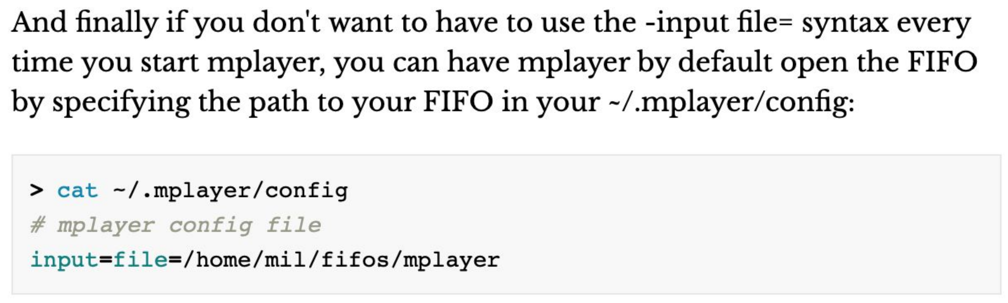
Jonathan Nusantara (jan265), Eric Hall (ewh73), Eric Kahn (edk52) HW2

1. What is the difference between a Linux pipe and a FIFO (or a named pipe)? What can you do with a FIFO (named pipe) that you cannot do with a pipe?

A pipe is used to feed the output of one command into the input of another command. A FIFO allows us to do this between processes, because it is represented as a file in the file system. It can also be named and used multiple times or by multiple processes simultaneously, while a normal pipe is only used once between two commands.

2. In Lab1, we used the guide "Using a fifo with mplayer" currently linked in the 'Guides' section of blackboard. This guide is based on a link noted at the end of the document. The link, https://milesalan.com/notes/mplayers-fifo/, also describes the use of a fifo with mplayer however it has one very confusing aspect. What is the source of the confusion that makes this guide a bit challenging to understand?


And finally if you don't want to have to use the -input file= syntax every time you start mplayer, you can have mplayer by default open the FIFO by specifying the path to your FIFO in your ~/.mplayer/config:

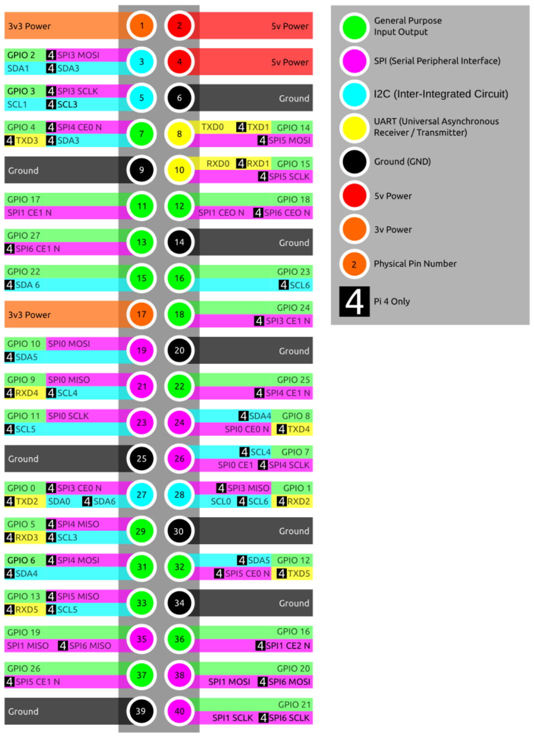
```
> cat ~/.mplayer/config
# mplayer config file
input=file=/home/mil/fifos/mplayer
```

From the above screenshot from the link, it is confusing how the path to the fifo is being specified in the config file, by stating "input=file=". It is confusing to define input in this way, defining both "input" and "file" at once.

3. List all the pins on the R-PI GPIO connector used by the piTFT screen. Explain what each pin is used for by the piTFT.

The PiTFT uses the Hardware SPI0 pins (pin 19, 21, 23, 24, 26) along with pin 18 and 22. The hardware SPI0 pins that are used (pin 19, 21, 23, 24, 26) translates to (MOSI, MISO, SCLK, CE0, CE1) respectively. MOSI is Master Output Slave Input, where Pi can send information to other devices, in this case the PiTFT. MISO is Master Input Slave Output, where Pi can receive information from other devices, in this case the PiTFT. SCLK is the serial clock pin to provide timing information to PiTFT. The CE0 and CE1 are pins where the Pi will indicate which device it is talking to, since several devices can actually share the same MOSI, MISO and SCLK pins. The pin 18 and 22 translate to GPIO 24 and 25 in BOARD numbering system, or GPIO 24 and 25 in BROADCOM, and they are used by the PiTFT for display purposes.

4. For the R-Pi 4, Model B, list all possible GPIO pins that may be used for projects and labs. Identify the maximum set (when not using any special functions). Also, list the minimum set, when all special functions (and the Adafruit 2.8 inch piTFT) are used.


