Dessine-moi une tortue

Antoine "entwanne" Rozo

Dessine-moi une tortue ✎ 

- Commande d'un robot sur le modèle de `turtle`
- Apprentissage de la programmation

Module `turtle`

- Commander une « tortue » virtuelle pour réaliser des dessins
- https://docs.python.org/fr/3/library/turtle.html

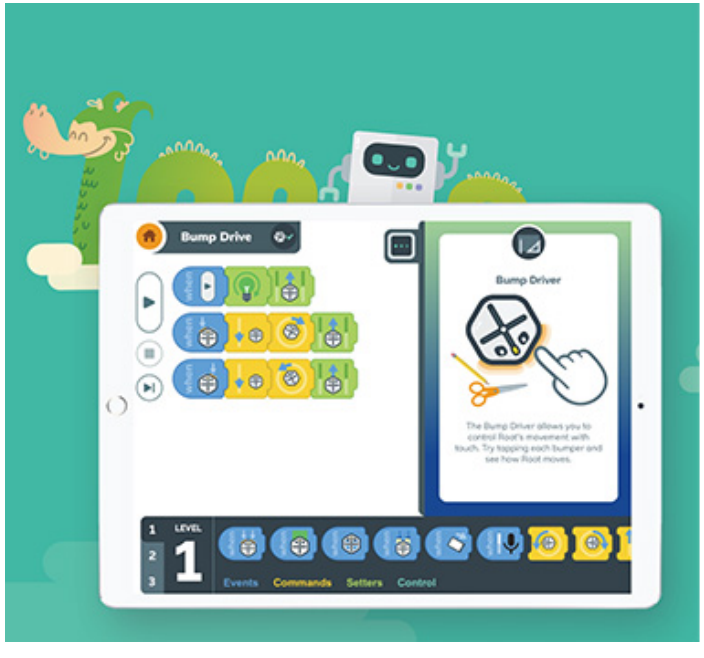
In [ ]:

```
import turtle

turtle.color('blue')
turtle.speed(1)

for i in range(4):
    turtle.forward(200)
    turtle.right(90)

turtle.done()
```

Root robot

Root robot

- Robot commercialisé par iRobot
- *Coding robot for kids*
- Programmation graphique par blocs (type Scratch)



Spécifications

- Protocol *Bluetooth Low Energy* / *GATT*
    - https://github.com/RootRobotics/root-robot-ble-protocol
- SDK disponible en Python mais un peu bugué
- Utilisation synchrone uniquement

Implémentation

- Bibliothèque asynchone pour manipuler le robot
    - https://github.com/entwanne/aiorobot
- Garder une interface la plus simple possible
- Réception et gestion des événements

Turtle dans la vraie vie

Turtle dans la vraie vie

- Diriger une tortue et dessiner
- *Helper* pour lancer le robot sur une coroutine

In [ ]:

```
from aiorobot import run_robot

async def main(robot):
    await robot.marker.down()
    for i in range(4):
        await robot.motor.drive(150)
        await robot.motor.rotate(900)
    await robot.marker.up()
    await robot.disconnect()

await run_robot(started=main)
```

Utlisation manuelle

- La lib permet d'utiliser le robot plus directement en instanciant l'objet

In [ ]:

```
from aiorobot import get_robot

async def main():
    async with get_robot() as robot:
        await robot.marker.down()
        for i in range(4):
            await robot.motor.drive(150)
            await robot.motor.rotate(900)
        await robot.marker.up()

await main()
```

Contrôle d'une voiture télécommandée

- Thread tkinter pour récupérer les touches du clavier

In [ ]:

```
from tkinter import Tk

def handle_keys(keyboard):
    left, right = 0, 0
    if 'Up' in keyboard:
        left += 100
        right += 100
    if 'Down' in keyboard:
        left -= 100
        right -= 100
    if 'Left' in keyboard:
        left -= 50
        right += 50
    if 'Right' in keyboard:
        left += 50
        right -= 50
    q.put_nowait((left, right))

def tk_thread():
    root = Tk()
    keyboard = Keyboard(root, update_func=handle_keys)
    root.bind('<KeyPress>', keyboard.key_pressed)
    root.bind('<KeyRelease>', keyboard.key_released)
    try:
        root.mainloop()
    finally:
        q.put_nowait(None)
```

Contrôle d'une voiture télécommandée

- Thread principal pour gérer les événements du robot
- Utilisation d'une queue pour gérer les interactions
- Les vitesses des moteurs des deux roues sont indépendantes

In [ ]:

```
from aiorobot import run_robot
from aiorobot.examples.thread import queue as q

async def start(robot):
    while True:
        event = await q.get()
        if event is None:
            break
        left, right = event
        await robot.motor.set_speed(left, right)
    await robot.motor.set_speed(0, 0)
    await robot.disconnect()

loop = asyncio.get_running_loop()
await asyncio.gather(
    run_robot(started=start),
    loop.run_in_executor(None, tk_thread),
)
```

