Overview of available samples

AMBIENT

ambi_choir
ambi_dark_woosh
ambi_drone
ambi_glass_hum
ambi_glass_rub
ambi_haunted_hum
ambi_lunar_land
ambi_piano
ambi_soft_buzz
ambi_swoosh

BASS

Bass_dnb_f
bass_drop_c
bass_hard_c
bass_hit_c
bass_thick_c
bass_trance_c
bass_voxy_c
bass_voxy_hit_c
bass_woodsy_c

KICK DRUMS

bd_808
bd_ada
bd_boom
bd_fat
bd_gas
bd_haus
bd_klub
bd_pure
bd_sone
bd_tek
bd_zome
bd_zum

SNARE DRUMS

sn_dolf
sn_dub
sn_zome

DRUMS Acoustic

drum_bass_hard
drum_bass_soft
drum_cowbell
drum_cymbal_closed
drum_cymbal_hard
drum_cymbal_open
drum_cymbal_pedal
drum_cymbal_soft
drum_heavy_kick
drum_roll
drum_snare_hard
drum_snare_soft
drum_splash_hard
drum_splash_soft
drum_tom_hi_hard
drum_tom_hi_soft
drum_tom_lo_hard
drum_tom_lo_soft
drum_tom_mid_hard
drum_tom_mid_soft

DRUMLOOPS

loop_amen_full
loop_amen
loop_breakbeat
loop_compus
loop_garzul
loop_industrial
loop_mika
loop_safari
loop_tabla
vinyl_scratch

DRUMS Electronic

elec_beep
elec_bell
elec_blip
elec_blip2
elec_blup
elec_bong
elec_chime
elec_cymbal
elec_filt_snare
elec_flip
elec_fuzz_tom
elec_hi_snare
elec_hollow_kick
elec_lo_snare
elec_mid_snare
elec_ping
elec_plip
elec_pop
elec_snare
elec_soft_kick
elec_tick
elec_triangle
elec_twang
elec_twip
elec_wood

PERCUSSION

perc_bell
perc_snap
perc_snap2
perc_swash
perc_till

DJ EFFECTS

vinyl_backspin
vinyl_hiss
vinyl_rewind

INDIAN TABLA

tabla_dhec
tabla_ghe1
tabla_ghe2
tabla_ghe3
tabla_ghe4
tabla_ghe5
tabla_ghe6
tabla_ghe7
tabla_ghe8
tabla_ke1
tabla_ke2
tabla_ke3
tabla_na_o
tabla_na_s
tabla_na
tabla_re
tabla_tas1
tabla_tas2
tabla_tas3
tabla_te_m
tabla_te_ne
tabla_te1
tabla_te2
tabla_tun1
tabla_tun2
tabla_tun3

GUITAR

guit_e_fifths
guit_e_slide
guit_em9
guit_harmonics

stroom.pxlmusic.be

Make a movie score or a jingle with coding…

… on a computer and Sonic Pi

Windows, Mac OS, Linux or Raspberry Pi

INSTRUCTIONS

This lesson was designed by
PXL-Music & PXL-Education
More information via

stroom.pxlmusic.be

Coding

Coding is the writing of rules - also called instructions - that the computer understands.

Together, that set of rules forms an **algorithm**.

With **CODE** you can let the computer play your music.

Sonic Pi is a free coding program, developed by Sam Aaron.

Coding is adventurous

Everyone who codes makes mistakes.

If that happens, the program does not do what you want.

Detecting and fixing those minor errors (*bugs*) is called **'debugging**’

Sonic Pi helps you find and fix the error by reporting an ***Error*** message at the bottom of the screen.

Still looking for an idea for your jingle? Feel free to use parts of this code

```
# example of a jingle
use_bpm 110
    sample :ambi_choir, release: 0.1, rate: -2
    sample :bass_dnb_f
    sample :bd_fat
    sleep 0.5
    sample :vinyl_scratch
    sample :bd_haus, amp: 2
    sample :elec_blip
    sleep 0.125
    sample :bd_haus, amp: 2
    sleep 0.125
    sample :bd_haus, amp: 2
    sleep 0.125
    sample :elec_bong
3.times do
    sleep 0.25
    sample :bass_drop_c, rate: 2, release: 0.2
end
sample :ambi_dark_woosh, rate: 2
sample :vinyl_scratch
sleep 1.5
sample :vinyl_backspin, rate: -2
```

16. Save your composition

There are two ways to save your code:

I. As a sound you can present to your friends without Sonic Pi

Rec ●

Click the REC-button

You will see the button flash.

Play the composition.

