Aktor: Audiodateien (*.mp3) abspielen

Das folgende Aktorscript spielt beispielhaft die 8-Bit Chiptune Melodie von Mac Gyver ab. Folgende Setupschritte werden dafür benötigt. Das Setup basiert auf einem Raspberry OS (Debian 12 Bookworm) auf einem Raspberry Pi 3 B+.

Dieser Sound Actor könnte zum Beispiel verwendet werden, um folgende Dinge zu verrichten:

- Alarmtöne und standardisierte Meldungen in der Werkstatt abspielen / abbrechen
- per TTS (Text to Speech) parametrierte Texte in Sprache umwandeln und abspielen
- einzelne Jingles oder Playlists wiedergeben

Konzept und Installation

Grundsätzlich gibt es zwei Dateien: `play_mp3.py` und `main.py`. Dieses Aktorscript basiert auf dem Python Process Template.

`play_mp3.py` macht nichts, außer eine mp3 abzuspielen. Dazu verwenden wir die am Raspberry Pi bereitgestellte 3,5mm Audioklinke, an der wir einen gewöhnlichen Lautsprecher anstecken und konfigurieren die Soundausgabe entsprechend darauf.

`main.py` startet `play_mp3.py`, wenn in der Client App auf `USE` geklickt wird und beendet den Prozess per kill, wenn auf `GIVEBACK` geklickt wird.

```
cd /opt/fabinfra/adapters/
git clone https://gitlab.com/fabinfra/fabaccess/actors/python_process_template.git mp3play
cd /opt/fabinfra/adapters/mp3play/
python3 -m venv env
. env/bin/activate #activate venv


pip3 install pygame psutil
```

Wir müssen die Alsa-Audiokonfiguration anpassen und das Standardwiedergabegerät setzen

```
vim /etc/asound.conf
```

```
pcm.!default {
type asym
playback.pcm {
type plug
slave.pcm "hw:1,0"
}
}
```

> Falls die Wiedergabe zu leise ist, kann diese mit `alsamixer` justiert werden.

Berechtigungen anpassen

```
sudo usermod -aG audio bffh
```

Script files

```
cd /opt/fabinfra/adapters/mp3play/play_mp3.py
```

```
#!/opt/fabinfra/adapters/mp3play/env/bin/python3

import pygame

def play():
    print("In Use")
    file = '/opt/fabinfra/adapters/mp3play/8bit-macgyver.mp3'
    pygame.init()
    pygame.mixer.init()
    pygame.mixer.music.load(file)
    pygame.mixer.music.set_volume(1.0)
    pygame.mixer.music.play()

    while pygame.mixer.music.get_busy() == True:
         pass

if __name__ == "__main__":
    play()
```

```
cd /opt/fabinfra/adapters/mp3play/main.py
```

```
#!/opt/fabinfra/adapters/mp3play/env/bin/python3

import argparse
import psutil
import subprocess

def on_free(args, actor_name):
    PROCNAME = "play_mp3.py"
    for proc in psutil.process_iter():
        if proc.status() != psutil.STATUS_ZOMBIE:
            if PROCNAME in " ".join(proc.cmdline()):
                proc.kill()

def on_use(args, actor_name, user_id):
    cmd = "/opt/fabinfra/adapters/mp3play/env/bin/python3
/opt/fabinfra/adapters/mp3play/play_mp3.py"
    try:
        proc = subprocess.Popen(cmd, shell=True, stdin=subprocess.PIPE,
stdout=subprocess.PIPE, stderr=subprocess.PIPE, start_new_session=True)
    except OSError as e:
        raise OSError("{0}\nCommand failed: errno={1} {2}".format(' '.join(cmd), e.errno,
e.strerror))

def on_tocheck(args, actor_name, user_id):
    print("To Check")

def on_blocked(args, actor_name, user_id):
    print("Blocked")

def on_disabled(args, actor_name):
    print("Disabled")

def on_reserve(args, actor_name, user_id):
    print("Reversed")

def on_raw(args, actor_name, data):
    print("Raw")



def main(args):
```

```
    new_state = args.state
    if new_state == "free":
        on_free(args, args.name)
    elif new_state == "inuse":
        on_use(args, args.name, args.userid)
    elif new_state == "tocheck":
        on_tocheck(args, args.name, args.userid)
    elif new_state == "blocked":
        on_blocked(args, args.name, args.userid)
    elif new_state == "disabled":
        on_disabled(args, args.name)
    elif new_state == "reserved":
        on_reserve(args, args.name, args.userid)
    elif new_state == "raw":
        on_raw(args, args.name, args.data)
    else:
        print("Process actor called with unknown state %s" % new_state)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("name", help="name of this actor as configured in bffh.dhall")

    subparsers = parser.add_subparsers(required=True, dest="state")

    parser_free = subparsers.add_parser("free")

    parser_inuse = subparsers.add_parser("inuse")
    parser_inuse.add_argument("userid", help="The user that is now using the machine")

    parser_tocheck = subparsers.add_parser("tocheck")
    parser_tocheck.add_argument("userid", help="The user that should go check the machine")

    parser_blocked = subparsers.add_parser("blocked")
    parser_blocked.add_argument("userid", help="The user that marked the machine as blocked")

    parser_disabled = subparsers.add_parser("disabled")

    parser_reserved = subparsers.add_parser("reserved")
    parser_reserved.add_argument("userid", help="The user that reserved the machine")
```

```
    parser_raw = subparsers.add_parser("raw")
    parser_raw.add_argument("data", help="Raw data for for this actor")

    args = parser.parse_args()
    main(args)
```

```
chown -R bbfh:bffh /opt/fabinfra/adapters/mp3play/
```