3v3 Power 1 2 5v Power
GPIO 2 4 SPI3 MOSI SDA1 4 SDA3 3 4 5v Power
GPIO 3 4 SPI3 SCLK SCL1 4 SCL3 5 6 Ground
GPIO 4 4 SPI4 CE0 N 4 TXD3 4 SDA3 7 8 TXD0 4 TXD1 GPIO 14 4 SPI5 MOSI
Ground 9 10 RXD0 4 RXD1 GPIO 15 4 SPI5 SCLK
GPIO 17 SPI1 CE1 N 11 12 GPIO 18 SPI1 CEO N 4 SPI6 CEO N
GPIO 27 4 SPI6 CE1 N 13 14 Ground
GPIO 22 4 SDA 6 15 16 GPIO 23 4 SCL6
3v3 Power 17 18 GPIO 24 4 SPI3 CE1 N
GPIO 10 SPI0 MOSI 4 SDA5 19 20 Ground
GPIO 9 SPI0 MISO 4 RXD4 4 SCL4 21 22 GPIO 25 4 SPI4 CE1 N
GPIO 11 SPI0 SCLK 4 SCL5 23 24 4 SDA4 GPIO 8 SPI0 CE0 N 4 TXD4
Ground 25 26 4 SCL4 GPIO 7 SPI0 CE1 4 SPI4 SCLK
GPIO 0 4 SPI3 CE0 N 4 TXD2 SDA0 4 SDA6 27 28 4 SPI3 MISO GPIO 1 SCL0 4 SCL6 4 RXD2
GPIO 5 4 SPI4 MISO 4 RXD3 4 SCL3 29 30 Ground
GPIO 6 4 SPI4 MOSI 4 SDA4 31 32 4 SDA5 GPIO 12 4 SPI5 CE0 N 4 TXD5
GPIO 13 4 SPI5 MISO 4 RXD5 4 SCL5 33 34 Ground
GPIO 19 SPI1 MISO 4 SPI6 MISO 35 36 GPIO 16 4 SPI1 CE2 N
GPIO 26 4 SPI5 CE1 N 37 38 GPIO 20 SPI1 MOSI 4 SPI6 MOSI
Ground 39 40 GPIO 21 SPI1 SCLK 4 SPI6 SCLK
General Purpose Input Output
SPI (Serial Peripheral Interface)
I2C (Inter-Integrated Circuit)
UART (Universal Asynchronous Receiver / Transmitter)
Ground (GND)
5v Power
3v Power
2 Physical Pin Number
4 Pi 4 Only

The maximum set of GPIO pins that we can use without any special functions used are 28, which is GPIO 0-28 using Board numbering. In Broadcom numbering, they will be pin 3, 5, 7-8, 10-13, 15-16, 18-19, 21-24, 26-29, 31-33, 35-38, and 40. The minimum set, which is when all special functions are used, there will be no GPIO pins that we can use! This is because with the latest upgrade in Pi4, now every GPIO pin has special functions, and so when all are being used, we are left with none.

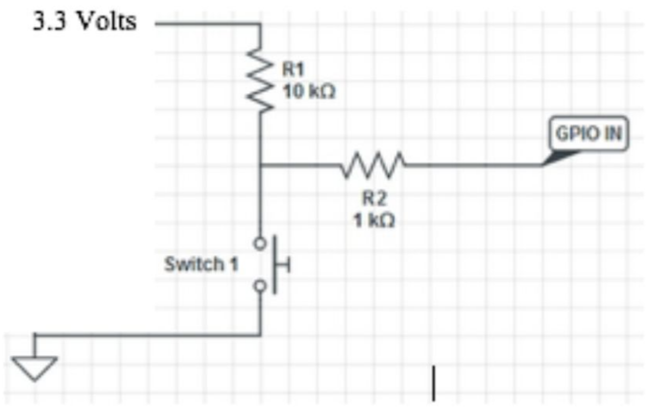
5. In Lab1, a shell script named “start_video” was created to run mplayer and video_control.py by using a single command. What is the correct ordering of operations within this script in order to this shell script to terminate correctly? Comment on the order of the calls within the script as well as which operation should run in the foreground and which should run in the background. Hint: Correct shell script operation should return to a command prompt (the $ in a command window) if everything completed in the correct order.

The correct order for start_video.sh is to start the python script, video_control.py, in the background using the “&” command at the end of the line. Then we can start the mplayer and set the input file to the FIFO so that any commands coming from the PiTFT button presses are processed by the python script, sent to the FIFO and then read by mplayer.

6. If you run the date command on the ECE5725 server, the date and time are accurate. The RPi does not have a battery-backed real-time clock so how does the RPi maintain accurate time? When would it be appropriate to add an external, battery-backed real-time clock to the RPi?

