Lab2 Week1 & Week2 Report

Names: Zixiao Wang, Yue Zhang
NetID: zw579, yz2455
Date: October 20, 2019

Introduction:

We did lab2 through 4 steps:

1. Firstly we added external buttons to RPi to expand the control functions of video_control.py from lab1 including a 30s forward button and a 30s rewind button.
2. We compared the performances of looping and callback mechanism.
3. Developed some test programs using Pygame including bouncing one/two balls on TFT screen.
4. Expanded the Pygame test programs with touchscreen buttons and user interface with two-level manually.

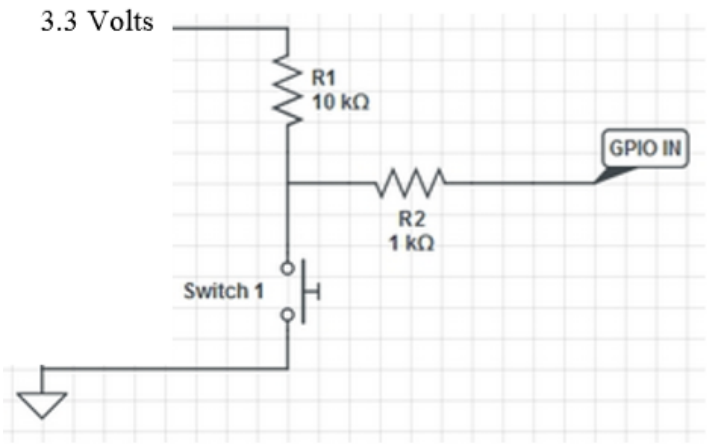
In a nutshell, we can send command through external button and touchscreen virtual buttons. Also we created a two-level user interface to control two balls bouncing on TFT screen with speed commands. All progress is backed up on another SD Card.

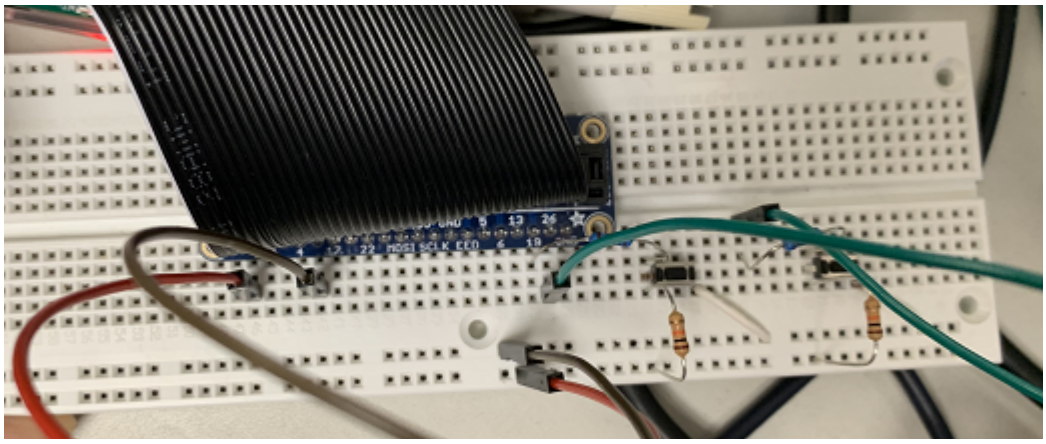
Design and Test:

Below are the steps we took and issues we came into and solved:

