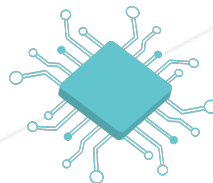# Sonic Pi

Turn Your Code Into Music!

Author: Elliot Tregoning etregoni@student.42.us.org

Summary: *Sonic Pi project of 42's HackHighSchool. No previous coding experience or music knowledge is required for this project, however it helps!*

Contents

Chapter I

Foreword

 Cromulon: SHOW ME WHAT YOU GOT.

Rick: I've seen enough of the galaxy to know that what we've got here is a Cromulon from the Cygnus-5 Expanse. So you can forget about nukes, and you can forget about math. This head won't go away until *burp* Earth shows them it's got a hit song.

Aide: You mean like Vivaldi?

Rick: No, Frasier. A live performance of a newly-written, catchy, original song. The Cromulon feed on the talent and showmanship of less-evolved lifeforms.

President: All right, all right. Thank you, Mr. Sanchez. Change of plan, people. Get me Pharrell, Randy Newman, Billy Corgan, and The-Dream. The-Dream? He wrote "Umbrella" and "Single Ladies"? You people haven't heard of The-Dream?

Aide: Sir! Pharrell, Newman, Corgan, and that Dream guy. They're all dead.

President: What? How is that possible? Do people just die when I name them?

Aide: The Grammys, sir. There was an earthquake and all the musicians... (sniffs, holding back tears) All the famous ones, they're gone.

President: Sanchez! Are you a musician?

Rick: I dabble, Mr. President.

President: Get this man and his grandson on a Blackhawk to Area 51.

...

Morty: Rick, are you really a musician?

Rick: Who's NOT a musician, Morty?

Morty: Me!

Rick: Yeah, not with that attitude... Alright, Morty, let's get ready to do it! Why don't you, uh, find a button on one of those keyboards and lay down some kind of beat?

Morty: Rick, I think we need to cut our losses. We get our family and portal out of here!

Rick: Morty! Good music comes from people who are relaxed. Just hit a button, Morty! Gimme a beat!

Morty: Oh man, ok, all right, um. . . *hits button on keyboard*

Rick: Ahhhhh yeahhhhh...You gotta get schwifty, You gotta get schwifty in heeeeere....

Chapter II

Introduction

In this project, you will be using a program called Sonic Pi, and writing in a language called SonicPiLang. Don't be intimidated by the unfamiliar name though, because it's essentially just the Ruby programming language, with a handful of Sonic Pi specific keywords and functions. Anything that works in Ruby will also work in the Sonic Pi program!

Chapter III

Goals

The goals of this project are to teach you how to think programatically, how functions and loops work, and most importantly to have fun! Your experience will depend on your level of knowledge..

Do you have zero experience in both programming and music? Great! This project will help you learn both at the same time. A simple melody is all that is required for this project, but we encourage you to get as creative as possible! There will be bonus points awarded for going above and beyond the expectations of the project!

Do you only have experience in music, or only have experience in programming? Good news! With the magic of Sonic Pi, your musical ability will help you improve your programming skills and vice versa!

Do you have a good amount of experience in both programming and music? That's cool too! You will be amazed what this program is capable of when you know what you're doing! It's truly addicting!

Chapter IV

General Instructions

In this project you will use the Sonic Pi program to write a song, or portion of a song. You can compose your own song, or you can recreate a song that you like. The requirements of this project may seem a bit intimidating at first, but once you get started you will quickly realize how easy it gets once you understand how things work! Every URL provided in this document has been included for a reason – READ IT! Or don't, I can't tell you what to do, but I really recommend that you read them.

Chapter V

Mandatory Part

V.0.1   General Requirements

- You may either write your own song, or write a cover of an already existing song.

- If you are covering a song, you do not need to cover the entire song, but your song should be at least 1 minute long.

- Your song must contain at least 3 instruments/effects – A good example would be: A backing track, a melody, and a drum beat, but you are free to choose your own!

- Your song must have different parts. You cannot just loop over the same notes for the entire song! This is okay for the backing track and drum beat, but not every instrument!

V.0.2   Code Requirements

- All code must be inside functions, with the exception of function calls. This means no global variables!

- The entry point of your song must be in a file named 'main' (main.rb)

- If using multiple buffers, you must put a comment at the top of the main telling how many buffers are included in the song, and give the names of the buffers.

- You must put a comment at the top of each file explaining what part of the song that file is responsible for.

- You must make use of loops. Any type of loops are fine!

- Your song must have an end condition, no infinite loops! This means the song must stop on it's own, without you pressing any buttons.

- You must make your own loops. This means you cannot use any samples that start with the word "loop"

Chapter VI

Bonus Part

VI.0.1 Some ideas for bonus points, but feel free to think of your own!

- Use of more than 3 instruments.

- Use of Ruby code that isn't part of SonicPi Lang (anything that isn't included in the sonic pi docs)

- Use of multiple buffers – Organize your code while you write it, and you won't even have to try for this bonus!

- Use of fx, such as reverb, flanger, etc.

- Use of synth and fx options such as 'amp', 'mix', 'cutoff', etc.

- Spend the extra effort to smoothly transition between effects and instruments.

- Be creative. Anything not included in this document may be defended for bonus points in your correction. It must be impressive to your corrector to get points though!

