FabArcade Menu Documentation

Guo, Sijia
Gönnheimer, Lina
Güths, Katharina
Iyer, Shailesh

January 25, 2023

Contents

1 Disclaimer

Many parts of this documentation closely follow the 2013 FabArcade Guide with slight changes. This is especially the case for section 4 – Setting up the Raspberry Pi.

The 2013 FabArcade Guide can be found here.
For more information on this project, visit the Official FabArcade Website.

2 Getting Started

This project uses Python 3.9. Follow the steps below to setup and run the FabArcade menu code on your own device. Please keep in mind that some of the packages in the requirements will only work on the Raspberry Pi.

- Start VScode and bring up the command pallete. In it, search for the *Python: Create Environment* option.
- Select *Venv* as the environment type.
- Select python version 3.9.

This will setup the virtual environment in the current workspace for you. To check if everything is setup correctly, pull up the integrated terminal in VScode. `(.venv)` should show up at the start of every new line. This tells you that the virtual environment is up and running properly.

After that, run the following commands:

```
python --version
# Should return Python 3.9
python3 --version
# Should return Python 3.9
```

In order to install the required dependencies to run the project go to the root folder with the `requirements.txt` file and type in the following command in the integrated VScode terminal.

```
python -m pip install -r requirements.txt
```

This will install all the requisite packages for the project to run. You can now run `src/main.py`. This should start the FabArcade menu.

WARNING: Make sure that there is a *games* folder in your directory. For more information on folder structures, please reference the section 3.2.

2.1 Constants.py

In `src/Constants.py` you can edit the constants for the menu. To make sure it works well on the Pi set the following constants to true/false depending on the test environment.
This is the configuration for the Pi:

```
IS_FULLSCREEN = True
# Enables the GPIO button on GPIO Button 17
ENABLE_HOME_BUTTON = True
GAMES_PATH = '/home/pi/games'
```

This is the configuration we are using for testing in our local setup. Just make sure to set the games path to the proper path.

```
IS_FULLSCREEN = False
# Disables the GPIO button.
ENABLE_HOME_BUTTON = False
GAMES_PATH = 'games'
```

3 File Structures

This section describes the file structure of the project.

3.1 File Structure of the Home Folder on the Raspberry Pi

```
# Structure of the arcades home folder
/home/arcade/
            fab-arcade-menu/    # The Codebase for the menu itself
            games/              # The games created by groups will be stored here
```

Figure 1: Home Folder

The Home folder on the Raspberry Pi contains the codebase for the menu as well as a *games* folder, where the code of the games are stored. For more information on the Raspberry Pi and how to set it up, please reference section 4.

3.2 File Structure of the Game Folders

The games folder contains further subfolders where each of the games created by the groups will be stored. At the root of every game folder, there should be a `manifest.json` file which gives the menu the required metadata for the game and how to start it. For further information of the manifest file, reference section 3.3. If you want to add your own Pygames game to the FabArcade, refer to section 5.

```
/games/                 # Games Folder
    game 1/             # Games Sub Folder
        ...             # Game Files and Code
        manifest.json   # Manifest File
    game 2/
    game 3
    ...
```

Figure 2: Games Folder

3.3 Manifest File

The manifest file contains all the metadata required to run, display and launch a game successfully from the menu. It is a JSON file and should be named `manifest.json`, all in lowercase. This file needs to be present at the root folder of every game project and have the following configuration:

```
{
    /**
     * The command to run in the terminal to start the game.
     * This is important.
    */
    start: 'String' ,
    /**
     * The name of the game. This is the name displayed in the menus.
     * Please make sure this is short max 20 character
    */
    name: 'String',
    // The current version number of the game
    version: 'String',
    // Year the game was created
    year:'String',
    // Relative string path for the display image of the game in the carousel
    coverArt: 'String',
    // Relative string path for the image of the game for the grid
    image: 'String',
    // Tells if the game is multiplayer or not.
    isMultiplayer: Boolean,
    // Tells if the game is legacy. Converts joystick to keyboard input
    isLegacy: Boolean,
    // Description for the Game max(200 characters)
    description: 'String',
    // Command to install the game properly
    install: 'String',
    // Commands to run before running install
    preinstall: 'String',
    // Names of the group members
    groupMembers: [
        'String',
        'String',
        ...
    ]
}
```

Figure 3: This is the required structure for the `manifest.json` file that each game needs to provide.

WARNING: If `manifest.json` is missing, the game is assumed to not exist and will not be shown in the fab arcade menu.

