from a full pathname

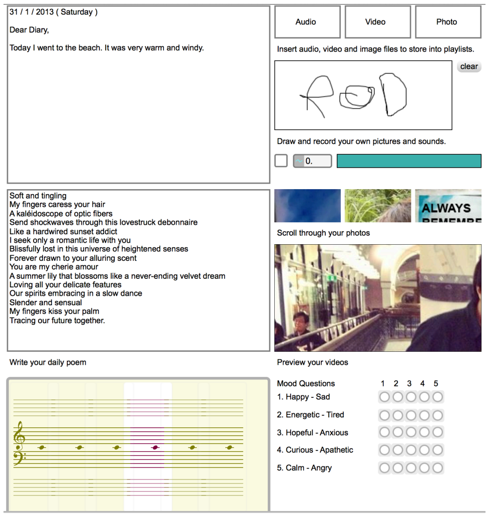
▼ Images (5)
fpic - Display a picture from a graphics file
imovie - Play a movie in a patcher window
jsui - Create user interfaces with Javascript
lcd - Draw graphics in a patcher window
playbar - Movie/SoundFiles controller

▼ Interaction (7)
dialog - Open a dialog box for text entry
key - Report keyboard presses
keyup - Report key information on release
modifiers - Report modifier key presses
mousefilter - Gate messages with the mouse
mousestate - Report the mouse information
numkey - Interpret numbers typed on the keyboard

jsui

Provides an environment to make user interface elements using Javascript. This provides all of the programming tools available in the js object, but also exposes the mgraphics

(Early diary interface prototype)



31 / 1 / 2013 (Saturday)

Dear Diary,

Today I went to the beach. It was very warm and windy.

Insert audio, video and image files to store into playlists.

Draw and record your own pictures and sounds.

Soft and tingling
My fingers caress your hair
A kaléidoscope of optic fibers
Send shockwaves through this lovestruck debonnaire
Like a hardwired sunset addict
I seek only a romantic life with you
Blissfully lost in this universe of heightened senses
Forever drawn to your alluring scent
You are my cherie amour
A summer lily that blossoms like a never-ending velvet dream
Loving all your delicate features
Our spirits embracing in a slow dance
Slender and sensual
My fingers kiss your palm
Tracing our future together.

Write your daily poem

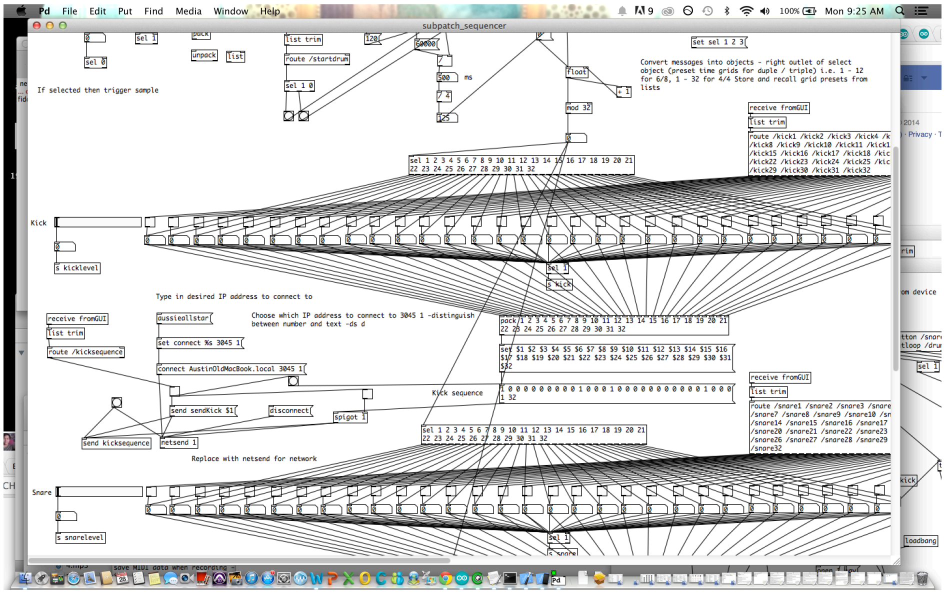
Scroll through your photos

Preview your videos

Mood Questions	1	2	3	4	5
1. Happy - Sad	○	○	○	○	○
2. Energetic - Tired	○	○	○	○	○
3. Hopeful - Anxious	○	○	○	○	○
4. Curious - Apathetic	○	○	○	○	○
5. Calm - Angry	○	○	○	○	○

(Early diary interface prototype)

2. **Pure Data:** - Object oriented music technology programming language with iOS function.
- Drum machine, BPM, Audio to MIDI converter, Beat Quantization, Synthesizer, MIDI file making and Xcode Control patches.



If selected then trigger sample

Convert messages into objects - right outlet of select object (preset time grids for duple / triple) i.e. 1 - 12 for 6/8, 1 - 32 for 4/4 Store and recall grid presets from lists

Type in desired IP address to connect to

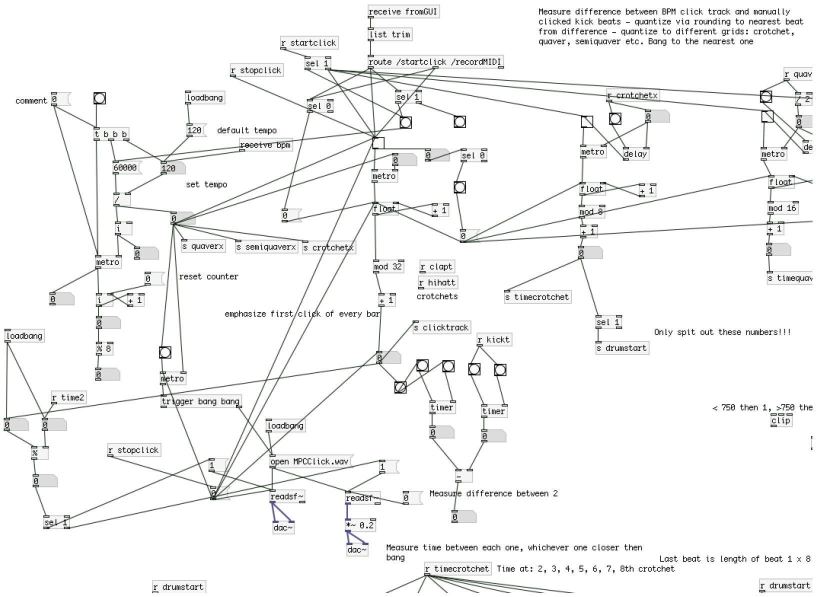
Choose which IP address to connect to 3045 1 -distinguish between number and text -ds d