1. Using the breakout cable, add two additional buttons to RPi setup used for playing video on PiTFT.

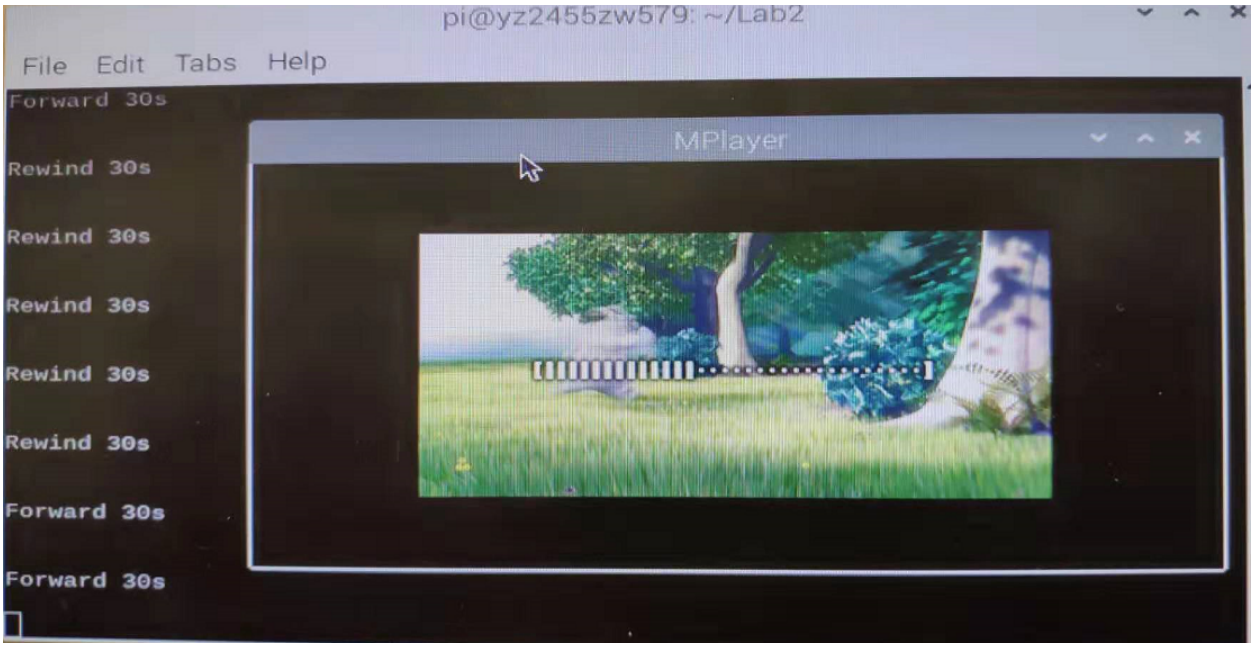
   We used GPIO19 and GPIO26 as our input pin and the as the circuit showing below:





2. Power up and create six_buttons.py to read two new buttons. And create more_video_control.py to fast forward 30s and rewind 30s. Then modify start_video to this new version.

   We modified the start_video by including new buttons and adding new functions. Now we can control video to forward and rewind 30s using external buttons.



3. Modify buttons presses to use threaded callback interrupt routines in more_vidoe_control_cb.py. Create a start_video_cb bash script.

   We modified all button press polling process to callback routine by adding event detector and callback functions.

4. Install Perf and test the performance of cal_v1.py, more_video_control.py and more_video_control_cb.py.

   Test result is shown below:
   For performance of cal_v1.py:

```
Performance counter stats for 'python /home/pi/Lab2/cal_v1.py':

        50.616563      task-clock (msec)         #    0.974 CPUs utilized
                0      context-switches          #    0.000 K/sec
                0      cpu-migrations            #    0.000 K/sec
              525      page-faults               #    0.010 M/sec
         60727973      cycles                    #    1.200 GHz
         27733318      instructions              #    0.46  insn per cycle

      0.051963337 seconds time elapsed

      0.031895000 seconds user
      0.021263000 seconds sys
```

```
 Performance counter stats for 'python /home/pi/Lab2/cal_v1.py':

         51.050937      task-clock (msec)         #    0.968 CPUs utilized
                 5      context-switches          #    0.098 K/sec
                 0      cpu-migrations            #    0.000 K/sec
               525      page-faults               #    0.010 M/sec
          61206505      cycles                    #    1.199 GHz
          27763611      instructions              #    0.45  insn per cycle

       0.052726250 seconds time elapsed

       0.042677000 seconds user
       0.010669000 seconds sys
```

```
 Performance counter stats for 'python /home/pi/Lab2/cal_v1.py':

        93.087344      task-clock (msec)         #    0.968 CPUs utilized
                7      context-switches          #    0.075 K/sec
                0      cpu-migrations            #    0.000 K/sec
              525      page-faults               #    0.006 M/sec
         55798289      cycles                    #    0.599 GHz
         27873469      instructions              #    0.50  insn per cycle

      0.096137851 seconds time elapsed

      0.064093000 seconds user
      0.032046000 seconds sys
```

```
Performance counter stats for 'python /home/pi/Lab2/cal_v1.py':

        94.870572      task-clock (msec)         #    0.974 CPUs utilized
                2      context-switches          #    0.021 K/sec
                0      cpu-migrations            #    0.000 K/sec
              526      page-faults               #    0.006 M/sec
         56896341      cycles                    #    0.600 GHz
         28218839      instructions              #    0.50  insn per cycle

      0.097378524 seconds time elapsed

      0.076400000 seconds user
      0.021828000 seconds sys
```

```
Performance counter stats for 'python /home/pi/Lab2/cal_v1.py':

         91.655365      task-clock (msec)         #    0.977 CPUs utilized
                 0      context-switches          #    0.000 K/sec
                 0      cpu-migrations            #    0.000 K/sec
               525      page-faults               #    0.006 M/sec
          54983565      cycles                    #    0.600 GHz
          27688210      instructions              #    0.50  insn per cycle

       0.093845882 seconds time elapsed

       0.076006000 seconds user
       0.019001000 seconds sys
```