4 Setting up the Raspberry Pi

We describe two methods of setting up the Pi, namely the novice method and the standard method. In the novice method, you will be able to set things up with relatively little effort and with very little knowledge. On the other hand, the standard method is more involved. You will be able to learn more about the Raspberry Pi and it also allows you a higher degree of customization.

4.1 Novice Method

In the novice method, you simply download the provided image from the FabArcade website and burn the image to an SD card via the Raspberry Pi Imager Utility. Raspberry Pi Imager Utility can be downloaded from the Raspberry Pi website and is available for all common operating systems, such as Windows, Linux and Mac OS. After downloading it, follow these simple steps:

1. Insert the SD card into an SD card reader and plug it into your laptop.
2. Start the Raspberry Pi Imager and click on the "choose OS" button.

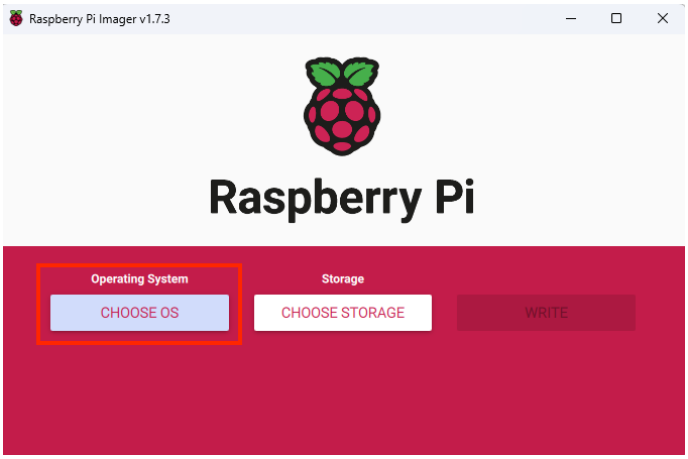

Figure 4: Select "choose OS"

3. Scroll down and select "use custom" in the list. Browse and select our image.

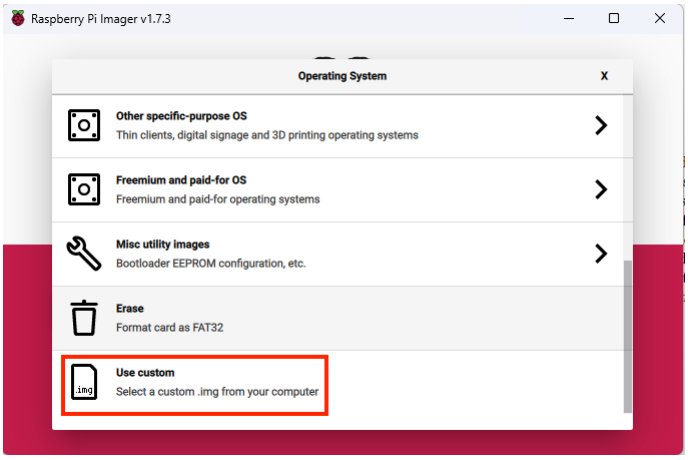

Figure 5: Select "use custom"

4. Choose your SD card as storage and press "write".

5. After it is finished writing, remove your SD card and insert it into the Raspberry Pi.

The Raspberry Pi should now boot up with the FabArcade menu.

4.2 Standard Method

In this method, you will set up and configure the Raspberry Pi from the ground up. Additional files are provided as help.

4.2.1 Installing Raspberry Pi OS

We will be using the Raspberry Pi Imager Utility.

1. Insert the SD card into the SD card slot of your laptop.
2. Press "Choose OS". Install the latest version of the Raspberry Pi OS 32-bit.

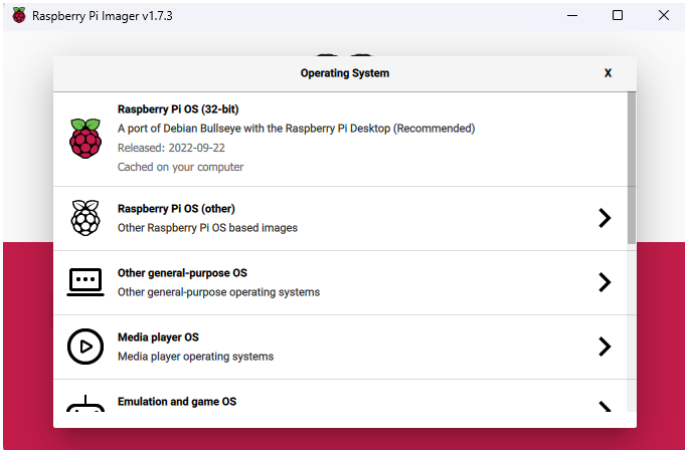

Figure 6: Select "Raspberry Pi OS 32-bit" to install it.