Replace with netsend for network

Pd
File
Edit
Put
Find
Media
Window
Help
Mon 9:25 AM
subpatch_sequencer
sel 1
sel 0
unpack
list
list trim
route /startdrum
sel 1 0
120
60000
/
500
ms
/ 4
125
float
+ 1
mod 32
set sel 1 2 3
receive fromGUI
list trim
route /kick1 /kick2 /kick3 /kick4 /kick8 /kick9 /kick10 /kick11 /kick15 /kick16 /kick17 /kick18 /kick22 /kick23 /kick24 /kick25 /kick29 /kick30 /kick31 /kick32
sel 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Kick
s kicklevel
sel 1
s kick
receive fromGUI
list trim
route /kicksequence
aussieallstar
set connect %s 3045 1
connect AustinOldMacBook.local 3045 1
send sendKick $1
disconnect
spigot 1
send kicksequence
netsend 1
Kick sequence
pack 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
set $1 $2 $3 $4 $5 $6 $7 $8 $9 $10 $11 $12 $13 $14 $15 $16 $17 $18 $19 $20 $21 $22 $23 $24 $25 $26 $27 $28 $29 $30 $31 $32
1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 32
receive fromGUI
list trim
route /snare1 /snare2 /snare3 /snare7 /snare8 /snare9 /snare10 /snare14 /snare15 /snare16 /snare17 /snare20 /snare21 /snare22 /snare23 /snare26 /snare27 /snare28 /snare29 /snare32
sel 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Snare
s snarelevel
sel 1
s snare
loadbang

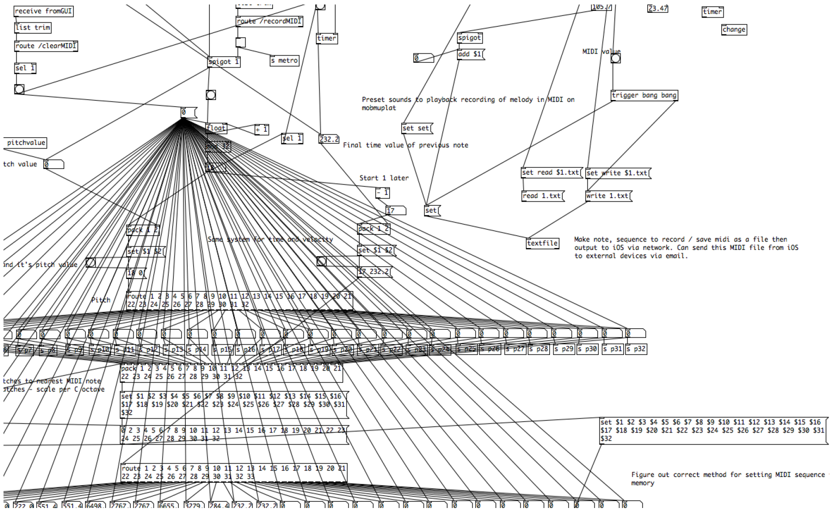
(Drum machine patch)

My Drum Machine patch features a 32-step drum sequencer, which receives a message of ‘1’ or ‘0’ for each designated kick, snare or hi hat selection. The patch receives the BPM float designated by the user. Listening to the click track, the user records their drum sequence and is able to save, playback and export it as a MIDI file.


receive fromGUI
list trim
route /startclick /recordMIDI
r startclick
r stopclick
sel 1
sel 0
loadbang
comment
t b b b
120
default tempo
receive bpm
60000
set tempo
metro
float
+ 1
mod 32
mod 8
mod 16
sel 0
s quaverx
s semiquaverx
s crotchetx
r crotchetx
r quav
delay
reset counter
emphasize first click of every bar
r clapt
r hihatt
crotchets
s timecrotchet
s timequav
s clicktrack
r kickt
sel 1
s drumstart
Only spit out these numbers!!!
% 8
loadbang
r time2
trigger bang bang
r stopclick
loadbang
open MPCClick.wav
readsf~
dac~
*~ 0.2
timer
Measure difference between 2
< 750 then 1, >750 the
clip
Measure time between each one, whichever one closer then bang
Last beat is length of beat 1 x 8
r timecrotchet
Time at: 2, 3, 4, 5, 6, 7, 8th crotchet
r drumstart
Measure difference between BPM click track and manually clicked kick beats - quantize via rounding to nearest beat from difference - quantize to different grids: crotchet, quaver, semiquaver etc. Bang to the nearest one

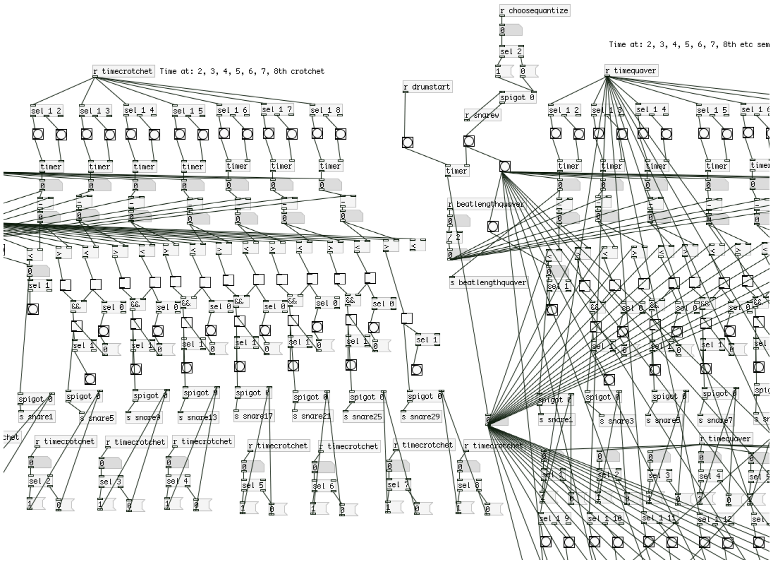
(BPM patch)

My BPM patch converts the designated BPM into milliseconds for the real time beat quantization with the click track and 32-step drum machine grid. This patch and its corresponding click track are crucial to the user's recording process, whether that be with the drum machine, synthesizer or recorder.


receive fromGUI
list trim
route /clearMIDI
sel 1
route /recordMIDI
spigot 1
s metro
timer
spigot
add $1
timer
change
MIDI value
trigger bang bang
Preset sounds to playback recording of melody in MIDI on mobmuplat
set set
Final time value of previous note
Start 1 later
set read $1.txt
set write $1.txt
read 1.txt
write 1.txt
textfile
Make note, sequence to record / save midi as a file then output to iOS via network. Can send this MIDI file from iOS to external devices via email.
pitchvalue
pack 1 2
set $1 $2
Same system for time and velocity
Pitch
route 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
s p27
s p28
s p29
s p30
s p31
s p32
set $1 $2 $3 $4 $5 $6 $7 $8 $9 $10 $11 $12 $13 $14 $15 $16 $17 $18 $19 $20 $21 $22 $23 $24 $25 $26 $27 $28 $29 $30 $31 $32
Figure out correct method for setting MIDI sequence memory