For more_video_control_perf.py with different sleeping time from 0.2s to 20 milliseconds, 2 milliseconds, 200 microseconds, 20 microseconds :

```
Performance counter stats for 'python more_video_control_perf.py':

        127.249376      task-clock (msec)         #    0.025 CPUs utilized
                30      context-switches          #    0.236 K/sec
                 0      cpu-migrations            #    0.000 K/sec
               643      page-faults               #    0.005 M/sec
          81813027      cycles                    #    0.643 GHz
          39989547      instructions              #    0.49  insn per cycle

       5.135241131 seconds time elapsed

       0.110231000 seconds user
       0.020042000 seconds sys
```

```
Performance counter stats for 'python more_video_control_perf.py':

       147.041249      task-clock (msec)         #    0.029 CPUs utilized

             252      context-switches          #    0.002 M/sec

               0      cpu-migrations            #    0.000 K/sec

             643      page-faults               #    0.004 M/sec

        88311849      cycles                    #    0.601 GHz

        44066924      instructions              #    0.50  insn per cycle


      5.144698476 seconds time elapsed

      0.138999000 seconds user
      0.010692000 seconds sys


Performance counter stats for 'python more_video_control_perf.py':

       242.767015      task-clock (msec)         #    0.047 CPUs utilized

            2373      context-switches          #    0.010 M/sec

               0      cpu-migrations            #    0.000 K/sec

             644      page-faults               #    0.003 M/sec

       146798814      cycles                    #    0.605 GHz

        76603655      instructions              #    0.52  insn per cycle


      5.134305737 seconds time elapsed

      0.208029000 seconds user
      0.057785000 seconds sys
```

```
Performance counter stats for 'python more_video_control_perf.py':

        828.808883      task-clock (msec)         #    0.163 CPUs utilized

             16321      context-switches          #    0.020 M/sec

                 0      cpu-migrations            #    0.000 K/sec

               643      page-faults               #    0.776 K/sec

         540003655      cycles                    #    0.652 GHz

         278906833      instructions              #    0.52  insn per cycle


       5.085266068 seconds time elapsed

       0.262318000 seconds user
       0.746598000 seconds sys

Performance counter stats for 'python more_video_control_perf.py':

       1363.428730      task-clock (msec)         #    0.267 CPUs utilized

             48423      context-switches          #    0.036 M/sec

                 0      cpu-migrations            #    0.000 K/sec

               645      page-faults               #    0.473 K/sec

        1375880620      cycles                    #    1.009 GHz

         738611696      instructions              #    0.54  insn per cycle


       5.111093128 seconds time elapsed

       0.936674000 seconds user
       0.783747000 seconds sys
```

For more_video_control_cb_perf.py with different sleeping time from 0.2s to 20 milliseconds, 2 milliseconds, 200 microseconds, 20 microseconds :