3. Select your SD Card.
4. Click on the small settings wheel in utility. The advanced options can be customized here.
5. Set the hostname to `fabarcade`, enable ssh, and set a username and password for the Pi. Here, we will be using `arcade` as the username and password. Make sure to set your password to something other than *"raspberry"* which is the default password for the pi.

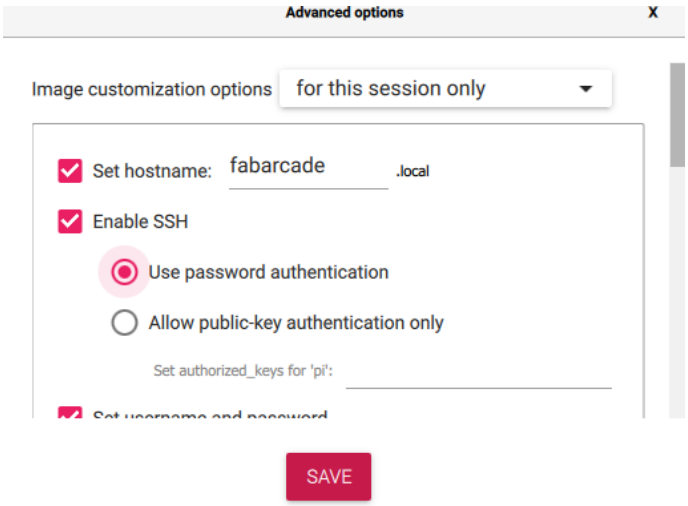

Figure 7: Settings for advanced options

6. Configure the WIFI for the Pi so that you can have access to the internet and set the Wireless LAN country. We will be going with *DE*. Remember to also set the locale settings to the correct timezone and check in the autologin for the Pi.

7. After you are done with all the configurations, write to the SD card. Click "yes" to the prompt that says all the data from the SD card will be erased.

8. Once the write is successful, you can remove the SD card from the SD card slot.

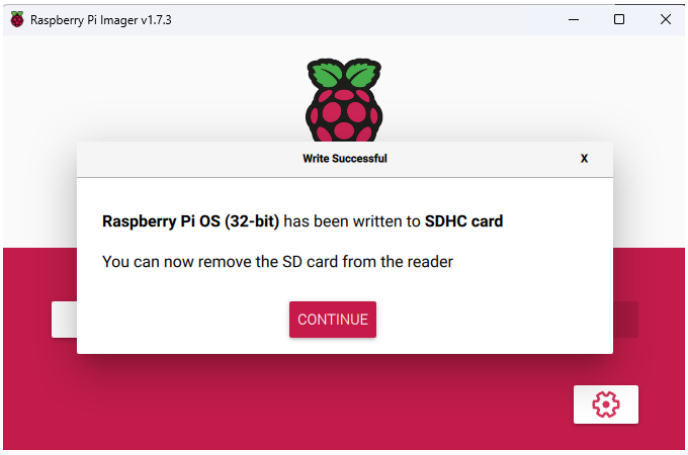

Figure 8: A pop-up window will inform you when the write was successful.

On the SD card, there should be two partitions now for the Raspberry Pi. If you are on Windows, only the boot drive is visible.

In order to make sure that the Raspberry Pi works well with the Arduino Joysticks that are being emulated, plug the SD card back into your laptop. Find the *cmdline.txt* file in the boot drive and add the following to the end of the file, all in one line:

```
usbhid.quirks=0x2341:0x8036:0x040 usbhid.jspoll=1
```

You can also configure the WIFI for the Pi and add more networks to it using `wpa_supplicant.conf`. You can add the `wpa_supplicant.conf` file to the boot drive as well, allowing it to pick up the WIFI networks on first boot.
Now, insert the SD card back into the Raspberry Pi. Connect the Pi to power, an external monitor, a mouse and a keyboard. The Pi should boot up and generate SSH keys. This first boot might take some time.