Chapter VII

Turn-in and peer-evaluation

Your project must be submitted as a directory named sonicpi, with all .rb files in this directory.
You will get the link to this repository when you subscribe to this project.

If you are reading this before starting the project, please stop what you are doing and follow these instructions!
Replace <your vogsphere repo> the repository link given for the project. (without the < >!)

```
$ git clone <your vogsphere repo> sonicpi_repo
$ cd sonicpi_repo && mkdir sonicpi && cd sonicpi
```

That's it. Just remember to push your project to the repository before marking it as finished!

If you have already started this project, and need to get your work into your repository, follow the following instructions, but next time remember to clone your repository before starting. It's less work!
First, open your terminal, and navigate to where you have your project saved, but do not go into the directory. Replace <your project dir> with the name of your project's directory, (without the < >!)

```
$ mkdir sonicpi_repo
$ mv <your project dir> sonicpi_repo/sonicpi && cd <your project dir>
$ git init
$ git add sonicpi
$ git remote add origin <your vogsphere repo>
$ git commit -m "first commit"
$ git push -u origin master
$ cd sonicpii_repo
```

From this point forward, all you will need to do is push your individual files every time you make changes! Good luck!

Chapter VIII

Hints / Examples

Links to hints, plus a bit of extra help will go here.

Most of the stuff you need to know is in the built-in documentation of the Sonic Pi program, but here are a few hints to help you along.

`Global Variable Example`

```
# main.rb - entry point
# includes 3 more files, lead.rb, drums.rb, and bass.rb
# this is a comment and above is an example of
# how to label your files!

# below is a global variable, do not use these!
global = "I am a global variable because I am outside of a
    function"

define :music_track do
    # please put your variables inside functions.
    variable = "I am not a global variable because I am in a
        function!"
end

# below is a function call, the only code that should be outside
    of functions!
music_track.
```

Here is an example of a built in Sonic Pi loop. This loop will play the note in the body 5 times.

```
5.times do
    # plays note the sound of the 60th key on a MIDI keyboard
    play 60
    # delays for 1 beat, dependent on bpm
    sleep 1
end
```

A few extras

```
# You may use numbers that correspond to a MIDI keyboard,
# or musical notation. Either works.
# Note 60 on a MIDI keyboard
play 60
# G sharp in the 5th octave
play :Gs5
# Example of a chord
play [:Ds5, :Ds4, :As4]
# Also a chord
play chord(:E3, :minor)
# Press the help button in the upper right corner of the
# program to view the docs.
```

The following is a short snippet of the example song I have created for this course. I won't tell you what any of it does. That's your job to find out! Copy/paste it into your Sonic Pi program and give it a run!

```ruby
# buff0.rb - Lead Buffer

define :lead do
    use_synth :piano
    use_synth_defaults note: 60, amp: 0.4
    with_fx :reverb, room: 0.8 do
            measure1 = [:Fs5, :Fs5, :Fs5, :Fs5, :Fs5, :Fs5, :Fs5, :As5
                , :As5, :As5]
            release1 = [1, 1, 1, 1, 1, 1, 0.5, 1, 0.5, 1]
            timing1 = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.25, 0.25, 0.25,
                0.25]
            measure2 = [:Ds6, :Ds6, :Ds6, :Ds6, :Ds6, :Ds6, :Cs6, :Cs6
                ]

            2.times do
                measure1.zip(timing1, release1).each do |m, t, r|
                    play m, release: r
                    sleep t
                end
                measure2.each do |m|
                    play m
                    sleep 0.5
                end
                timing1.each do |t|
                    play :As5
                    sleep t
                end
                8.times do
                    play :F5
                    sleep 0.5
                end
            end
    end
end
```

If you have any questions or problems, feel free to email me at etregoni@student.42.us - I WILL NOT help you with your project, and I WILL NOT help you cheat, but if you are stuck, or have an issue you cannot resolve, you can feel free to contact me.

Chapter IX

Troubleshooting / Correction Help

IX.0.1 How to install Sonic Pi

To install Sonic Pi, visit http://www.sonic-pi.net and follow these instructions when you get there:

```
When you get to this screen, click the macOS button.
```

```
After clicking on that button, click on the Download button here.
```

Once you have downloaded the program - and this is the most important part!
- Click on the file you downloaded. You should see a window like the following
image. To make things easy, drag this file on to your desktop, and let it
install. You can also put it anywhere in your home folder, but do not put
it in the Applications folder, because that requires an administrator's password!

IX.0.2   How to correct this project

Once you have installed Sonic Pi on your machine, open it, and then load each .rb file from their project, in separate buffers. If their project is all in one file, skip this section.

Once you have loaded all of the buffers, go to each buffer individually, and press the play button. (Command + R)

Finally, change to the buffer where you loaded their main.rb file, and press Command + R to start their song!

IX.0.3   Troubleshooting Tips

- Press Command + / to automatically comment a line.

- If your sound does not work, close the program and open it back up again. This often happens when you plug in an output device such as headphones, while Sonic Pi is already open. Do not forget to save your work!

- If you get any sort of errors when opening Sonic Pi, log off, and back on again. If this does not fix your problem, move the Sonic Pi icon to the trash, and install it again.

- Press the 'Help' button in the top right corner to show the documentation.

- If you have a problem that is not mentioned here, ask a mentor! If they do not know, you can email me or ask them to find me!

- Good luck!!!