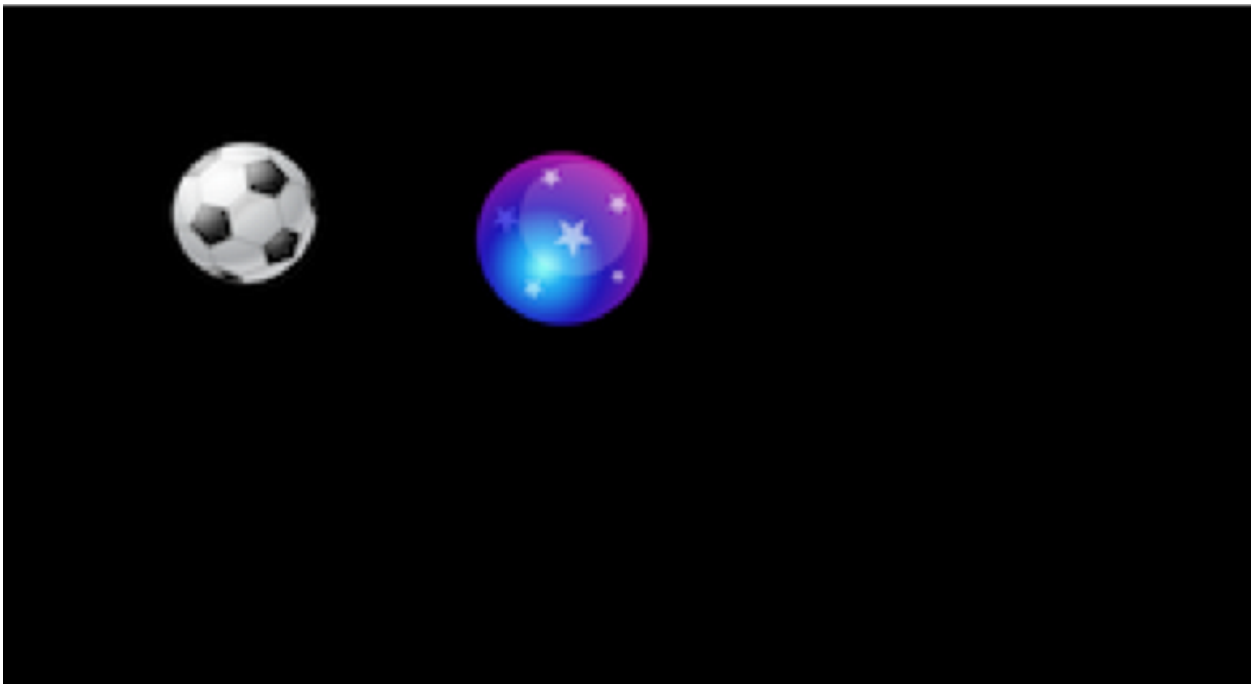
```
Performance counter stats for 'python more_video_control_cb_perf.py':

        101.447921      task-clock (msec)         #    0.020 CPUs utilized
                35      context-switches          #    0.345 K/sec
                 2      cpu-migrations            #    0.020 K/sec
               649      page-faults               #    0.006 M/sec
         101353362      cycles                    #    0.999 GHz
          42692744      instructions              #    0.42  insn per cycle

       5.117725227 seconds time elapsed

       0.062369000 seconds user
       0.041579000 seconds sys
Performance counter stats for 'python more_video_control_cb_perf.py':

        110.570678      task-clock (msec)         #    0.022 CPUs utilized
               258      context-switches          #    0.002 M/sec
                 1      cpu-migrations            #    0.009 K/sec
               648      page-faults               #    0.006 M/sec
         105990210      cycles                    #    0.959 GHz
          45400644      instructions              #    0.43  insn per cycle

       5.111069402 seconds time elapsed

       0.060831000 seconds user
       0.050692000 seconds sys
```

```
Performance counter stats for 'python more_video_control_cb_perf.py':

        178.344964      task-clock (msec)         #    0.035 CPUs utilized
              2396      context-switches          #    0.013 M/sec
                 3      cpu-migrations            #    0.017 K/sec
               648      page-faults               #    0.004 M/sec
         147939285      cycles                    #    0.830 GHz
          65076386      instructions              #    0.44  insn per cycle

       5.102572338 seconds time elapsed

       0.108056000 seconds user
       0.097250000 seconds sys
       Performance counter stats for 'python more_video_control_cb_perf.py':

              629.004012      task-clock (msec)         #    0.122 CPUs utilized
                   17362      context-switches          #    0.028 M/sec
                       7      cpu-migrations            #    0.011 K/sec
                     649      page-faults               #    0.001 M/sec
               409336550      cycles                    #    0.651 GHz
               195605658      instructions              #    0.48  insn per cycle

             5.159790497 seconds time elapsed

             0.177824000 seconds user
             0.628314000 seconds sys
```

```
Performance counter stats for 'python more_video_control_cb_perf.py':

       5130.712370      task-clock (msec)         #    0.998 CPUs utilized
               208      context-switches          #    0.041 K/sec
                 2      cpu-migrations            #    0.000 K/sec
               649      page-faults               #    0.126 K/sec
        6085730270      cycles                    #    1.186 GHz
        3769655992      instructions              #    0.62  insn per cycle

       5.142583785 seconds time elapsed

       3.944853000 seconds user
       1.188449000 seconds sys
```

5. Implement the “Bounce” code in python using Pygame and extend it to include 2 balls transparent to each other at first then colliding with one another. Also implement a physical button to bail out.

