SEAN MCMANUS

Sean McManus is the author of *Raspberry Pi For Dummies* (with Mike Cook), and *Scratch Programming in Easy Steps*.
sean.co.uk
twitter.com/musicandwords

Notes go green when you play them, and you hear a recording from Sonic Pi

Get a note wrong and you hear a thud, and the note goes red. Keep trying!

As you progress, more sharps and flats are introduced to the game

CLEF HERO: CREATE A MUSIC GAME USING THE PIANO HAT

You'll Need

- Piano HAT **magpi.cc/1OALwNT**
- Some speakers on your Pi
- Pygame Zero **magpi.cc/1XdhRji**

Put a piano on your Pi, and learn to tinkle the ivories. This game teaches you to read music and program the Piano HAT.

Pimoroni's Piano HAT provides a musical keyboard for your Pi, with LEDs for illuminating the keys. In Clef Hero, you're challenged to play a pattern of notes shown on the stave. It starts easy, but gets harder as more notes, sharps, and flats are introduced. As you move up the stave you'll reuse keys for the higher octave so, for example, the 'D' key is the right answer for either D on the stave. Standard sheet music wouldn't normally include the mishmash of sharps and flats you can get on the higher levels, but that makes Clef Hero a challenging puzzle even for those with some experience.

>STEP-01

Make some sounds

First, we'll make a single audio file that contains all the notes we'll need. The Listing 1 code for Sonic Pi (on page 54) will play the notes in order. Enter the listing in one of the Sonic Pi buffer spaces. Press the Rec button to start recording, press Run to play the notes, and then press Rec again to save your recording. You can customise the sounds and use different synths, but don't make the sounds too long or the game will become unplayable.

>STEP-02

Split the note files

To split your sound recording into individual files for each note, use Audacity. Install it by entering the command `sudo apt-get update && sudo apt-get install audacity` in a terminal. Open your audio file – the default option to make a copy to edit is fine, if you're asked. From the Analyze menu, choose Silence Finder. Set the minimum duration to 0.10 and the label placement to 0.05, then click OK. Go to the Edit menu and click Preferences. In the Import/Export options, untick 'Show Metadata Editor'. From the File menu, choose Export Multiple. Use the WAV export format, choose 'Numbering after File name prefix', and enter the File name prefix of 'note'. Create a directory called **clef**, and a directory called **sounds** inside that. Choose the **sounds** directory as your export location and click Export.

Language
>PYTHON

clef.py

```
# Clef Hero by Sean McManus
import pianohat, random, time
WIDTH, HEIGHT = 600, 440
RED = (255,0,0)
GREEN = (0,255,0)
BLUE = (0,0,255)
notes_to_play = list()
note_colours = list()
level = 1
notes_data = [
[0, sounds.note_01, 1, ""], [2, sounds.note_03, 2, ""],
[4, sounds.note_05, 3, ""], [5, sounds.note_06, 4, ""],
[7, sounds.note_08, 5, ""], [9, sounds.note_10, 6, ""],
[11, sounds.note_12, 7,""], [0, sounds.note_13, 8, ""],
[2, sounds.note_15, 9, ""], [4, sounds.note_17, 10, ""],
[5, sounds.note_18, 11, ""],[7, sounds.note_20, 12, ""],
[9, sounds.note_22, 13, ""],
[1, sounds.note_02, 1, "#"],[3, sounds.note_04, 2, "#"],
[6, sounds.note_07, 4, "#"],[8, sounds.note_09, 5, "#"],
[10, sounds.note_11, 6, "#"], [1, sounds.note_14, 8, "#"],
[3, sounds.note_16, 9, "#"], [6, sounds.note_19, 11, "#"],
[8, sounds.note_21, 12, "#"], [10, sounds.note_23, 13, "#"],
[1, sounds.note_02, 2, "b"], [3, sounds.note_04, 3, "b"],
[6, sounds.note_07, 5, "b"], [8, sounds.note_09, 6, "b"],
[10, sounds.note_11, 7, "b"], [1, sounds.note_14, 9, "b"],
[3, sounds.note_16, 10, "b"],[6, sounds.note_19, 12, "b"],
[8, sounds.note_21, 13, "b"]
]

def round_setup():
    global note_position, note_number, notes_to_play
    del notes_to_play[:]
    del note_colours[:]
    level_data = notes_data[0 : level * 4]
    for i in range(8):
        notes_to_play.append(random.choice(level_data))
        note_colours.append(BLUE)
    note_position = 0
    note_number = 0
    clock.schedule_unique(hint_on, 5)

def draw():
    screen.blit(images.clef_background,(0,0))
    screen.draw.text("Clef", (310,90), color="blue", fontsize=120)
    screen.draw.text("Clef", (315,85), color="white", fontsize=120)
    screen.draw.text("Hero", (310,180), color="blue", fontsize=120)
    screen.draw.text("Hero", (315,175), color="white", fontsize=120)
    BOX = Rect((100,290), (400,120))
    SHADOW = Rect((105,295), (400,120))
    screen.draw.filled_rect(SHADOW, (0,0,0))
    screen.draw.filled_rect(BOX, (255,255,255))
    screen.blit(images.treble_clef,(105,305))
    for y in range(5):
        screen.draw.line((110, 380 - y*16), (490, 380 - y*16), (0,0,0))
    show_notes()

def show_notes():
    for i in range(8):
        draw_note(i)

def draw_note(note_number):
    screen.draw.filled_circle((180 + note_number * 35, 404 - notes_to_play[
note_number][2]*8), 7, note_colours[note_number])
    if notes_to_play[note_number][2] == 1 or notes_to_play[note_number][2] == 13:
        screen.draw.line((170 + note_number * 35, 404 - notes_to_play[
note_number][2]*8), (190 + note_number * 35, 404 - notes_to_play[note_number]
```