4.2.2 First Login and Setup

Don't worry if you do not have an external monitor for the Raspberry Pi. Grab an Ethernet cable and connect the Pi to your laptop. We will ssh onto it. On Linux or on the Windows Powershell you can ssh to the Pi using the command `ssh pi@[hostname/IP address]`. Since we set the hostname of the Pi earlier, we can use that to ssh into the `ssh arcade@fabarcade`. The terminal will now prompt you to enter the password. The default password for the Raspberry Pi is *"raspberry"*, however in section 4.2.1, step 5, we changed it to `arcade`. Enter the command `sudo raspi-config` to configure the Pi. Set the WIFI country to the current country you are occupying, so that the Pi connects to the WIFI. Additionally, you can also set up your keyboard under localisation options. Set `ctrl + alt + backspace` to terminate the X Server.
Next, you can set up the boot configurations. Type the command

```
sudo nano /boot/config.txt
```

In the file, uncomment the following lines:

```
hdmi_force_hotplug
framebuffer_width=1280
framebuffer_height=720
```

```
# uncomment to force a console size. By default it will be display's size minus
# overscan.
framebuffer_width=1280
framebuffer_height=720

# uncomment if hdmi display is not detected and composite is being output
hdmi_force_hotplug=1
```

Figure 9: Uncomment these lines in `config.txt`.

Also make sure that *disable overscan* is enabled and not commented. Press `ctrl-O` to write the file and press `enter` to save it.

4.2.3 Installing Software

The following software needs to be installed to make the FabArcade work as intended. Before that, update and upgrade the Pi to the latest version first using the following commands:

```
sudo apt update
sudo apt upgrade
```

To make things easier, we also recommend installing a better text editor like Vim using the command `sudo apt install vim`. Next, install the software required by pygame via

```
sudo apt install libgles2-mesa-dev \
    pkg-config libsdl2-dev libsdl2-image-dev \
    libsdl2-mixer-dev libsdl2-ttf-dev
```

To make the evdev and legacy mode work correctly we need to add the following udev rules to `/etc/udev/rules.d/50-uinput.rules`

```
KERNEL=="uinput",GROUP="udev_group", MODE="0666"
```

Then, add `arcade` to the new `udev_group` via

```
sudo groupadd udev_group
sudo usermod -a -G udev_group arcade
```

To make it work correctly on a fresh boot, create `/etc/modules-load.d/modules.conf` with the contents of `uinput`. To help test the joystick inputs, you can also install `evtest` so you are able to get the correct `evinput` key codes designated to the button presses and axis from the FabArcade. For this, use the command `sudo apt install evtest`.
Now that your Pi is up to date and equipped with the necessary tools, install the rest of the software. This procedure is analogous to step 14 from section 3.3.2 of the 2013 FabArcade Guide. Use the following commands:

- `sudo apt install xorg` (enter Y when asked) – The X server is executed from *xinit* and it is necessary for the graphical output. This will take some time to install.
- `sudo apt install openbox` – *Openbox* is a window manager and will ensure the focus of java applications. Note that it is usually installed by default on Raspbian, such that the process might be aborted.
- `sudo apt install xli` – *xli* can draw an image on the root windows of the X Server.
- `sudo apt install xbindkeys` (enter Y when asked) – We will use *xbindkeys* to kill java games.
- `sudo apt install libsdl1.2-dev` (enter Y when asked) – This lib is used by gngeo and advmame. The installation will take some time.
- `sudo apt install samba samba-common-bin` (enter Y when asked) – *Samba* will make your Pi available to receive and serve files in the local network. It is very useful if you want to transfer roms to you FabArcade.
- `sudo apt install alsaplayer-common` and `alsaplayer-text` (enter Y when asked) – *alsaplayer* is used by some java games to play back audio files. The installation will take some time.
- Optional: Install htop to see processes and how they perform, via `sudo apt install htop`

Additionally, install java for the old java based games via

```
sudo apt update
sudo apt install default-jdk
```

4.2.4 Setup the Files and User

Now copy the files that we have provided to FabArcade. Set the samba password of the user using the command `smbpasswd -a arcade`. Then, navigate to the `fab-arcade-menu/scripts` in the folders provided and link the samba file. Now, link the samba configuration file to our *smb.conf*. Again, this follows the 2013 FabArcade Guide. First remove the old *smb.conf* and type `sudo rm /etc/samba/smb.conf`. After that you should link the new file to this place by typing (one command):

```
sudo ln -s /home/arcade/fab-arcade-menu/scripts/smb.conf
/etc/samba/smb.conf
```

Next, we will set up the files required to make the menu boot up as soon as the Pi boots up from text mode. First, change the Pi's default login from *Desktop Autologin* to *Console Autologin* by going to `sudo raspi-config > System Options > Boot`. There, select *B2 Console Autologin*. Afterwards, navigate to our `rc.local` file in `scripts/rc.local` in our codebase and modify the contents of `/etc/rc.local` on the Pi to be the same as our `rc.local`. You can also replace the file present in `/etc/rc.local` with our version and replace `/etc/X11/xinit/xinitrc` with our `xinitrc`. Lastly, replace the files of `/etc/xdg/openbox/autostart` with our `autostart` and similarly replace `rc.xml` in `/etc/xdg/openbox/rc.xml` with our version of it as well. Reboot the Pi and the FabArcade menu should start up on reboot.