   We implemented first one_bounce.py, then two_bounce.py and two_collide.py. In two_collide.py. Technique used for velocity changing was based on ECE 4760 lab write up.



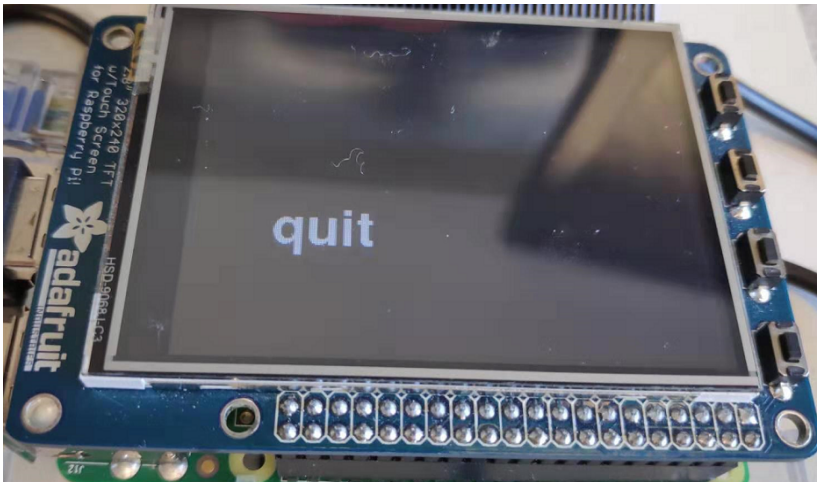
6. Follow commands to enable correct operation of touch function of piTFT.

   Here's log information after update.

```
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                      Version            Architecture Description
+++-=========================-==================-============-===============================================================
un  libsdl-1.3-0              <none>             <none>       (no description available)
ii  libsdl-image1.2:armhf     1.2.12-10+deb10u1  armhf        Image loading library for Simple DirectMedia Layer 1.2, libraries
ii  libsdl-mixer1.2:armhf     1.2.12-15          armhf        Mixer library for Simple DirectMedia Layer 1.2, libraries
ii  libsdl-ttf2.0-0:armhf     2.0.11-6           armhf        TrueType Font library for Simple DirectMedia Layer 1.2, libraries
un  libsdl1.2                 <none>             <none>       (no description available)
un  libsdl1.2-all             <none>             <none>       (no description available)
un  libsdl1.2-esd             <none>             <none>       (no description available)
un  libsdl1.2-nas             <none>             <none>       (no description available)
un  libsdl1.2-oss             <none>             <none>       (no description available)
ii  libsdl1.2debian:armhf      1.2.15-5           armhf        Simple DirectMedia Layer
un  libsdl1.2debian-all        <none>             <none>       (no description available)
un  libsdl1.2debian-alsa       <none>             <none>       (no description available)
un  libsdl1.2debian-esd        <none>             <none>       (no description available)
un  libsdl1.2debian-nas        <none>             <none>       (no description available)
un  libsdl1.2debian-oss        <none>             <none>       (no description available)
un  libsdl1.2debian-pulseaudio <none>             <none>       (no description available)
ii  libsdl2-2.0-0:armhf        2.0.9+dfsg1-1+rpt1 armhf        Simple DirectMedia Layer
~
```

7. Design a quit_button.py that display quit button on lower edge of screen. It should be designed so that touching "quit" ends the program and returns to Linux console. (Also with a physical bail-out button and a code time-out)

   quit_button.py works as expected:

