Manuel technique de Speed Reflex

Par Lucas Técher, Martin Tabaka et Mathis Maillot

Mai 2024

I Introduction

Ce manuel technique décrit en détail le fonctionnement de Speed Reflex, un jeu de rythme utilisant deux cartes microbit. Nous examinerons le code source, les détails techniques de la communication radio, la gestion du son et la prise en compte des actions des joueurs.

II Émetteur : `emetor.py`

II.i Imports et configuration initiale

```python
from microbit import *
import random
import radio
import music
```

— `microbit` : Bibliothèque pour contrôler les fonctionnalités du microbit.

— `random` : Pour générer des séquences d'actions aléatoires.

— `radio` : Pour la communication entre les deux microbits.

— `music` : Pour gérer les sons joués par les microbits.

II.ii Fonctions principales

— `generate_random_sequence(length)`
Génère une séquence d'actions aléatoires sans répétition consécutive.

```python
    def generate_random_sequence(length):
        instructions = ["up", "down", "left", "right", "0", "1", "2", "A", "B"]
        sequence = []
        prev_action = ""

        for _ in range(length):
            action = random.choice(instructions)
            while prev_action == action:
                action = random.choice(instructions)

            sequence.append(action)
            prev_action = action
        return sequence
```

— `display_action(action)`
Affiche l'action actuelle sur l'écran du microbit.

```
    def display_action(action):
        hash_table = {
            "up": Image.ARROW_S,
            "down": Image.ARROW_N,
            "left": Image.ARROW_W,
            "right": Image.ARROW_E,
        }
        if action not in hash_table.keys():
            display.show(action)
        else:
            display.show(hash_table[action])
```

— `game_over()`
Affiche et joue une animation et un son de défaite.

```
    def game_over():
        music.play(music.POWER_DOWN, wait=False)
        display.scroll("GAME OVER")
        ending = []
        for k in range(6, 0, -1):
            ending.append(Image.ANGRY.shift_up(k))
        for k in range(0, 6):
            ending.append(Image.ANGRY.shift_down(k))

        display.show(ending, delay=100, loop=True)
```

— `win_game()`
Affiche et joue une animation et son de victoire.

```
    def win_game():
        music.play(music.RINGTONE, wait=False)
        radio.send("win")
        display.scroll("WIN")
        ending = []
        for k in range(6, 0, -1):
            ending.append(Image.HAPPY.shift_up(k))
        for k in range(0, 6):
            ending.append(Image.HAPPY.shift_down(k))

        display.show(ending, delay=100, loop=True)
```

II.iii Boucle principale

Initialise le jeu et gère la séquence des actions.

```
startup = False
music.play(music.PRELUDE, wait=False, loop=True)
```

```

while not startup:
    for clock in Image.ALL_CLOCKS:
        if not pin_logo.is_touched():
            display.show(clock)
            sleep(100)
        else:
            startup = True

radio.send("startup")
music.play(music.POWER_UP)

flash_animation = [Image().invert() * (i/9) for i in range(9, -1, -1)]
display.show(flash_animation, delay=100)
sequence = generate_random_sequence(15)

incoming = radio.receive()
while not incoming == "ready":
    incoming = radio.receive()

music.play(music.NYAN, wait=False, loop=True)
for action in sequence:
    display_action(action)
    radio.send(action)
    while True:
        status_receive = radio.receive()
        if status_receive == "next":
            break
        elif status_receive == "loose":
            game_over()

win_game()
```

III Récepteur : `receptor.py`

III.i Imports et configuration initiale

```
from microbit import *
import radio
import music
```

III.ii Fonctions principales

— `game_over()`
Gère l'animation et le son de défaite.

```
    def game_over():
        music.stop()
        radio.send("loose")
```

```python
        display.show(Image.SAD)
        sleep(500)
        music.play(music.POWER_DOWN, wait=False, pin=None)
```

— `win_game()`
  Gère l'animation et le son de victoire.

```python
    def win_game():
        music.stop()
        display.show(Image.HEART)
        sleep(500)
        music.play(music.RINGTONE, wait=False, pin=None)
```

III.iii Boucle principale

Gère la réception des actions et l'interaction du joueur.

```python
incoming = radio.receive()
while not incoming == "startup":
    incoming = radio.receive()
    display.show(Image.ASLEEP)

music.play(music.POWER_UP, pin=None)
flash_animation = [Image().invert() * (i/9) for i in range(9, -1, -1)]
display.show(flash_animation, delay=100)
radio.send("ready")

timer_duration = 3 * 1000
running = True
music.play(music.NYAN, wait=False, loop=True, pin=None)
while running:
    display.clear()
    incoming = radio.receive()

    if incoming == "win":
        win_game()
    elif incoming != None:
        action_done = False
        timer_start = running_time()
        while not action_done:
            spent_time = running_time() - timer_start
            left_time = timer_duration - spent_time

            display.show(int(left_time / 1000) + 1, wait=False)
            if left_time <= 0:
                game_over()
                running = False
                break

            if accelerometer.is_gesture("up") and incoming == "up":
                action_done = True
            elif accelerometer.is_gesture("down") and incoming == "down":
                action_done = True
            elif accelerometer.is_gesture("left") and incoming == "left":
                action_done = True
            elif accelerometer.is_gesture("right") and incoming == "right":
                action_done = True
            elif button_a.is_pressed() and incoming == "A":
                action_done = True
            elif button_b.is_pressed() and incoming == "B":
                action_done = True
            elif pin0.is_touched() and incoming == "0":
                action_done = True
            elif pin1.is_touched() and incoming == "1":
                action_done = True
            elif pin2.is_touched() and incoming == "2":
                action_done = True

        if action_done:
            display.show(Image.HAPPY)

        sleep(500)
        radio.send("next")
```