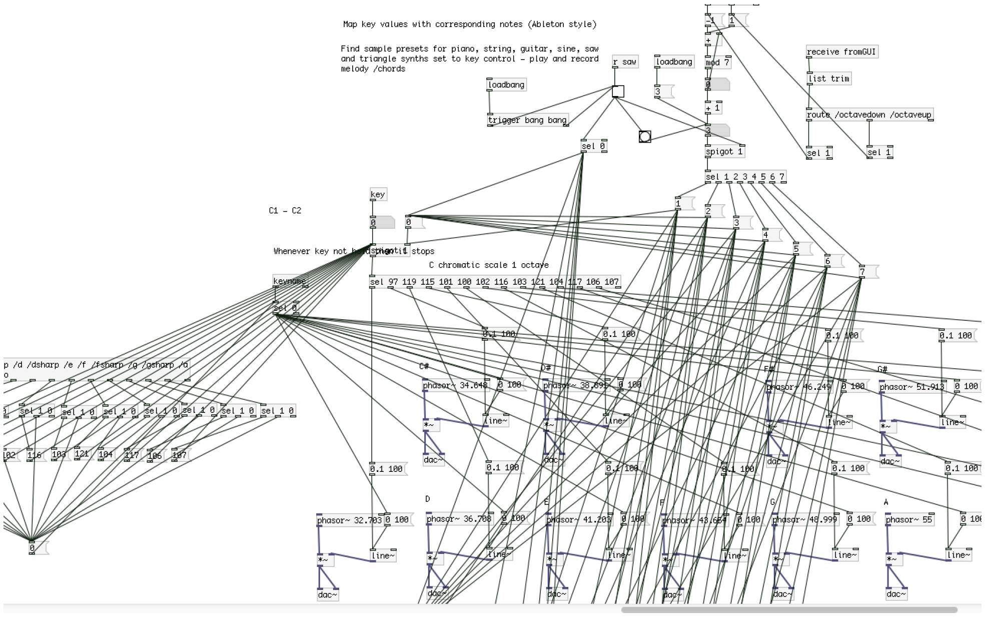
(Audio to MIDI converter patch)

My Audio to MIDI converter patch is perhaps the most ambitious one I programmed in Pure Data. It utilizes the 'fiddle' object to analyze incoming audio data into frequencies and MIDI pitch values. Using an amplitude filter (to cancel background ambience) and the 'timer' object, I was able to deduce the duration of each incoming note. The patch converts an audio recording of a guitar melody of up to 32 distinct notes, where the user is then able to save, playback and export it as a MIDI file.



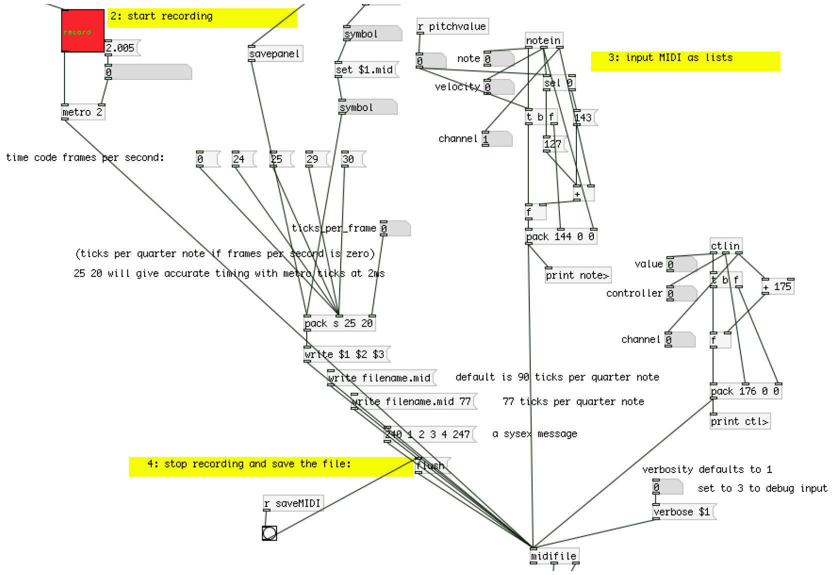
(Beat Quantization patch)

My Beat Quantization patch receives the millisecond duration of each ‘beat’ from the BPM patch, which allows it to quantize real time inputs to the nearest 8th note. Using the ‘timer’ and ‘&&’ objects, the logic behind my beat quantization patch lies in dividing each beat duration in half and then filtering those inputs which are less than half a beat early or late to the nearest beat value. The quantization is visible in the resulting Drum Machine patch where the quantized beats are marked as selected toggles.


Map key values with corresponding notes (Ableton style)
Find sample presets for piano, string, guitar, sine, saw and triangle synths set to key control - play and record melody /chords
loadbang
trigger bang bang
r saw
loadbang
mod 7
spigot 1
sel 1 2 3 4 5 6 7
receive fromGUI
list trim
route /octavedown /octaveup
key
C1 - C2
Whenever key not held, note stops
C chromatic scale 1 octave
keyname
sel 97 119 115 101 100 102 116 103 121 104 117 106 107
phasor~ 34.648
phasor~ 32.703
phasor~ 36.708
phasor~ 41.203
phasor~ 46.249
phasor~ 48.999
phasor~ 51.913
phasor~ 55
line~
dac~
C#
D
E
F
F#
G
G#
A

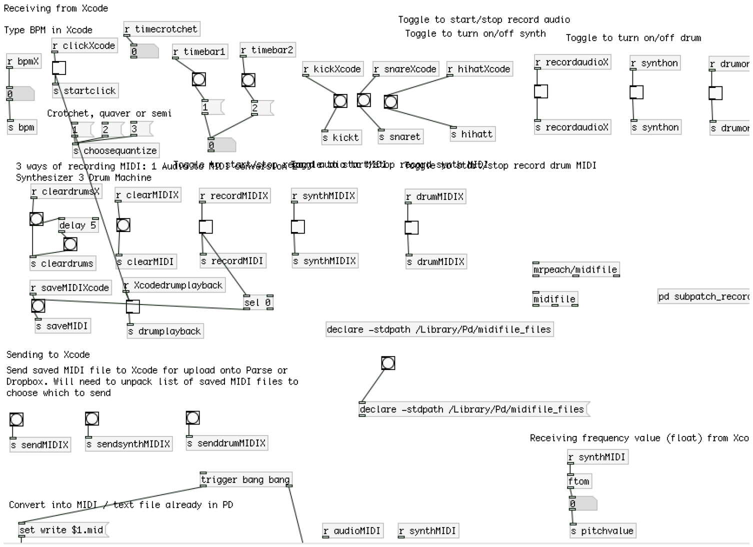
(Synthesizer patch)

My Synthesizer patch allows the user to select a sine, square or triangle wave (or various combinations of the three). The original prototype utilized the ‘keyin’ object, which tracked the user’s ‘keydown’ events from the letters A to K (representing an octave from C). The final synthesizer patch receives messages from the Xcode storyboard corresponding to a frequency to MIDI key, which in turn triggers the corresponding pitch selected from the original system. The user is then able to save, playback and export it as a MIDI file.