Dessin d'une rosace

- Tracé d'arcs de cercle
- Soumis à quelques imprécisions parfois

In [ ]:

```
from aiorobot import run_robot

async def flower(robot):
    for _ in range(3):
        await robot.motor.rotate(600)
        await robot.motor.drive_arc(1200, 200)
        await robot.motor.rotate(600)

async def main(robot):
    await robot.motor.drive(200)

    await robot.marker.down()
    await robot.motor.drive_arc(3600, 200)
    await flower(robot)

    await robot.motor.drive_arc(600, 200)
    await flower(robot)

    await robot.marker.up()
    await robot.motor.drive(200)
    await robot.disconnect()

await run_robot(started=main)
```

Un peu plus qu'une tortue

Actions simultanées

- Il est possible d'exécuter d'autres commandes sur le robot pendant qu'il roule
- Lui demander d'allumer des LED par exemple

In [ ]:

```
from aiorobot import run_robot

async def drive(robot):
    await robot.motor.drive_arc(1800, 200)

async def color(robot):
    colors = [
        (255, 0, 0),
        (0, 255, 0),
        (0, 0, 255),
    ]
    for i in itertools.count():
        await robot.led.on(colors[i % 3])
        await asyncio.sleep(0.5)

async def main(robot):
    asyncio.create_task(color(robot))
    await drive(robot)
    await robot.disconnect()

await run_robot(started=main)
```

1, 2, 3, 4, musique ♫

- Il sait aussi jouer des notes de musique
- On les précise à l'aide de leurs fréquences

In [ ]:

```
from aiorobot import run_robot

async def start(robot):
    notes = [131, 145, 165, 175, 196, 220, 247, 262]
    notes = [n*2 for n in notes]

    for note in itertools.cycle(notes + notes[::-1]):
        await robot.music.play(note, 500)

async def stop(robot):
    await robot.disconnect()

await run_robot(started=start, stopped=stop)
```

Événements

- Le robot interagit avec son environnement
- Il dispose de différents capteurs qui déclenchent des événements
    - Bumpers, réaction au toucher, luminosité, détection de couleurs, etc.

In [ ]:

```
from aiorobot import run_robot

async def start(robot):
    await robot.motor.set_speed(100, 100)

async def stop(robot):
    await robot.disconnect()

async def bump(robot, timestamp, bumper):
    if bumper & bumper.LEFT:
        await robot.motor.set_speed(50, 100)
        await asyncio.sleep(1)
        await robot.motor.set_speed(100, 100)
    elif bumper & bumper.RIGHT:
        await robot.motor.set_speed(100, 50)
        await asyncio.sleep(1)
        await robot.motor.set_speed(100, 100)

await run_robot(started=start, stopped=stop, bumper_event=bump)
```

Simulateur

Simulateur

- Le robot est intrinsèquement lent
- On ne peut exécuter qu'un programme à la fois
- Il est difficile de l'utiliser pour débuguer un programme
- ⇛ Écriture d'un simulateur pour simplifier le développement

In [ ]:

```
from aiorobot import run_robot
from aiorobot.fake_driver import Client

async def main(robot):
    await robot.marker.down()
    for i in range(4):
        await robot.motor.drive(150)
        await robot.motor.rotate(900)
    await robot.marker.up()
    await robot.disconnect()

await run_robot(started=main, client_cls=Client)
```

Simulateur

- Permet d'augmenter la vitesse d'exécution du robot
- Fonctions pour tracer les cordes entre des points répartis autour d'un cercle
- Révéler une figure à l'aide d'une configuration précise des cordes

In [ ]:

```
from aiorobot import run_robot
from aiorobot.fake_driver import Client, FakeRobot

FakeRobot.speed = 500
N, F = 50, 2

def get_point(i):
    angle = (i / N) * 2 * math.pi
    x, y = 400 + 250 * math.cos(angle), 300 + 250 * math.sin(angle)
    return x + y*1j

async def goto(robot, src, dst, angle_src):
    vec = dst - src
    angle_dst = cmath.log(vec / abs(vec)).imag
    await robot.motor.rotate(int(math.degrees(angle_src - angle_dst) * 10))
    await robot.motor.drive(int(abs(vec)))
    return dst, angle_dst

points = [get_point(i) for i in range(N)]
to_draw = set(range(N))
segments = []
```

Simulateur

- Ce dessin aurait mis 8 minutes sur le vrai robot

In [ ]:

```
async def main(robot):
    z = 100 + 500j
    angle = 0

    while to_draw:
        i = min(
            to_draw,
            key=lambda i: abs(points[i] - z),
        )
        j = (i*F) % len(points)
        to_draw.remove(i)

        z1 = points[i]
        z2 = points[j]

        if z1 != z:
            await robot.marker.up()
            z, angle = await goto(robot, z, z1, angle)

        if z2 != z:
            await robot.marker.down()
            z, angle = await goto(robot, z, z2, angle)

    await robot.disconnect()

await run_robot(started=main, client_cls=Client)
```

Et maintenant ?

Et maintenant ?

- Finir d'implémenter le protocole du robot
    - Détection des couleurs notamment
- Offrir une interface plus simple sur certaints modules
    - Repère et coordonnées dans le plan
    - Notes de musique
- Continuer le simulateur
    - Ne gère pour le moment que les déplacements et le tracé
    - Ajouter la gestion des événements

Questions ?