5 Adding your own Games

If you want to add your own Pygames game to the FabArcade, then it needs to fulfill the following requirements.

5.1 Project Structure and Management

The setup of your project has a huge impact on the ability to get reproducible builds for your game. This is important as any change in specifications, dependencies, or Python versions could lead to weird, hard to debug issues. It also makes it easier for you to debug issues and solve them since it allows you to isolate the problems to just your code and not your dependencies or anything else. Hence, it is important that you set up the project as outlined below. These are also general best practices for setting up a python project. We will talk about specifics further down but the link below provides a good baseline for setting up your project.

Structuring Your Project — The Hitchhiker's Guide to Python

Python has a feature called virtual environments, that you need for setting up isolation for your project. It allows the user to run an isolated Python instance with its own interpreter etc., making sure that other Python instances/versions are not being used to run your Python project. It also allows you to isolate the project dependencies installed using pip for your project from the global dependencies for python that are installed system wide. Furthermore, it allows the menu to run the game on the exact Python level that you specify. Our menu requires that a virtual environment is present for a python game.

How to Set Up a Virtual Environment in Python – And Why It's Useful

Use the command below to set up your virtual environment. Name your virtual environment `.venv`.

```
python -m venv .venv
```

After you have the virtual environment set up, the next important thing to do is to install your dependencies. You will notice that if you try and run your project, it will not work. It will probably fail saying a module is missing in the virtual environment. You need to install your dependencies separately in the virtual environment and you also have to freeze those dependencies in a `requirements.txt` file.

User Guide - pip documentation v22.3.1

Use the command below.

```
\~ pip freeze > requirements.txt
```

This allows the installation of all of your dependencies, using a single command.

```
\~ pip install -r requirements.txt
```

Make sure that you have a `.venv` folder for the virtual environment, and a `requirements.txt` file.

5.2 Folder Structure and Manifest File

Please reference section 3.2 and section 3.3 respectively for the required structure of your folder and `manifest.json`.

5.3 Image Requirements

The menu can display a different image in each of its views. To make things cohesive, we set the following requirements for the images. If the manifest does not include any images, a standard image will be generated so that the game can still be accessed through the menu.
The first image is referred to as `coverArt` in the manifest will be used in the carousel view:

- The cover art should be a square ($500x500$ pixels)
- The image should have the name of the game in it
- The image should be in `.png` format

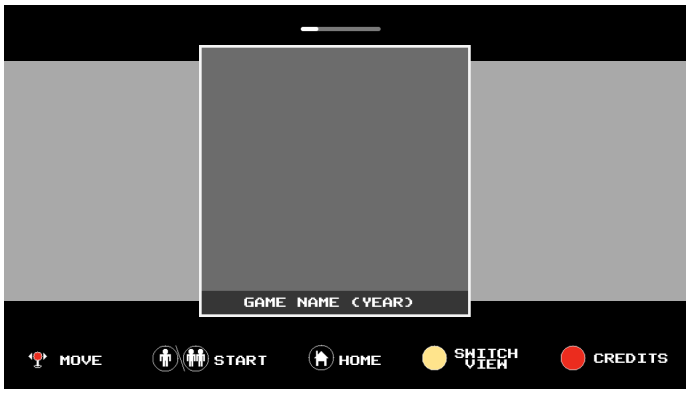

Figure 10: This is what the carousel-style menu view looks like. The gray square will contain the cover art of the game. Name and year will be adjusted accordingly.

The second image is simply referred to as `image` in the manifest. This image will be used in the grid view:

- The image should be a square ($140x140$ pixels)
- The image should be in `.png` format

Figure 11: In the grid-style menu view, the images of twelve games are shown, together with a description of the selected one on the right.

5.4 Additional Information

- The selection of whether the game is being played in single player or multi-player mode should happen in your games.
- The player should be allowed to return to the home screen directly by pressing the home button on the FabArcade.
- Make sure to check for coin inputs to start your game.
- It's useful to test your games with an XBox Controller since that is exactly how the FabArcade controls are being implemented.