Das Script manuell testen

```
# Die mp3 abspielen
/opt/fabinfra/adapters/mp3play/env/bin/python3 /opt/fabinfra/adapters/mp3play/play_mp3.py

# Die mp3 abspielen, aber per Aktor-Script
/opt/fabinfra/adapters/mp3play/env/bin/python3 /opt/fabinfra/adapters/mp3play/main.py state
inuse Admin

# Die mp3 stoppen, falls sie noch läuft
/opt/fabinfra/adapters/mp3play/env/bin/python3 /opt/fabinfra/adapters/mp3play/main.py state
free
```

bffh.dhall Snippet

```
        mp3play =
        {
           module = "Process",
            params =
            {
                cmd = "/opt/fabinfra/adapters/mp3play/env/bin/python3",
                args = "/opt/fabinfra/adapters/mp3play/main.py",
            }
        },
```

FabAccess Config Generator Snippet

```
vim /opt/fabinfra/fabaccess-config-generator/actors.ini
```

```
[mp3play]
module = Process
```

```
param_cmd = "/opt/fabinfra/adapters/mp3play/env/bin/python3"
param_args = "/opt/fabinfra/adapters/mp3play/main.py state inuse $actor_id"
```

TTS-Beispiel

Folgendes Snippet kann verwendet werden, um Text in mp3 zu verwandeln und diese dann im Anschluss abzuspielen (mit dem VLC Player Kommando `vlc`). Dafür wird gTTS (Google) verwendet:

```
from gtts import gTTS
import os
text = 'Hallo liebe Werkstattnutzer. Wir schließen die Werkstatt um 20 Uhr. Es ist jetzt 19.45 Uhr. Bitte räumt eure letzten Sachen auf. Wir wünschen euch einen guten Abend. Kommt gut nach Hause!'
language = 'de'
obj = gTTS(text=text, lang=language, slow=False, tld=language)
obj.save("Werkstattansage.mp3")
os.system("vlc Werkstattansage.mp3")
```

Version #21
Erstellt: 22 November 2024 12:08:35 von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 24 Februar 2025 02:29:59 von Mario Voigt (Stadtfabrikanten e.V.)