(MIDI file making patch)

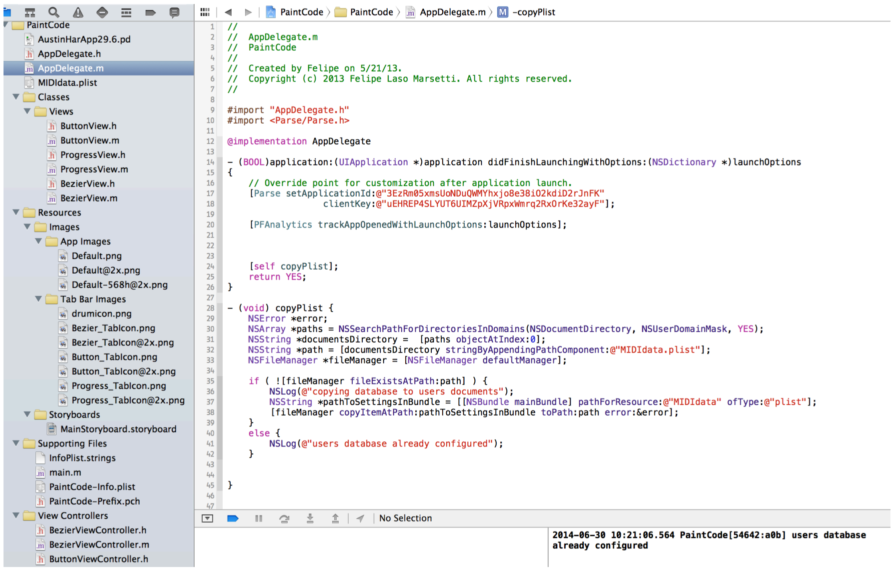
The ‘midifile’ object is a crucial part of my app’s functionality. This is the patch that collects all the MIDI pitch and duration data from the drum machine, synthesizer and recorder, and then compiles it into a .mid file. ‘Midifile’ is a Pure Data external which required a very extensive implementation and compilation process. I modified this patch to automatically update the user id of each MIDI recording.



(Xcode Control patch)

My Xcode Control patch is the main patch for receiving and sending data between Pure Data and Xcode via the libpd library. Floats, messages and symbols are transferred here to other Pure Data patches such as the Drum Machine and Synthesizer, whilst the '.mid' file compiled in the 'midifile' patch is sent to Xcode to be saved in the app's document's directory folder and also uploaded online via the Parse server API.

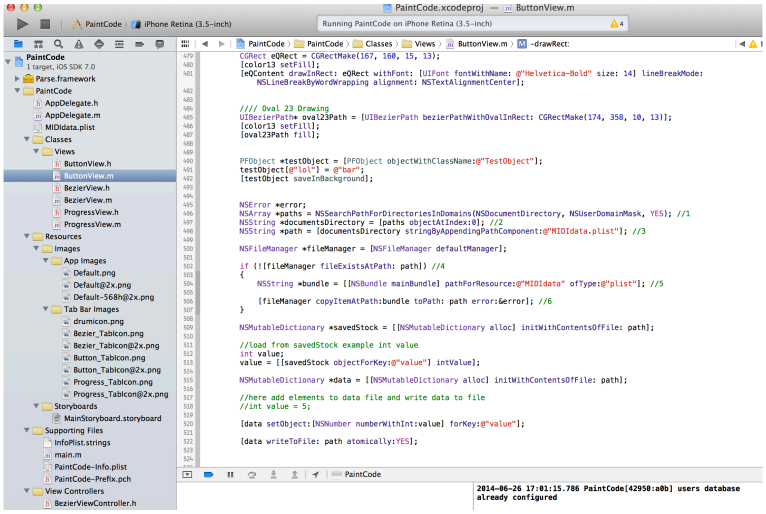
3. **Objective C:** - C based programming language in Xcode used for iOS development
   - Drum Machine, Synthesizer, Recorder, PaintCode UI, Document's Directory, libpd library, .plist resource and Parse API.



```
//
//  AppDelegate.m
//  PaintCode
//
//  Created by Felipe on 5/21/13.
//  Copyright (c) 2013 Felipe Laso Marsetti. All rights reserved.
//

#import "AppDelegate.h"
#import <Parse/Parse.h>

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
    [Parse setApplicationId:@"3EzRm05xmsUoNDuQWMYhxjo8e38iO2kdiD2rJnFK"
                  clientKey:@"uEHREP4SLYUT6UIMZpXjVRpxWmrq2RxOrKe32ayF"];

    [PFAnalytics trackAppOpenedWithLaunchOptions:launchOptions];



    [self copyPlist];
    return YES;
}

- (void) copyPlist {
    NSError *error;
    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
    NSString *documentsDirectory =  [paths objectAtIndex:0];
    NSString *path = [documentsDirectory stringByAppendingPathComponent:@"MIDIdata.plist"];
    NSFileManager *fileManager = [NSFileManager defaultManager];

    if ( ![fileManager fileExistsAtPath:path] ) {
        NSLog(@"copying database to users documents");
        NSString *pathToSettingsInBundle = [[NSBundle mainBundle] pathForResource:@"MIDIdata" ofType:@"plist"];
        [fileManager copyItemAtPath:pathToSettingsInBundle toPath:path error:&error];
    }
    else {
        NSLog(@"users database already configured");
    }


}
```

```
2014-06-30 10:21:06.564 PaintCode[54642:a0b] users database
already configured
```

(App Delegate .m file)