>STEP-03

Fix your hyphens

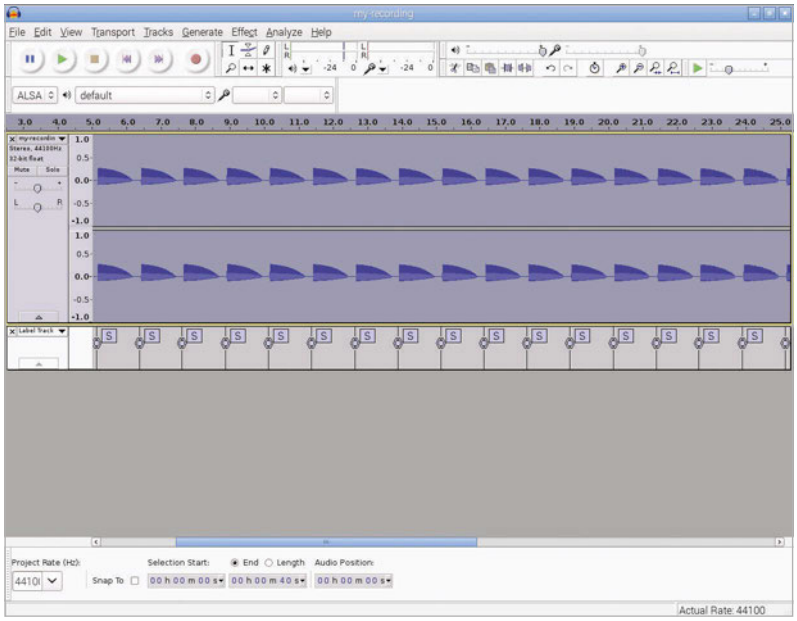
Audacity exports your files with names like note-01.wav, note-02.wav, and so on, but Pygame Zero requires underscores, not hyphens. To do the bulk rename, we recommend you install mmv with **`sudo apt-get install mmv`**. Then **`cd`** to the **sounds** directory where you have the files Audacity exported, and enter the command **`mmv "note-*.wav" "note_#1.wav"`**. Copy **thud.wav** into **sounds** too, from **pi/Pimoroni/pianohat/sounds/drums**. It's used when the player presses the wrong key.

>STEP-04

Prepare your art

You'll need a treble clef and a background image. We're using a treble clef from **magpi.cc/1XdhNA3** (download the small version) and a background by Gerd Altmann (**magpi.cc/1XdhOnK**) – again, use the small one. Rename your clef to **treble_clef.png** and your background to **clef_background.jpg**.

Below Use Audacity to split your Sonic Pi recording into notes



```
[2]*8), note_colours[note_number])
   screen.draw.text(notes_to_play[note_number][3], (
162 + note_number * 35, 395 - notes_to_play[note_number][2]*8),
color = note_colours[note_number], fontsize=24)

def update():
   draw_note(note_position)

def handle_note(piano_key, pressed):
   global note_position, note_colours, level
   if pressed == False: # key was released, not pressed
      return
   if piano_key == 12: # if top C pressed
      piano_key = 0   # treat it the same as bottom C
   clock.unschedule(hint_on)
   if piano_key == notes_to_play[note_position][0]:
      note_colours[note_position] = GREEN
      notes_to_play[note_position][1].play()
      lights_out()
      if note_position < 7:
         note_position += 1
      else:
         lights_on()
         if level < 8:
            level += 1
            round_setup()
      clock.schedule_unique(hint_on, 5)
   else:
      note_colours[note_position] = RED
```

Listing 1

Language >SONIC PI

```
note = 60
with_synth :tb303 do
  22.times do
    play note
    note = note + 1
    sleep 1.25
  end
end
```

```
      sounds.thud.play()

def hint_on():
   pianohat.set_led(notes_to_play[note_position][0], True)

def lights_out():
   for light in range(16):
      pianohat.set_led(light, False)

def lights_on():
   for light in range(13):
      pianohat.set_led(light, True)
   clock.schedule_unique(lights_out, 1)

lights_on()
round_setup()
pianohat.auto_leds(False)
pianohat.on_note(handle_note)
```

Resize your clef to 62×110 pixels. You can use **`sudo apt-get install imagemagick`** and then **`convert treble_clef.png -resize 62x100 treble_clef.png`** to resize the image from the command line. Create a directory called **images** inside the **clef** directory and put your pictures there. This is where Pygame Zero looks for all its images.

Below Use Sonic Pi to make (and customise) the sounds for this game

>STEP-05

Build the Clef Hero game

The main code listing shown contains the Python code for the Clef Hero game. Call it **clef.py** and put it into your **clef** directory, so it sits immediately above the **sounds** and **images** directories, as Pygame Zero will expect. You run it with **`sudo pgzrun clef.py`** from LXTerminal in the desktop environment. Each level has eight notes. When you play a note correctly, it goes green. When you complete the level, another eight notes are chosen randomly. The range of notes starts small, but increases with each screen you finish until all notes are in play. Tap the black notes carefully: it's easy to also hit a white key by mistake.

>STEP-06

It's time for your solo!

There's lots you can do to customise Clef Hero. The list **notes_data** describes the notes – the data is Piano HAT key, sound file, the stave line or space numbered from C=1 at the bottom, and the sharp or flat symbol. To have notes arrive in a different order, change their place in this list. To play with all the notes from the start, add **`random.shuffle(notes_data)`** immediately after **notes_data** is defined. Why not add a score or a time limit? Or adapt the game for the bass clef? Jam with it!