For RPi that does not have a battery, it gets the real-time clock from the internet/network, specifically getting data from an NTP server using timesyncd every time the device reboots. However, this is troublesome if you are not able to have access to a network and need to run the date command and get accurate time. So if you need the current time without having access to the internet, then we would need a battery-backed real-time clock for the RPi. This will allow the RPi to continuously track the current time accurately.

7. For the following RPi GPIO circuit, describe why R2 is necessary in the figure:



Describe a possible 'software situation' that would damage the GPIO without R2. How does R2 prevent the problem? Why is the value of 1k ohm selected for R2?

R2 is needed because if through the software the pin/GPIO is set up as an output and set to output a 1, without R2, when the button is pressed, 3.3V is now shorted to ground, resulting in a lot of current that will destroy this GPIO pin. With R2, we now have some resistance to lower the current so 3.3V isn't shorted to ground. A value of 1K is chosen because this will result in 3.3 mA which is a reasonable amount of current along the wire for the RPi to handle.

8. In Lab1 and Lab2, you use a python code video_control.py, for example, to respond to buttons connected to the RPi GPIO pins. video_control.py passes these events to an instance of mplayer, controlling the video under playback. The two processes communicate using video_fifo. mplayer and video_control.py are launched from the bash script, start_video.

   Describe what would happen if start_video is run WITHOUT first creating the fifo video_fifo. In this special case, if you press a button, what it the response of video_control.py? What is the response of mplayer? Will video_fifo be correctly created? Will control of mplayer be correct?

If we start_video without first creating the fifo video_fifo, it will not cause an error. Instead, linux will create a file called video_fifo for us, but it is NOT a fifo and is just a regular file. When we try to press the button, the video_control.py will write out the commands to the video_fifo. Then, the mplayer will take the commands and follow it, which would seem normal. However, these commands will stay in that video_fifo file. So when you next open the mplayer, it will immediately execute the commands from video_fifo file, and you will notice mplayer executing commands when you have not pressed the buttons. In this case, you will then realize you made a mistake, and to remove the existing video_fifo file and create a proper video_fifo fifo.

9. While using the PiTFT buttons to control video, you may have encountered a problem with some of the buttons. It might appear that you had several 'hits' of the buttons when you pressed the button only once. There are several causes for this issue; what is one problem that could cause this issue? (hint: there are at least three possibilities observed by teams in the lab).

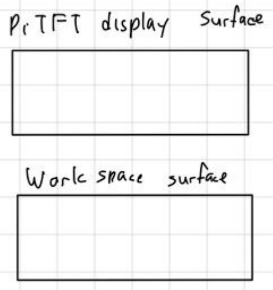
One issue is bouncing. This occurs because a person pressing the key once is interpreted as many key presses (like holding down a letter in a text editor). To solve this, we can tell the python script to sleep for a couple tenths of a second after receiving an input. A second solution is to record the time when a key press occurs only count an input as valid if a couple tenths of a second have passed between the current input and the last input.

10. Describe pygame screen elements including a surface and a rect. How are these used to animate an image? Draw a step by step diagram illustrating the process of animating 2 frames of an animation on the PiTFT.

A pygame surface is an area where we can assemble a picture. There is a surface for the screen (our PiTFT) which is what is actually displayed. There is also a workspace surface, where we can assemble a picture before displaying it to the screen. A rectangle is an object/boundary for an object that we can create and combine with a surface, so that it will eventually be displayed on the screen.

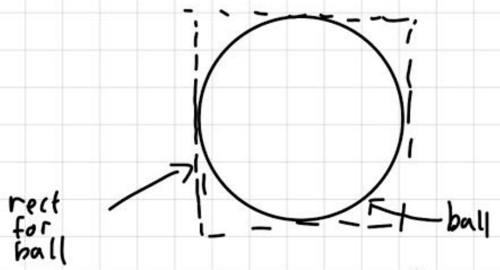
Diagram starts on the next page.

① Surface is created.
We have 2 :- PiTFT
- Workspace

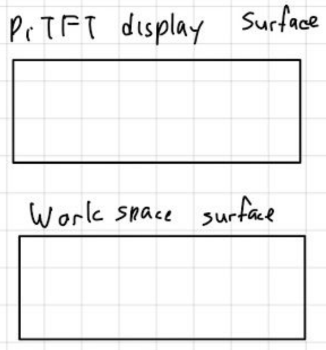


② Object is created. Ex: ball loaded from.png.