In the end, press the REC button again.

Sonic Pi will now open a new window.

Use the name from the previous step. Store it as instructed by your teacher.

II. As text

Save ♡

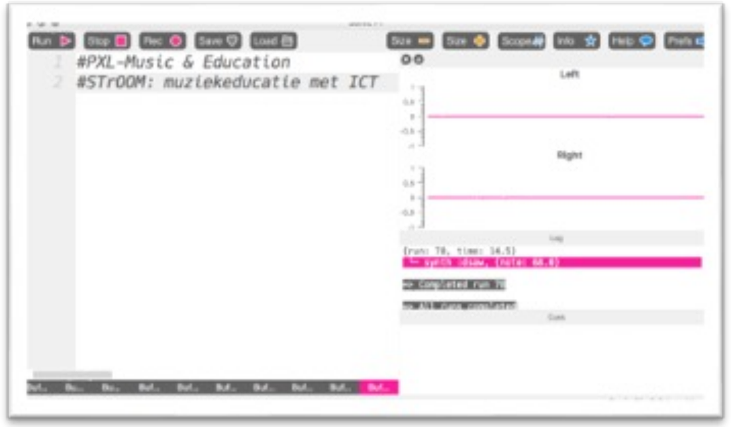
Click the SAVE button:

Sonic Pi will now open a new window.

Use the name from the previous step. Store it as instructed by your teacher.

Cheat sheet

On the left side of the screen, you can see the text you are typing: this is your CODE



```
#PXL-Music & Education
#STrOOM: muziekeducatie met ICT
```

Run ▷

When you want to play your code:
Click the RUN button:

On a Windows computer, type alt + r
On a Mac, type cmd + r

Stop ■

To stop your code:
Click the STOP button:

On a Windows computer, type alt + s
Op een Mac type cmd + s

Scope

To *'see'* your sound,
click the SCOPE button:

1. Program your 1st code

Type:

```
play 64
```

Press Alt + R or click the RUN button:

You should hear a tone.
Musicians call this pitch a "mi" or an "E."

If you don't hear any sound, ask your teacher for advice.

Now, try a lower or higher number than 64.

What do you hear?

15. Give your code and your composition a name

Be proud of what you create.

Coders also give a name to their code.
Just like the music composers.

In Sonic Pi, you can write any text provided you use the # *sign*

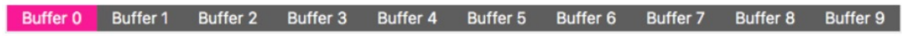
Therefore, come up with a title for your code and fill in your name.

For example, type

```
# Beats by Hamed and Illa
```

14. Step by step

1. In Sonic Pi, choose an empty ‘*buffer*’ for your new composition *(at the bottom of the screen)*



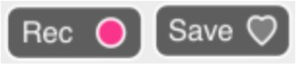
2. Start with giving your code a name with the #-sign. Choose ambient or drum sounds and create a composition with the live_loop instruction.

3. Create multiple loops with a different sound

4. Adjust the tempo with the use_bpm instruction

5. Listen to your result. What might you use this for? As a jingle? As a film score?

6. Are you satisfied? Save your code to the desktop with the REC and SAVE buttons. Remember the title and your name!



2. A single tone is a bit boring!

So, type:

```
play 64
play 66
play 68
```

Press Alt + R or click the RUN button:

You can hear the three notes play together.
This is a chord: several notes (or notes) play simultaneously.

Try some other numbers.

Which combination do you like?

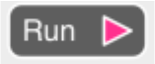
3. Creating a melody

You can turn a chord into a melody by adding short pauses or 'rests' between each note.

The instruction for this is **SLEEP**, followed by a value between 0.1 and 1

For example, type:

```
play 64
sleep 1
play 66
sleep 0.5
play 68
```

Press Alt + R or click the RUN button: 

*Now, you hear a **melody**: a sequence of notes.*

Experiment with other notes and shorter or longer rests. Your melody may also consist of several notes.

Try to have a simple melody for your team in 5 minutes.

13. What do you want to create?

In a few moments, you will present your result.

But for this assignment, you first have to imagine what you want to make:

- A jingle for the radio or a DJ?
- Music for advertising?
- Music for an exciting movie scene?
- Ask yourself the following questions:
- Should my music be short or long?
- Should the tempo be fast or slow?
- Should I work with atmospheric sounds or rather with melodies? Or both?

Tips:

- Be sure to check the list to choose a sample (there
- are cool DJ effects)
- Don’t make your idea to complicated

12. Use more octaves

If you want to make your arpeggio more exciting, you can use the **num_octaves** instruction.

