Setting up the ShugrPi
on a Raspberry Pi 4b (4GB RAM)

Introduction

The Stormwrecker Handheld Undersized Gaming Raspberry (or SHUGR) Pi is designed to run a 'custom OS' game-launcher script on startup and give the impression of a dedicated gaming console for Pygame. The required assets for this project can be found inside of the repository you got this document from.

Below is a semi-accurate explanation of how to set up the ShugrPi.

Step 1 – Flashing the Pi

The correct operating system (OS) is Raspberry Pi OS Lite 64-bit. In Raspberry Pi Imager, enable ssh, add a wpa_supplicant.conf file with the correct specifications, and disable telemetry. Our hostname is 'shugrpi' and our username will also be 'shugrpi'. Our home directory (~) is now '/home/shugrpi'.

Step 2 – Connecting to the Internet

NetworkManager seems to be a roadblock when trying to ssh into the Pi, so disable it:

```
sudo systemctl stop NetworkManager
sudo systemctl disable NetworkManager
```

Instead, we will use dhcpcd5 for networking:

```
sudo apt install dhcpcd5
```

Next, configure dhcpcd:

```
sudo nano /etc/dhcpcd.conf
```

Inside of the file, write this:

```
interface wlan0
static ip_address=192.168.1.100/24  # replace with your actual IP
```

```
static routers=192.168.1.1  # replace with your actual IP
static domain_name_servers=8.8.8.8 8.8.4.4 # replace with your actual DNS
```

Restart dhcpcd:

```
sudo systemctl restart dhcpcd
```

Now, we're going to make a startup process to connect to the internet automatically:

```
sudo nano /usr/local/bin/connect_wifi.sh
```

Write this inside of the file:

```
#!/bin/bash
# Stop any existing DHCP client
sudo dhclient -r wlan0
# Start wpa_supplicant
sudo wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant/wpa_supplicant.conf
# Obtain an IP address
sudo dhclient wlan0
```

Now we have to make a service to actually run this:

```
sudo nano /etc/systemd/system/wifi-connect.service
```

Write this inside of the file:

```
[Unit]
Description=WiFi Connection Manager
After=network.target
Wants=network.target

[Service]
Type=simple
Restart=on-failure
RestartSec=5s
ExecStart=/usr/local/bin/wifi-connect

[Install]
WantedBy=multi-user.target
```

Make this service executable:

```
sudo chmod +x /usr/local/bin/connect_wifi.sh
```

Reload all services, so that our new one gets picked up by the system:

```
sudo systemctl daemon-reload
```

Start our service:

```
sudo systemctl enable wifi-connect.service
sudo systemctl start wifi-connect.service
```

Now on reboot, the service will run and thus connect to the internet automatically.

Step 3 – Mounting a Static IP (optional)

If for some reason, you find it a good idea to have a static IP for the Pi, then just add one.

Go to this file:

```
sudo nano /etc/dhcpcd.conf
```

Inside of the file, write this:

```
interface wlan0
static ip_address=192.168.1.100/24  # replace with your actual IP
static routers=192.168.1.1  # replace with your actual IP
static domain_name_servers=8.8.8.8 8.8.4.4 # replace with your actual DNS
```

Restart dhcpcd:

```
sudo systemctl restart dhcpcd
```

In the file, you can replace all the numbers with what you want. Remember to have `/24` at the end of the first number and that your second number matches the first one all the way up to the last set. Leave that on 1.

Step 4 – Installing Python and Pygame

To run Python and Pygame on the Pi, you need to install them (obviously). You can do this in one line.

Run this:

```
sudo apt install -y python3 python3-pygame
```

That's pretty much all you need (I think). One thing to pay attention to is that using 'sudo apt install my_package' will make 'my_package' globally accessible by anywhere on your Pi. You can use variations of the above line to install other Python packages, but not all packages are available this way (e.g. the av

library doesn't install this way). Since we only have one folder using Python (mentioned in step 5), we could get away with using an actual venv folder and using pip to install our packages. To install Python and pip, we could enter in this line:

```
sudo apt install -y python3 python3-pip
```

To create a venv, get in the right directory, then run this:

```
python3 -m venv path/to/venv      # or just ./venv
```

Activate it so we can install our packages:

```
source path/to/venv/Scripts/activate    # or just ./venv
```

After activating venv, simply use pip to install Pygame and any other packages we would need:

```
pip3 install pygame
```

I am unsure if the trailing '3' is necessary when using pip or Python, but better safe than sorry. Now, you have Python and Pygame on the Pi, and you're ready to get some assets going.