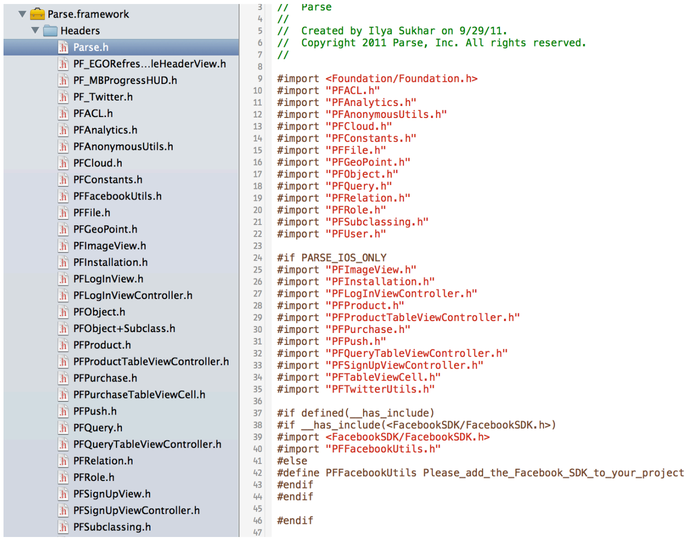


```
        CGRect eQRect = CGRectMake(167, 160, 15, 13);
        [color13 setFill];
        [eQContent drawInRect: eQRect withFont: [UIFont fontWithName: @"Helvetica-Bold" size: 14] lineBreakMode:
            NSLineBreakByWordWrapping alignment: NSTextAlignmentCenter];


        //// Oval 23 Drawing
        UIBezierPath* oval23Path = [UIBezierPath bezierPathWithOvalInRect: CGRectMake(174, 358, 10, 13)];
        [color13 setFill];
        [oval23Path fill];


        PFObject *testObject = [PFObject objectWithClassName:@"TestObject"];
        testObject[@"lol"] = @"bar";
        [testObject saveInBackground];


        NSError *error;
        NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES); //1
        NSString *documentsDirectory = [paths objectAtIndex:0]; //2
        NSString *path = [documentsDirectory stringByAppendingPathComponent:@"MIDIdata.plist"]; //3

        NSFileManager *fileManager = [NSFileManager defaultManager];

        if (![fileManager fileExistsAtPath: path]) //4
        {
            NSString *bundle = [[NSBundle mainBundle] pathForResource:@"MIDIdata" ofType:@"plist"]; //5

            [fileManager copyItemAtPath:bundle toPath: path error:&error]; //6
        }

        NSMutableDictionary *savedStock = [[NSMutableDictionary alloc] initWithContentsOfFile: path];

        //load from savedStock example int value
        int value;
        value = [[savedStock objectForKey:@"value"] intValue];

        NSMutableDictionary *data = [[NSMutableDictionary alloc] initWithContentsOfFile: path];

        //here add elements to data file and write data to file
        //int value = 5;

        [data setObject:[NSNumber numberWithInt:value] forKey:@"value"];

        [data writeToFile: path atomically:YES];
```

```
2014-06-26 17:01:15.786 PaintCode[42950:a0b] users database
already configured
```

(Document's Directory initialization code)

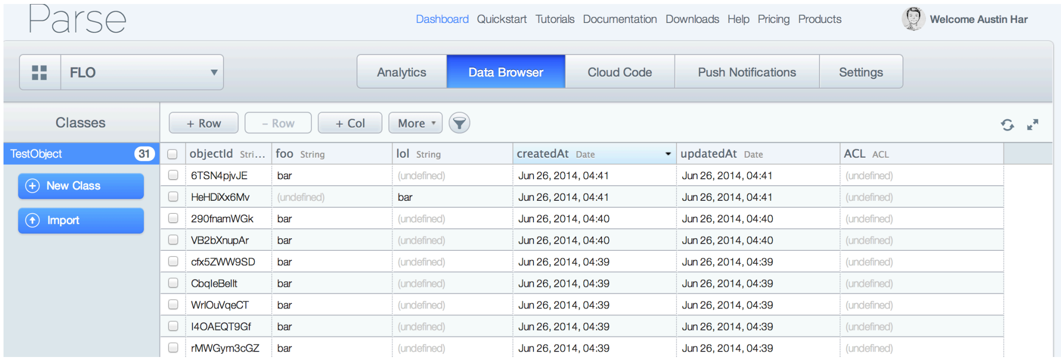
The Parse API and Document's Directory initialization code is called with the first method 'drawRect', along with the UI code from PaintCode. The NSBundle object allocates memory space for the MIDI data created in 'FLO' via the Pure Data 'midifile' object that is connected to Xcode through the libpd library. This data is then stored into the app's document's directory folder and uploaded into the Parse server via the PFObject function.



```
//  Parse
//
//  Created by Ilya Sukhar on 9/29/11.
//  Copyright 2011 Parse, Inc. All rights reserved.
//

#import <Foundation/Foundation.h>
#import "PFACL.h"
#import "PFAnalytics.h"
#import "PFAnonymousUtils.h"
#import "PFCloud.h"
#import "PFConstants.h"
#import "PFFile.h"
#import "PFGeoPoint.h"
#import "PFObject.h"
#import "PFQuery.h"
#import "PFRelation.h"
#import "PFRole.h"
#import "PFSubclassing.h"
#import "PFUser.h"

#if PARSE_IOS_ONLY
#import "PFImageView.h"
#import "PFInstallation.h"
#import "PFLogInViewController.h"
#import "PFProduct.h"
#import "PFProductTableViewController.h"
#import "PFPurchase.h"
#import "PFPush.h"
#import "PFQueryTableViewController.h"
#import "PFSignUpViewController.h"
#import "PFTableViewCell.h"
#import "PFTwitterUtils.h"

#if defined(__has_include)
#if __has_include(<FacebookSDK/FacebookSDK.h>)
#import <FacebookSDK/FacebookSDK.h>
#import "PFFacebookUtils.h"
#else
#define PFFacebookUtils Please_add_the_Facebook_SDK_to_your_project
#endif
#endif

#endif
```

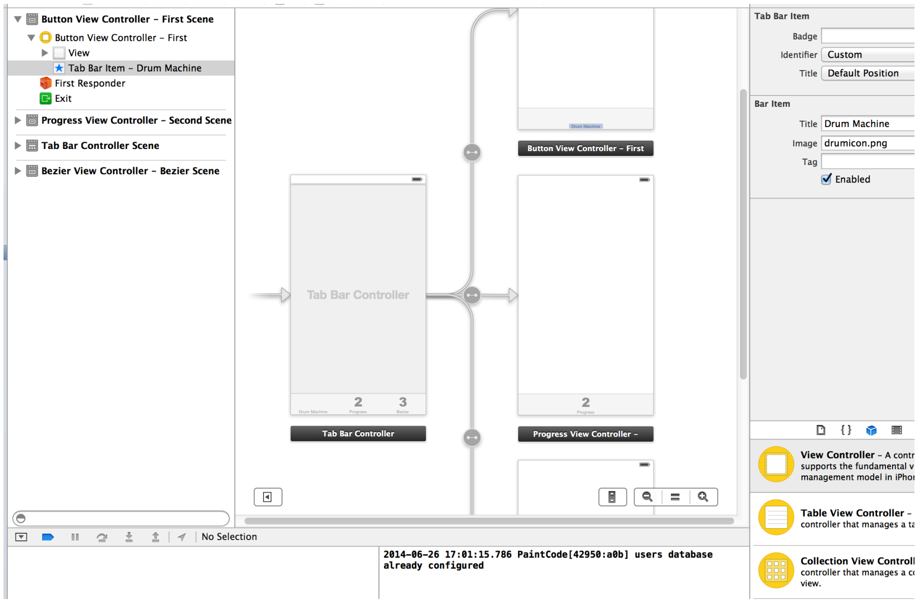
(Parse API Framework)

The Parse framework is imported into the relevant 'h' and 'm' files of the various views so that the MIDI data created in each of the functions (drum machine, synthesizer and recorder) will be able to be uploaded directly onto the server (as seen below).