This allows you to play multiple octaves.

Try the following code.

```
live_loop :arpeggio do
play_pattern_timed chord(:E2, :minor, num_octaves: 2), [0.25]
end
```

4. Repeating a melody

You can repeat your melody a number of times with the *'.times do'* instruction.

Note: for each do-instruction, you must end with an **end** instruction. We call this a 'block'.

For example, type:

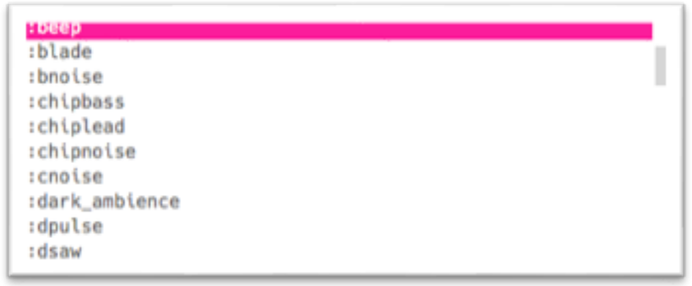
```
3.times do
    play 64
    sleep 1
    play 66
    sleep 0.5
    play 68
    sleep 0.25
end
```

Press Alt + R or click the RUN button:

Experiment with (more or less) repetitions, different notes, and shorter or longer rests.

5. Choosing a different sound

Typing **'use_synth'** will bring up a list of possible sounds:



```
:beep
:blade
:bnoise
:chipbass
:chiplead
:chipnoise
:cnoise
:dark_ambience
:dpulse
:dsaw
```

Try to choose a sound that fits your melody.

For example, type:

```
3.times do
    use_synth :dsaw
    play 64
    sleep 1
    play 66
    sleep 0.5
    play 68
    sleep 0.25
end
```

Press Alt + R or click the RUN button: Run ▶

11. Arpeggios

An arpeggio is a rhythmic repetition of notes. Arpeggios are very popular in film, dance, and pop music.

First, choose a chord with

A,B,C,D,E or F

Now think about what emotion you want.

- Major is happy (:major)
- Minor is wistful (:minor)

But you will see that there are many more.

In Sonic Pi, you use **the play_pattern_timed chord** instruction.

In this example we use an E chord:

```
live_loop :arpeggio do
play_pattern_timed chord(:E3, :m7), [0.25]
end
```

10. Add a second loop

Add a second simple loop with a different sample.

You can use a second ambient sample but also a bass drum.

Eg.

```
use_bpm 70
live_loop :rhythm do
  sample :ambi_soft_buzz, amp: 1.5, rate: 0.2
  sleep 3

live_loop :kick do
sample :bd_808, rate: 1
sleep 2
end
```

6. Working with an ‘ambient sample’

Choose an empty 'buffer' at the bottom of your screen.

Choose one of the ambient samples on the cheat sheet *(at the back of this booklet)*

For example, type: **sample :ambi_soft_buzz**

Press Alt + R or click the RUN button: Run ▷

For a quieter sound *(amp:)* that seems to come more from the left *(pan:),* type:

```
sample :ambi_soft_buzz, amp: 0.5, pan: -1
```

For a louder sound *(amp:)* coming more from the right *(pan:)*, type:

```
sample :ambi_soft_buzz, amp: 1.5, pan: 1
```

7. Change the ‘speed’ of your sample

You can increase or decrease the playback speed of your sample by using the *rate* function.

- rate: 1 for normal speed
- rate: 0.5 for half speed
- rate: 2 for double-speed

Experiment with a different rate.

Try to give the sample a different feel.

```
sample :ambi_soft_buzz, amp: 1.5, rate: 0.5
```

8. Change the ‘direction’

Play the sample “*in the other direction*.”

This way, you will first hear the end of the sound and then the beginning.

Give the rate a negative value:

For example, type:

```
sample :ambi_soft_buzz, amp: 1.5, rate: -0.5
```

9. Create a rhythm with your sample

Choose the *rate:* which feels best for your sample.

Repeat your sample with the ***'live_loop'*** instruction, followed by a name of your choice, and *do*.

End your code with ***end***

eg.

```
live_loop :rhythm do
  sample :ambi_soft_buzz, amp: 1.5, rate: 0.2
  sleep 3
end
```

The music will keep on repeating until you stop with the **STOP** button or press **ALT+S**

Tip: You can adjust the tempo with the ***use_bpm*** function

```
use_bpm 70
live_loop :rhythm do
  sample :ambi_soft_buzz, amp: 1.5, rate: 0.2
  sleep 3
end
```