Step 5 – Transferring the Files

At this point, you should be able to SSH into the Pi from your PC and wreak some havoc. Let's start by copying all of our assets over to the Pi itself. Our target directory on our Pi is this:

```
/home/shugrpi/emulator
```

If you downloaded the assets from the link in the introduction like a good human, you should have a master folder called 'shugr_pi_master'. This contains an 'emulator' directory and a 'games' directory. To transfer these files from your PC to the Pi, open a normal command line (not SSH command line) on your PC, then run the following commands:

```
scp -r "path/to/shugr_pi_master/emulator/*"
"shugrpi@shugrpi.local:/home/shugrpi/emulator"
```

and

```
scp -r "path/to/shugr_pi_master/games/*"
"shugrpi@shugrpi.local:/home/shugrpi/emulator/games"
```

Make sure your replace ‘path/to/’ with the actual path to the folder you downloaded. Notice how on the Pi, the ‘games’ folder is inside of the emulator folder instead of being a separate folder itself.

Now you have all the scripts and assets you need. You’re now ready to set up the proper environment to run all this stuff in.

Step 6 – Setting up our Environment

Now, we will set up our environment in such a way that it dedicates ALL available resources (i.e. CPU, GPU, RAM, etc.) to running our Pygame scripts. Another thing to consider is that there is no way to display Pygame windows in a terminal, and that’s why we need Openbox, a lightweight window manager.

Let’s get it:

```
sudo apt install --no-install-recommends xserver-xorg xinit openbox xterm
```

This is just the barebones libraries required for our task.

Next, let’s disable all desktop components, leaving our Pi completely ‘headless’.

```
sudo systemctl disable lightdm gdm sddm
```

We’ll now create an auto-start file to run Openbox on startup.

Let’s make this file:

```
sudo nano /etc/xdg/openbox/autostart
```

Inside the file, write this:

```
#!/bin/bash
# Disable screen blanking
xset s off
xset -dpms
xset s noblank
# Focus follows mouse (helps with window control)
xsetroot -cursor_name left_ptr &

# Launch Pygame with full GPU acceleration
export DISPLAY=:0
export SDL_VIDEODRIVER=kmsdrm
unclutter -idle 0 &
python3 /home/shugrpi/emulator/emulator.py
```

Next, we’ll create the .xinitrc file that will run the above mentioned code (similar to making a process):

```
nano ~/.xinitrc
```

Put this line inside of it:

```
exec openbox-session
```

So where we're at, we have the xinitrc file (or basically, a Openbox version of a process), but we need to create a service that will actually run it.

So where we'd normally have the startup login prompt, we would like the xinitrc file to run in place of it.

Let's disable the startup login prompt:

```
sudo systemctl disable getty@tty1
```

Now let's create our replacement service. First, make the directory that will house the service:

```
sudo mkdir -p /etc/systemd/system/getty@tty1.service.d
```

The file we will write to:

```
sudo nano /etc/systemd/system/getty@tty1.service.d/override.conf
```

Put this inside of it:

```
[Service]
ExecStart=
ExecStart=-/bin/sh -c 'exec /usr/bin/xinit /home/shugrpi/.xinitrc -- vt1'
```

Now, we have the service to replace where our login prompt would normally be.

Reboot the Pi for all the changes to take effect.

Epilogue

To summarize where we're at, our Pi automatically connects to the internet, probably has a static IP address, and now autostarts our game-launcher script via Openbox. The finished product should look for all the world like a gaming console for Pygame. This is a pretty good place to stop, but you may need to do some troubleshooting during some steps (especially steps 2 and 3). Enjoy your ShugrPi experience, and I hope all of the code actually works they the way it should. Otherwise, you're on your own. 😳