Pixelserver2 App Entwicklungsanleitung

Andreas

30. Dezember 2018

Inhaltsverzeichnis

1 Allgemeines

Apps für den Pixelserver sind normale Konsolenanwendungen. Sie werden vom Pixelserver gestartet und generieren dann Bilder. Diese geben sie über die Standardausgabe aus und der Pixelserver zeigt sie dann an. Dies erlaubt es Apps in quasi allen Programmiersprachen zu entwickeln. Wenn der Nutzer eine App startet wird das entsprechende Programm ausgeführt und es wird vom Pixelserver beendet, wenn der Nutzer eine andere App auswählt. Hierbei kann Apps vom Nutzer ein zusätzlicher freier Parameter übergeben werden.

1.1 Persistente Apps

Einzelne Apps können auch als persistent markiert werden. Das heißt sie laufen dann auch im Hintergrund weiter, wenn der User eine andere App auswählt. Sie können auch keine Parameter übernehmen. Diese persistenten Apps sollten nur verwendet werden, wenn es für die Funktionalität **zwingend** notwendig ist.

1.2 Konfiguration

Damit der Pixelserver Apps kennt müssen sich in der *config.py* eingetragen werden. In der Variable *Apps* befindet sich eine Liste der Apps. Jede hat das folgende Format:

```
{"guiname": "<Name fuer Nutzer>",
 "name": "<Kennung fuer API>",
 "cmd": "<Pfad zur ausfuehrbaren Datei>",
```

```
 "persistent": True/False},
```

Üblicherweise sollten die Apps im Verzeichnis *apps/* liegen.

Beispiel:

```
{"guiname": "Pong",
 "name": "pong",
 "cmd": "apps/pong.py"},
```

2 Parameter

An die ausgeführte Anwendung werden immer drei Parameter übergeben:

- **1** (Breite): Die Breite des Bildschirms in Pixeln. Im folgenden N_x
- **2** (Höhe): Die Höhe des Bildschirms in Pixeln. Im folgenden N_y
- **3** (Parameter): Ein von der App frei nutzbarer Parameter. Persistente Apps müssen für korrekte Funktionalität ignorieren.

3 Bildausgabe

Ein Bild besteht aus $N_x \cdot N_y$ Pixeln. Jeder Pixel besteht aus den Farben *Rot*, *Grün* und *Blau*. Jede Farbe wird als ein Byte

dargestellt, wobei 0 minimaler Intensität und 255 maximaler Intensität entspricht.

Für ein Bild müssen die Pixel Zeilenweise mit der Pixelreinfolge Rot, Grün, Blau auf die Standardausgabe geschrieben werden. Dieses muss in einem einzigen Schreibbefehl ausgegeben werden.

Der Index i_{Farbe} des Pixels x in der Breite und y in der Höhe ergibt sich wie folgt:

$$\begin{aligned} i_{rot} &= 3 \cdot (x + N_x \cdot y) + 0 \\ i_{gruen} &= 3 \cdot (x + N_x \cdot y) + 1 \\ i_{blau} &= 3 \cdot (x + N_x \cdot y) + 2 \end{aligned}$$

Hierbei werden $x \in \{0, 1, \ldots, N_x - 1\}$, $y \in \{0, 1, \ldots, N_y - 1\}$ und $i_{Farbe} \in \{0, 1, \ldots, N_x \cdot N_y - 1\}$ von 0 an gezählt.

Wenn eine App einige Sekunden (Standardeinstellung: 40 Sekunden) kein Bild ausgibt, wird sie vom Pixelserver als abgestürzt betrachtet und beendet.

4 setup-apps.sh

Um die Apps zu installieren/kompilieren sollte der User vor dem ersten Start des Pixelservers das Skript *setup-apps.sh* aufrufen. Somit sollte der Entwickler einer App hier seine App eintragen, damit sie korrekt gebaut wird. Für einige Sprachen sind schon Schema in der *setup-apps.sh* hierfür vorgesehen.

4.1 Skriptsprachen

Für Skriptsprachen sollten die Skripte im Allgemeinen ausführbare gemacht werden und es sollte eine Zeile der Form:

```
chmod +x apps/newapp.py
```

hinzugefügt werden.

4.2 C/C++

Apps in C und C++ liegen in *apps/c-src* und werden mit CMake kompiliert. Sie sollten zu der entsprechenden *apps/c-src/CMakeLists.txt* hinzugefügt werden.

Es werden dann im Verzeichnis *build/c* gebaut und sollten mit einer Zeile der Form

```
mv build/c/nywapp apps/
```

in der *setup-apps.sh* an den richtigen Ort kopiert werden.

5 Python App Beispiel

Hier betrachten wir eine Beispielapp in Python die zufällige Pixel nacheinander Rot, Grün oder Blau macht. Die Geschwindigkeit des Setzens der Pixel soll hier über den Parameter (in Millisekunden) eingestellt werden.

Dazu wird in *apps/example.py* folgende Datei angelegt

```
#!/usr/bin/env python3

import os
import sys
import random
import time

# Groesse des Bildschirms bestimmen
Nx = int(sys.argv[1])
Ny = int(sys.argv[2])

# Bestimme den Parameter
time_ms = 100
try:
    time_ms = int(sys.argv[3])
except:
    pass

# Puffer fuer Pixel erstellen und mit 0 initialisieren
buffer = bytearray(b"\x00" * (3 * Nx * Ny))

curPixel = 0

while True:
    # Zufaellige Pixel waeheln
    # rot
    x_r = random.randint(0, Nx-1)
    y_r = random.randint(0, Ny-1)
    i_r = 3*(x_r+Nx*y_r)
    # gruen
```

```
    x_g = random.randint(0, Nx-1)
    y_g = random.randint(0, Ny-1)
    i_g = 3*(x_g+Nx*y_g)
    # blau
    x_b = random.randint(0, Nx-1)
    y_b = random.randint(0, Ny-1)
    i_b = 3*(x_b+Nx*y_b)

    # Pixel in Puffer schreiben
    # rot
    buffer[i_r+0] = 0xff # Rotanteil
    buffer[i_r+1] = 0x00 # Gruenanteil
    buffer[i_r+2] = 0x00 # Blauanteil
    # gruen
    buffer[i_g+0] = 0x00 # Rotanteil
    buffer[i_g+1] = 0xff # Gruenanteil
    buffer[i_g+2] = 0x00 # Blauanteil
    # blau
    buffer[i_b+0] = 0x00 # Rotanteil
    buffer[i_b+1] = 0x00 # Gruenanteil
    buffer[i_b+2] = 0xff # Blauanteil



    # Zeige den Puffer an
    os.write(1, buffer)
    # warte time_ms ms
    time.sleep(time_ms*0.001)
```

In *config.py* wird folgendes zu den *Apps* hinzugefügt:

```
{"guiname": "Beispiel",
 "name": "example",
 "cmd": "apps/example.py"},
```

Zuletzt wird zur *setup-apps.sh*

```
chmod +x apps/example.py
```

hinzugefügt und die App ist einsatzbereit.