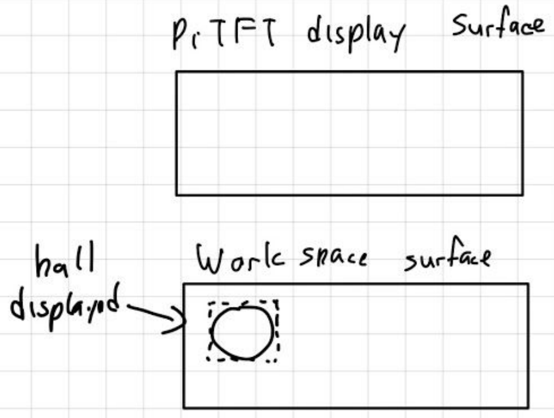
Get a rectangle for ball, which acts as boundary



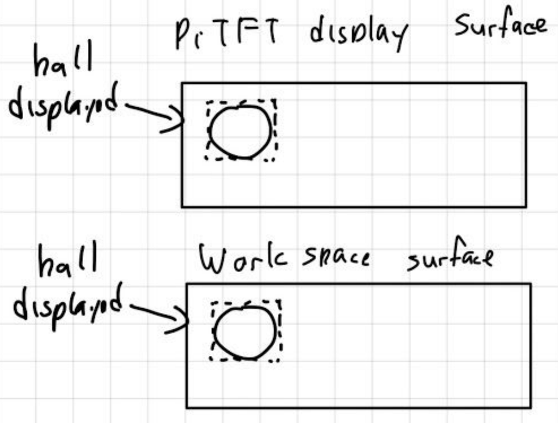
③ Move the ball to coordinate we want. Also, make PiTFT screen black.
Can't see the difference on screen as it is empty now.



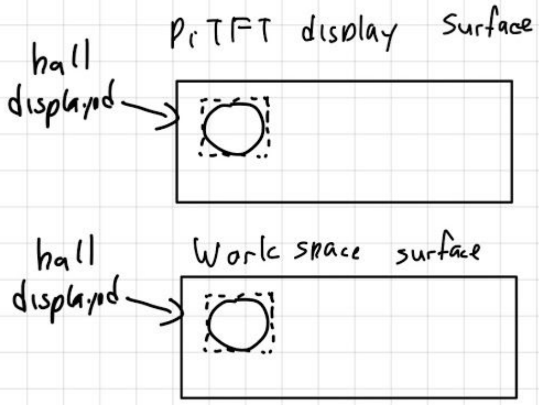
④ Using .blit(), combine ball and ball rect to workspace surface



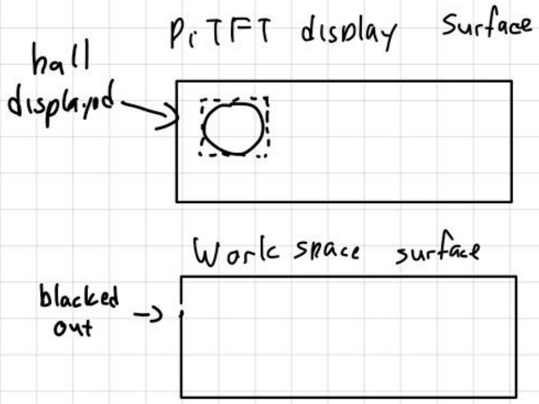
⑤ Display on PiTFT.
Using pygame.display.flip()



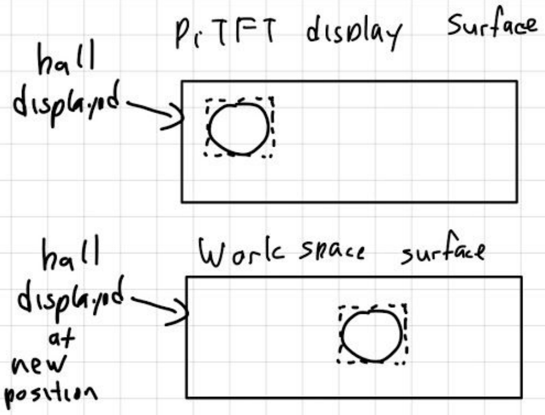
6) Iterate the infinite loop.
Move the ball.
No change on surface yet



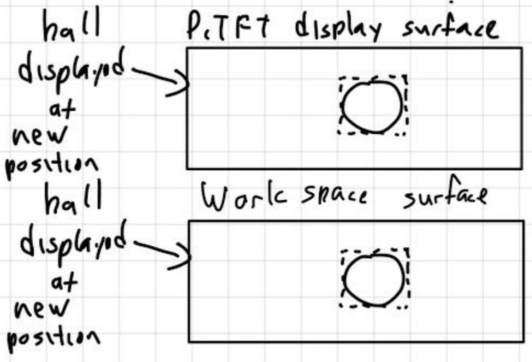
7) Screen.fill (black)
This blacks out
or erase workspace



8) screen.blit() display
ball in new position



9) Run .flip().
What's on the workspace
will be reflected
on PiTFT


PiTFT display surface
ball displayed at new position
Work space surface
ball displayed at new position