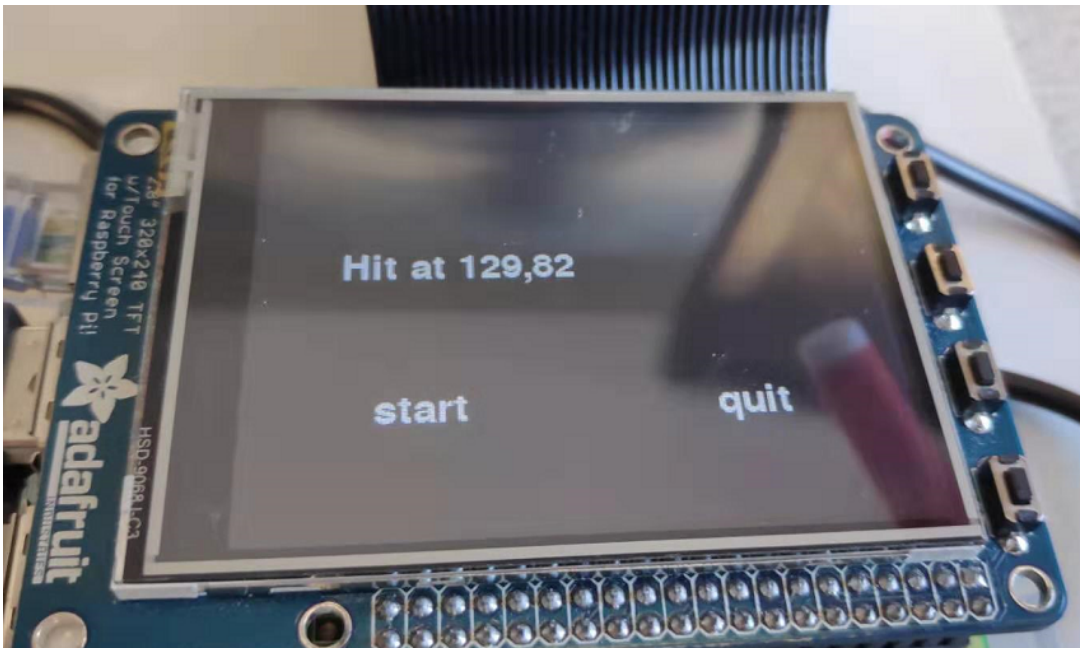


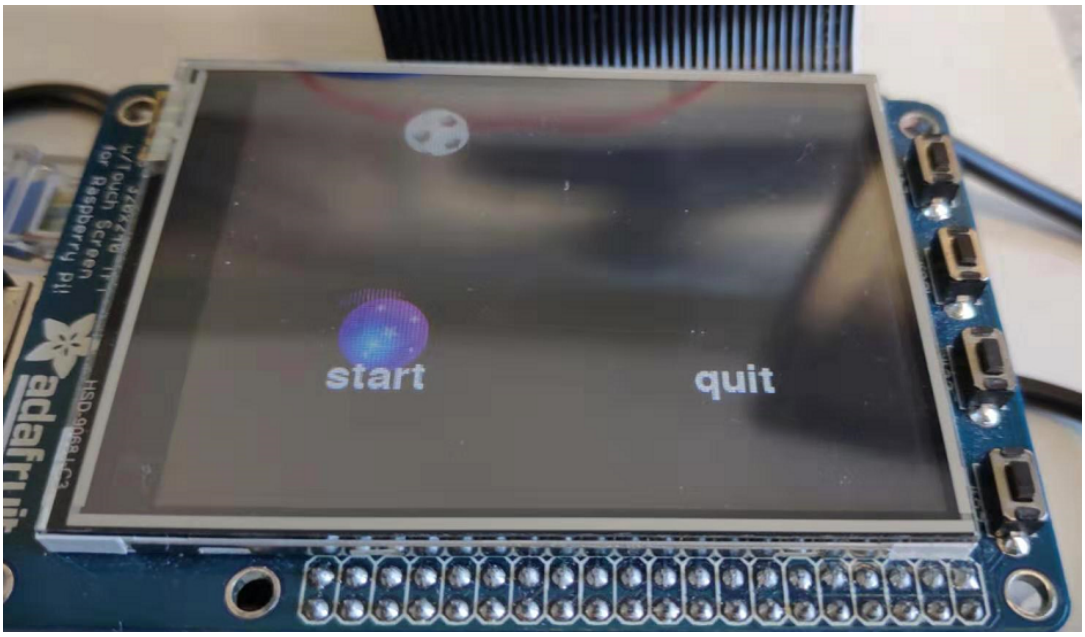
8. Expand quit_button.py into screen_coordiates.py with quit button to display "Hit at x, y" when tapping any location on screen showing coordinates of the hit.

   screen_coordinates.py worked as expected:

9. Design two_buttons.py with start and quit button. Hitting start begin playback two_collide.py, hitting quit ends the program and return to console, hitting any other location on screen displays coordinates.