□	objectId Stri…	foo String	lol String	createdAt Date	updatedAt Date	ACL ACL
□	6TSN4pjvJE	bar	(undefined)	Jun 26, 2014, 04:41	Jun 26, 2014, 04:41	(undefined)
□	HeHDiXx6Mv	(undefined)	bar	Jun 26, 2014, 04:41	Jun 26, 2014, 04:41	(undefined)
□	290fnamWGk	bar	(undefined)	Jun 26, 2014, 04:40	Jun 26, 2014, 04:40	(undefined)
□	VB2bXnupAr	bar	(undefined)	Jun 26, 2014, 04:40	Jun 26, 2014, 04:40	(undefined)
□	cfx5ZWW9SD	bar	(undefined)	Jun 26, 2014, 04:39	Jun 26, 2014, 04:39	(undefined)
□	CbqleBellt	bar	(undefined)	Jun 26, 2014, 04:39	Jun 26, 2014, 04:39	(undefined)
□	WrlOuVqeCT	bar	(undefined)	Jun 26, 2014, 04:39	Jun 26, 2014, 04:39	(undefined)
□	I4OAEQT9Gf	bar	(undefined)	Jun 26, 2014, 04:39	Jun 26, 2014, 04:39	(undefined)
□	rMWGym3cGZ	bar	(undefined)	Jun 26, 2014, 04:39	Jun 26, 2014, 04:39	(undefined)

(Parse server database log)


Button View Controller – First Scene
Button View Controller – First
View
Tab Bar Item – Drum Machine
First Responder
Exit
Progress View Controller – Second Scene
Tab Bar Controller Scene
Bezier View Controller – Bezier Scene
Tab Bar Controller
2
3
Tab Bar Controller
Button View Controller – First
2
Progress View Controller –
No Selection
View Controller – A contr
supports the fundamental v
management model in iPhon
Table View Controller –
controller that manages a ta
Collection View Controll
controller that manages a co
view.

Tab Bar Item

Badge

Identifier Custom

Title Default Position

Bar Item

Title Drum Machine

Image drumicon.png

Tag

☑ Enabled

```
2014-06-26 17:01:15.786 PaintCode[42950:a0b] users database
already configured
```

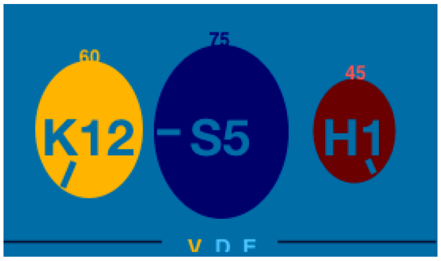
(iOS Storyboard)

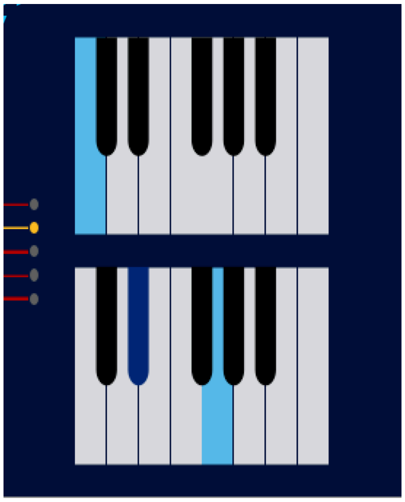
As I used ‘PaintCode’ for a code generated UI on each of these view controllers, I placed a ‘View’ object on each of the view controllers to receive and send events from the iOS screen. Essentially, the ‘View’ object acts as a screen for the ‘PaintCode’ UI to be displayed.

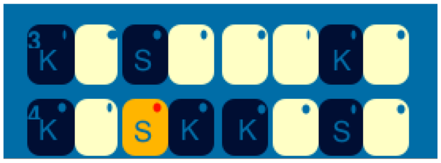
3. Innovative Aspects of the Work

‘FLO’ is innovative and differs in comparison to successful apps such as 'Day One', 'Live Journal' and ‘Evernote’ (note taking apps), Native Instruments ‘iMaschine’ and Propellerhead ‘Figure’ (beat making apps) in three fundamental aspects:

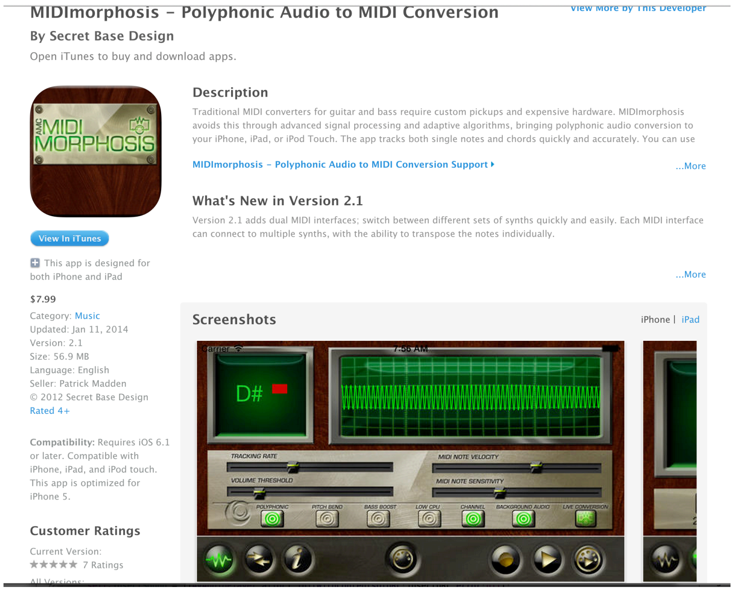
1. **Template for music creation, storage and transfer:** ‘FLO’ offers a music specific template for not only creating, but also storing and transferring ideas for further development on common DAWs such as Ableton or Pro Tools. Current beat making apps function by having the entire track produced on the app itself, then exporting it as a single bounced audio file. This only allows very limited options for further development on professional DAWs, whereas the MIDI capability of ‘FLO’ is a significant advantage for further creative development.







2. **Audio to MIDI conversion:** ‘FLO’ features a highly appealing Audio to MIDI conversion feature, which is not currently found in major music apps (and those which attempted it have received largely negative reviews). ‘MIDIMorphosis’ is by far the best-received iOS MIDI conversion app but is made specifically for guitar and bass only.



MIDImorphosis – Polyphonic Audio to MIDI Conversion

View More by This Developer

By Secret Base Design

Open iTunes to buy and download apps.