IV Détails techniques

1. Communication radio :
   — La communication radui entre les microbits utilise le module `radio`.
   — L'émetteur envoie des actions à effectuer et le récepteur envoie des statuts (”next” ou ”loose”).
2. Gestion du son :
   — Les sons sont gérés par le module `music`.
   — Différents sons sont utilisés pour les événements (démarrage, victoire, défaite).
3. Prise en compte des actions :
   — Le récepteur détecte les actions à l'aide de capteurs intégrés : accéléromètre pour les gestes, boutons A et B, et pins tactiles.
4. Séquence et timing :
   — Une séquence aléatoire d'actions est générée pour chaque partie par la fonction `generate_random_sequence(length)` et grâce au module `random`.
   — Un minuteur est utilisé pour imposer une limite de temps pour chaque action. La fonction `running_time()` est utilisée pour suivre le temps écoulé.

V Code Source

V.i Émetteur : `emetor.py`

```
from microbit import *
import random
import radio
```

```python
import music

def generate_random_sequence(length):
   instructions = ["up", "down", "left", "right", "0", "1", "2", "A", "B"]
   sequence = []
   prev_action = ""

   for _ in range(length):
       action = random.choice(instructions)
       while prev_action == action:
           action = random.choice(instructions)

       sequence.append(action)
       prev_action = action

   return sequence

def display_action(action):
   hash_table = {
       "up": Image.ARROW_S,
       "down": Image.ARROW_N,
       "left": Image.ARROW_W,
       "right": Image.ARROW_E,
   }

   if action not in hash_table.keys():
       display.show(action)
   else:
       display.show(hash_table[action])

def game_over():
   music.play(music.POWER_DOWN, wait=False)
   display.scroll("GAME OVER")
   ending = []
   for k in range(6, 0, -1):
       ending.append(Image.ANGRY.shift_up(k))
   for k in range(0, 6):
       ending.append(Image.ANGRY.shift_down(k))

   display.show(ending, delay=100, loop=True)

def win_game():
   music.play(music.RINGTONE, wait=False)
   radio.send("win")
   display.scroll("WIN")
   ending = []
   for k in range(6, 0, -1):
       ending.append(Image.HAPPY.shift_up(k))
   for k in range(0, 6):
       ending.append(Image.HAPPY.shift_down(k))

```

```python
    display.show(ending, delay=100, loop=True)

startup = False
music.play(music.PRELUDE, wait=False, loop=True)

while not startup:
    for clock in Image.ALL_CLOCKS:
        if not pin_logo.is_touched():
            display.show(clock)
            sleep(100)
        else:
            startup = True

radio.send("startup")
music.play(music.POWER_UP)

flash_animation = [Image().invert() * (i/9) for i in range(9, -1, -1)]
display.show(flash_animation, delay=100)
sequence = generate_random_sequence(15)

incoming = radio.receive()
while not incoming == "ready":
    incoming = radio.receive()

music.play(music.NYAN, wait=False, loop=True)
for action in sequence:
    display_action(action)
    radio.send(action)
    while True:
        status_receive = radio.receive()
        if status_receive == "next":
            break
        elif status_receive == "loose":
            game_over()

win_game()
```

V.ii Récepteur : receptor.py

```python
from microbit import *
import radio
import music

def game_over():
    music.stop()
    radio.send("loose")
    display.show(Image.SAD)
    sleep(500)
    music.play(music.POWER_DOWN, wait=False, pin=None)

```

```python
def win_game():
   music.stop()
   display.show(Image.HEART)
   sleep(500)
   music.play(music.RINGTONE, wait=False, pin=None)

incoming = radio.receive()
while not incoming == "startup":
   incoming = radio.receive()
   display.show(Image.ASLEEP)

music.play(music.POWER_UP, pin=None)
flash_animation = [Image().invert() * (i/9) for i in range(9, -1, -1)]
display.show(flash_animation, delay=100)
radio.send("ready")

timer_duration = 3 * 1000
running = True
music.play(music.NYAN, wait=False, loop=True, pin=None)
while running:
   display.clear()
   incoming = radio.receive()

   if incoming == "win":
       win_game()
   elif incoming != None:
       action_done = False
       timer_start = running_time()
       while not action_done:
           spent_time = running_time() - timer_start
           left_time = timer_duration - spent_time

           display.show(int(left_time / 1000) + 1, wait=False)
           if left_time <= 0:
               game_over()
               running = False
               break

           if accelerometer.is_gesture("up") and incoming == "up":
               action_done = True
           elif accelerometer.is_gesture("down") and incoming == "down":
               action_done = True
           elif accelerometer.is_gesture("left") and incoming == "left":
               action_done = True
           elif accelerometer.is_gesture("right") and incoming == "right":
               action_done = True
           elif button_a.is_pressed() and incoming == "A":
               action_done = True
           elif button_b.is_pressed() and incoming == "B":
               action_done = True
           elif pin0.is_touched() and incoming == "0":
               action_done = True
           elif pin1.is_touched() and incoming == "1":
               action_done = True
           elif pin2.is_touched() and incoming == "2":
               action_done = True

       if action_done:
           display.show(Image.HAPPY)

       sleep(500)
       radio.send("next")
```