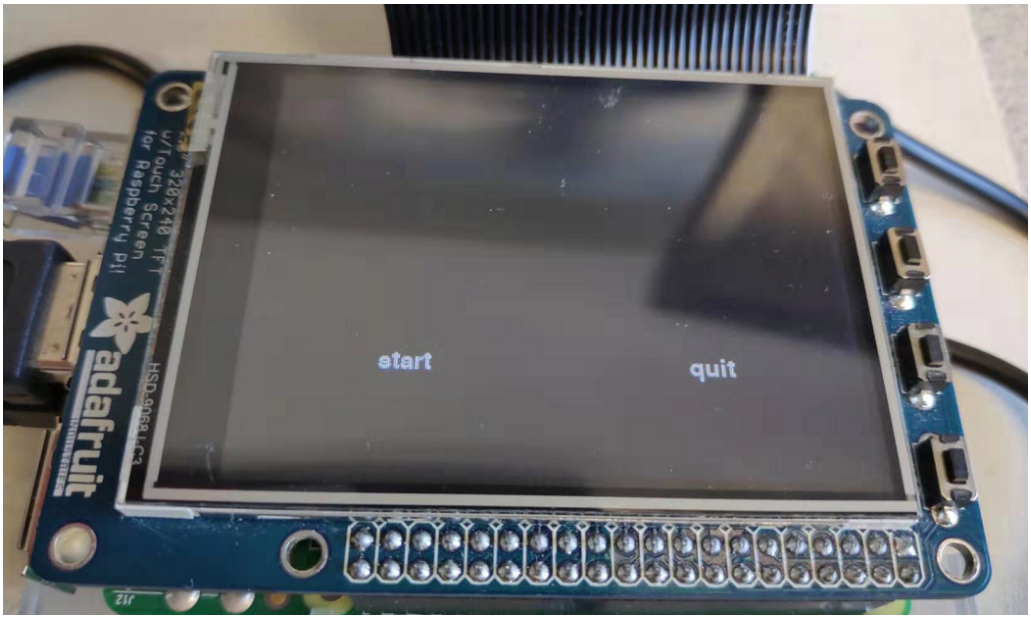
   two_buttons.py worked as expected:

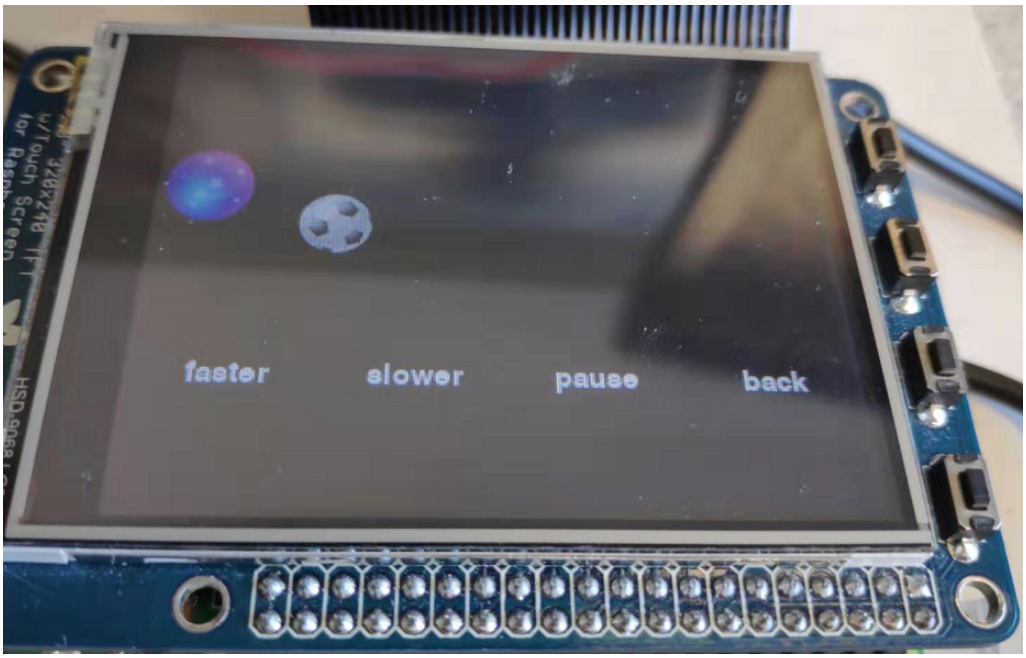




10. Design control_two_collide.py. Level 1 menu consists of start and quit button; hitting start playbacks two_collide and enter level2 menu. Level2 menu consists of “faster”, “slower”, “pause/restart” and “back” buttons to control ball speeds and to go back to menu 1.(Also a bail-out button is needed.)