Description

Traditional MIDI converters for guitar and bass require custom pickups and expensive hardware. MIDImorphosis avoids this through advanced signal processing and adaptive algorithms, bringing polyphonic audio conversion to your iPhone, iPad, or iPod Touch. The app tracks both single notes and chords quickly and accurately. You can use

MIDImorphosis – Polyphonic Audio to MIDI Conversion Support ▸ ...More

What's New in Version 2.1

Version 2.1 adds dual MIDI interfaces; switch between different sets of synths quickly and easily. Each MIDI interface can connect to multiple synths, with the ability to transpose the notes individually.

...More

View In iTunes

This app is designed for both iPhone and iPad

$7.99

Category: Music
Updated: Jan 11, 2014
Version: 2.1
Size: 56.9 MB
Language: English
Seller: Patrick Madden
© 2012 Secret Base Design
Rated 4+

Compatibility: Requires iOS 6.1 or later. Compatible with iPhone, iPad, and iPod touch. This app is optimized for iPhone 5.

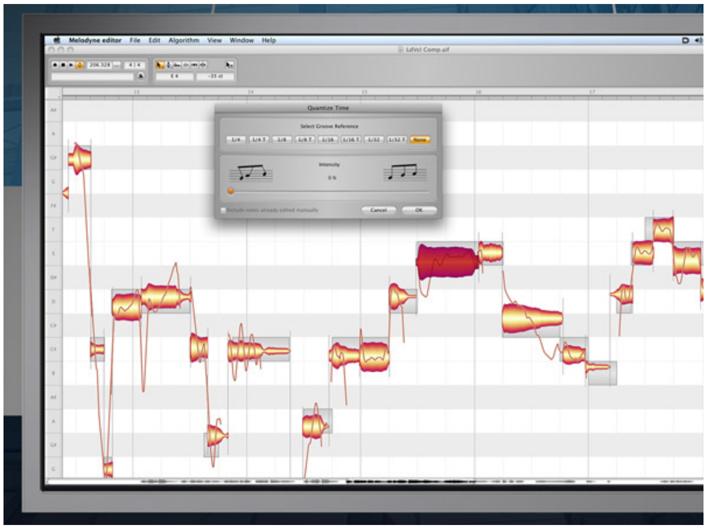
Customer Ratings

Current Version:
★★★★★ 7 Ratings

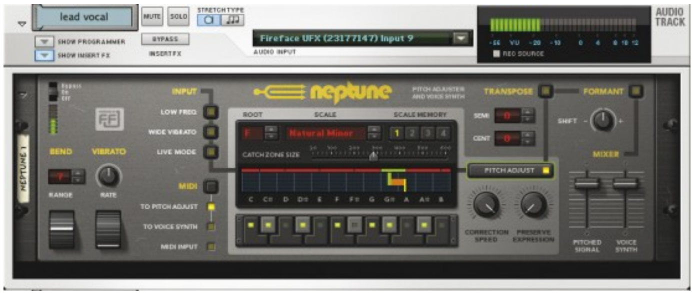
Screenshots iPhone | iPad

(MIDImorphosis on the App Store)

A raw guitar sound is a much purer tone than a human voice, thus there are significantly fewer pitch frequencies for the app to analyze. Furthermore, there are technological considerations regarding monophonic, homophonic and polyphonic analysis; the latter of which is also significantly easier with a pure tone such as guitar or bass. Whilst software such as Celemony's Melodyne and Neptune's Autotune are able to do Audio to MIDI conversion well, it is difficult to replicate their success on an iOS app due to the complexity of this technology and the audio quality of the iPhone and iPad microphone for recording raw sounds.



(Celemony Melodyne's pitch correction feature)

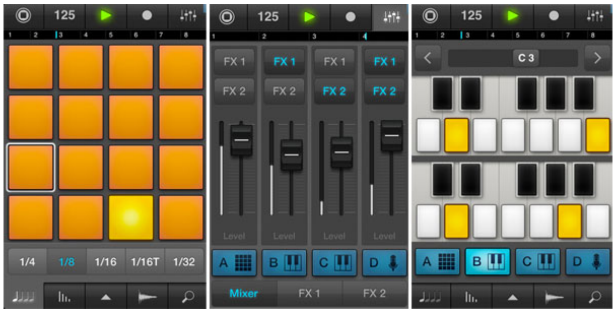


(Neptune Autotune's pitch correction feature)

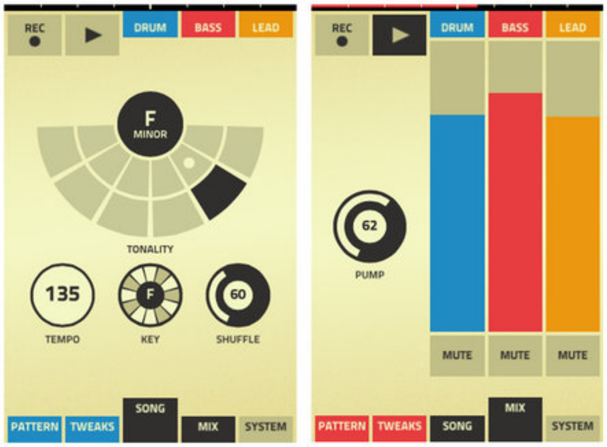
Audio to MIDI conversion involves complex amplitude and spectral analysis of frequency data from an audio source in order to determine a note's pitch, length and velocity. These processes require not only audio processing and mathematical operations, but also file management, storage and transfer, which require the combination of several programming languages. With 'FLO', I was required to implement Pure Data, Objective C and C++, along with their corresponding object libraries and foundations, libpd for Pure Data, Parse Foundation and AV Foundation for Xcode.

As of today, 'FLO' is able to replicate the efficiency of Audio to MIDI conversion with as much success as 'MIDIMorphosis' on guitar for its monophonic capabilities. As 'FLO' is predominantly designed for quick 'MIDI note taking', homophonic and polyphonic capabilities were not my top priority.

3. **MIDI note taking:** Unlike 'iMaschine' and 'Figure' which only transfer audio files via USB, 'FLO' can transfer MIDI files of beats and melodies as well audio recordings and photos online via the Parse API. MIDI files are superior to audio files in terms of size, transfer speed and flexibility for manipulation.