    Control_two_collide worked as expected:





Conclusions:

Things that worked well:

1. Connected the breakout cable (‘Pi-cobbler’) to protoboard, and used voltmeter to check the polarity of the plug.
2. Added two buttons in GPIO19 and GPIO26, created ‘six_buttons.py’ to make sure the availability of these two buttons.
3. Extend function of ‘video_control.py’ by creating ‘more_video_control.py’, and we added two new functions to video control: fast forward 30seconds and rewind 30 seconds.
4. Modified ‘start_video’ bash script by changing ‘video_control.py’ to ‘more_video_control.py’.
5. Modified the button presses to use threaded callback interrupt routines instead of polling loop, the callback routines file is ‘more_video_control_cb.py’.
6. The callback routines operate correctly in a bash script, and name the new bash script ‘start_video_cb’, the function of

‘start_video_cb’ is the same as the original file ‘start_video’ in polling loop method.

7. Installed perf_4.18.
8. Created a python file ‘cal_v1’ and used perf to measure it’s performance.
9. Compared the polling loop and callback routines method of video_control in a fixed time with perf.
10. Implemented one/two ball(s) bouncing on the wall.
11. Implemented two balls collide with each other on the piTFT.
12. Used piTFT touch control:
    a. Created ‘quit_button.py’ : only “quit” button display on the piTFT, the program ended when we tapped ‘quit’.
    b. Created ‘screen_coordinates.py’ : If we hit quit button, program ended. Otherwise, piTFT displayed ‘hit at (current coordinates)’.
    c. Created ‘two_button.py’ : If we hit ‘start’ button, two balls began colliding on the piTFT. If we hit ‘quit’ button, program ended.
    d. Created ‘control_two_collide.py’ :
    Page 1: Only ‘start’ and ‘quit’ buttons, if we hit ‘start’, jump into page 2. If we hit ‘quit’, program ended.
    Page 2: Two balls began colliding when we jump into this page. If we hit ‘pause’, two balls stop moving. If we hit ‘fast’, their speed increased. If we hit ‘slow’, their speed decreased but has a minimum. If we hit ‘back’, jump into page 1.

Things didn’t work well at first but fixed in the end:

1. In the “two_button.py” program, at first once we hit “start”, two balls began colliding on the surface and “quit” cannot stop them.
   Reason: We defined a “collide” function, which will continuously work after start hit. However, the mouse event detecting function is only included in the main loop, in which way the

"quit" detection will not stop the collide function which is outside the main loop.
Solution: Avoid defining "collide function" outside the main loop, instead we add it into the main loop under "start hit".

2. In the "control_two_collide.py" function, once we hit the "start", surface jump into page2, in which two balls began colliding and four buttons display, but they only collide a moment then stopped.
   Reason: the condition of two balls began colliding is "start" button hit, so after one time of loop, if no mouse event detected, position of (x,y) will not be in range of the "start" button, in which way two balls will stop.
   Solution: Add "flag" to remember the "start" state, the condition of colliding will be changed to whether position in range of start or "flag" is true.
3. In the "control_two_collide.py" function, as soon as we jump into the second page, speed will change without hitting either four of the control buttons.
   Reason: after the first time we hit "start" button, the position was stored, so when we jump into the second page, the button in the similar place as "start" will automatically work.
   Solution: Set position to (-1, -1) at first in each "while loop", then detect if new button event occured.

Issues that we encountered:

1. In the "control_two_collide.py" function, we hit "slower", the speeds will gradually decreased to zero.
   Solution: Add a minimum to the speeds.

Improvements Suggestion:

1. More pygame sample programs which include more pygame function to be provided.

2. The requirement could be changed to more balls colliding, which is more close to real game.

Items could be clearer:

The requirement of “two_button.py” function is a little confusing. Not sure about whether coordinates will display after two balls begin colliding.