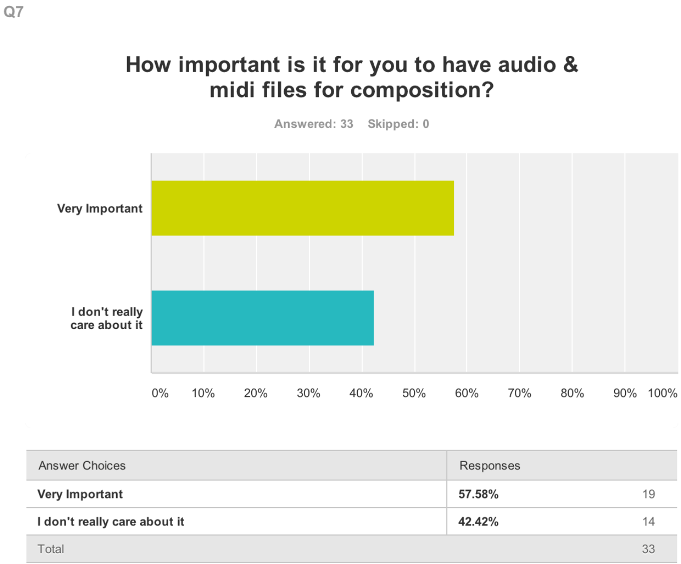


(Native Instrument iMaschine's UI design)



(Propellerhead Figure's UI design)

Based on my research with Berklee business majors Carl and Tanya into competing apps, we found few songwriting apps that compete with 'FLO's 'MIDI note taking' and Audio to MIDI conversion features.


Q7

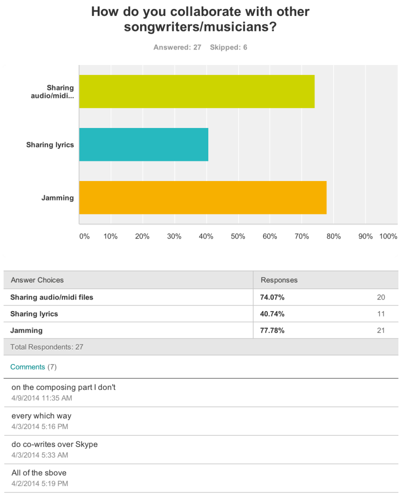
How important is it for you to have audio & midi files for composition?

Answered: 33 Skipped: 0

Answer Choices	Responses	
Very Important	57.58%	19
I don't really care about it	42.42%	14
Total		33

(Survey for 'FLO')

The survey that I conducted with my business partners confirmed the demand for these innovative features that 'FLO' offered. As seen in the survey responses, creating and sharing MIDI files proved to be a major aspect of compositional and collaboration process for modern day songwriters.



How do you collaborate with other songwriters/musicians?

Answered: 27 Skipped: 6

Answer Choices	Responses	
Sharing audio/midi files	**74.07%**	20
Sharing lyrics	**40.74%**	11
Jamming	**77.78%**	21
Total Respondents: 27		

Comments (7)

on the composing part I don't
4/9/2014 11:35 AM

every which way
4/3/2014 5:16 PM

do co-writes over Skype
4/3/2014 5:33 AM

All of the sbove
4/2/2014 5:19 PM

(Survey for ‘FLO’)

4. New Skills Acquired

I acquired many new skills for my Culminating Experience project. I had no programming experience prior to my studies at Berklee and I was required to learn several languages in order to program my iOS app:

1. MAX MSP
2. Pure Data
3. Objective C

It is also worth mentioning the different capacities in which I used these programming languages. Not only did I grow familiar with these programming languages, I also developed the fundamental skills and problem solving logic for conceptualizing, designing and executing an original iOS application from scratch:

1. Implementation of Parse API into Objective C
2. Beat Quantization logic
3. Audio to MIDI conversion logic
4. Real time synthesis
5. Implementation of external objects and libraries (‘midifile’ object and ‘libpd’ library)
6. Communication between several programming languages, object libraries, frameworks and servers

Asides from the new skills I gained in programming, I also gained new skills with software such as Ableton (which I had no prior experience), that provided the initial inspiration for 'FLO's MIDI note-taking function. Other skills included:

1. Independent research through online forums and tutorials
2. Time management skills
3. Presentation skills
4. Collaboration skills

---

5. Challenges Encountered

As my Culminating Experience project involved the extensive use of programming languages, which I had no prior experience with before coming to Berklee, I faced many challenges throughout the entire process. During the initial conceptualization stages, the biggest challenge I faced was the need to be innovative. This challenge was especially difficult given the many new and innovative technologies emerging for iOS applications today. With guidance from my program director Stephen Webber, advisor Ben Hogue and visiting professors David Mash and Richard Boulanger, I was able to incorporate the defining characteristics of an innovative product into my final app.

Upon reaching a focused objective for my project, the process was then complicated by our substantial coursework and my limited programming skills. Throughout the entire execution process, the steep learning curve required for improving one's programming skills became evident and I revised my earlier ambitious vision for more specific weekly goals given the short timeline. As I developed my skills, I was then able to develop a more realistic vision for my final application.

---

6. Future Ramifications for the Project

In our modern digital information age, recording personal experiences and sharing them with the world via social networking is increasingly expected of us. 'FLO' has the potential to contribute to this 21st century zeitgeist by allowing creative content to be recorded, stored and transferred as MIDI files online. This would open new and exciting doors for creative collaborations between ‘FLO’ users as their ideas could potentially be integrated as a feature on social networking platforms such as Facebook or Twitter via integrating the relevant API into its code.

As indicated in my survey, the majority of musicians collaborate with each other via sharing files such as MIDI, audio samples and text. This shows the potential for ‘FLO’ to blossom into a larger scale project with a team of programmers and established business partners (i.e. Ableton, Pro Tools, Facebook) to develop a useful and innovative format for everyday users to share their musical notes online. Furthermore, visiting artists such as DJ Guru stated that a MIDI note-taking iOS application could become the ‘standard’ for capturing musical ideas.

7. Conclusion

In summary, ‘FLO’ is designed to provide the utmost speed and efficiency in capturing the basic essential details of a songwriter’s daily inspirations. Through its 3 features, ‘FLO’ allows users to quickly record, store and transfer musical ideas as MIDI files onto DAWs, such as Ableton and Pro Tools, for further development through the Parse server API. From here, users can import that MIDI data onto their computer to start or continue working on a bigger project such as a musical composition or track.

Creating ‘FLO’ was a very challenging and fulfilling experience for me. Not only did I acquire a varied range of new skills, I was also able to open my mind to different perspectives of music technology as a developer and user. This was a new and ambitious project for me and I am thankful for all the inspiration and guidance that I received during my studies at Berklee.

7. Footnotes

The following are the main sources from which I developed my programming skills, along with consultations from my supervisor Ben Hogue:

Peter Brinkmann, 2012. *Real-Time Audio Synthesis on Android and iOS: Making Musical Apps.* O-Reilly Media Inc.

www.stackoverflow.com (online question and answer forum for programmers)

www.lynda.com

www.youtube.com

http://ipgames.wordpress.com/tutorials/writeread-data-to-plist-file/

http://www.raywenderlich.com/36341/paintcode-tutorial-dynamic-buttons

https://developer.apple.com/

https://www.parse.com/

http://en.flossmanuals.net/csound/e-csound-in-ios/

https://github.com/libpd/pd-for-ios/wiki/ios