Pong Test - Michael Bateman

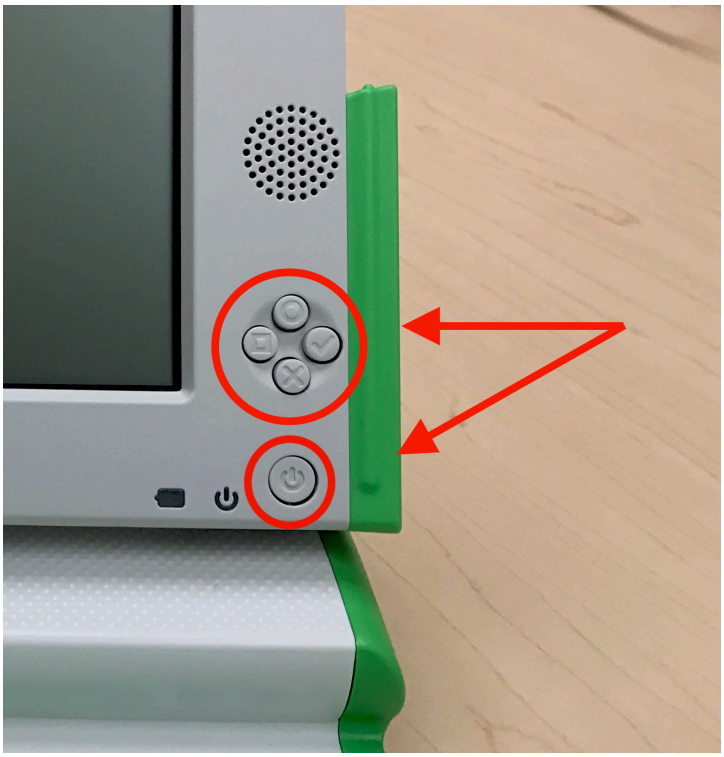
Re-image OLPC

IMPORTANT: Plug in your your OLPC before starting.

1. Turn off your OLPC by pressing the power button

2. Plug in orange USB re-image stick to the OLPC

3. Hold down the 4 game keys and the power button

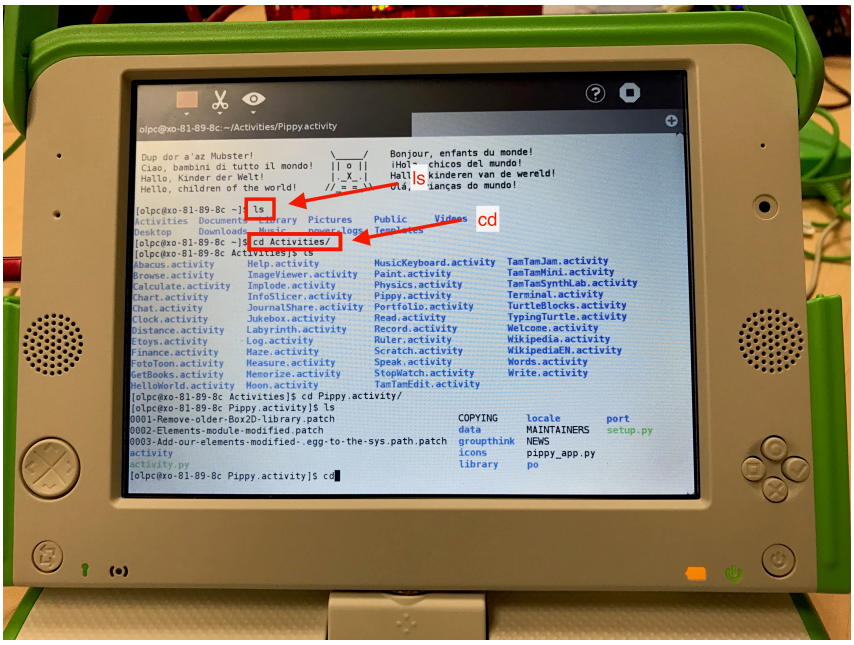


Unix Commands

We need to know unix commands to transfer the clean version of Pong on our OLPC to our Mac's.

pwd - Print Working Directory, write the full path name of the current working directory.
ls - List files, lists all files in the current working directory.
cd <directory> - Change Directory, used to change the current working directory.
cp <source_file> <target_file> - Copy files. Add -r before <source_file> to copy a directory.
mount - Allows a drive with file system to be mounted and used. Also shows the path to the drive.

Here, we are transferring the original pong file to our Mac - note: we are using the ls and cd commands



Sub-Programs

In my version of Pong, I have used many imports including the Pygame module which is responsible for the graphics. Our Mac's do not have the Pippy module installed so we cannot use that module. Instead, we have commented out the Pippy module.

```
#import pippy
import pygame
import sys
from pygame.locals import *
from random import *
```

Here, we are using the "random" module to create a random background colour:

```
# bounce on bottom
elif i == 1 \
    and ball_position[i] >= ball_limit[i] - ball_radius:
    ball_position[i] = ball_limit[i] - ball_radius - 1
    ball_mvect[i] = -1 * ball_mvect[i]
    Background_Red = randint(0,254)
    Background_Green = randint(0,254)
    Background_Blue = randint(0,254)
```

Loops

while loop - Repeats statements or a group of statements while a condition is true.
for loop - Executes a sequence of statements a certain amount of times.
nested loops - A for or while loop contained in another loop.

In my game of Pong, I used both while and for loops. For one "while loop" we needed to change it because we do not have Pippy installed. We changed it to, "while True:".

```
textRect.centerx = screen.get_rect().centerx
textRect.centery = screen.get_rect().centery

#while pippy.pygame.next_frame():
while True:

    # display msg
```

Variables

A variable is used to store information that can be used many times in a program. Variables are useful because if you want to change one thing (ex. length of line) then you can just adjust the variable, instead of sorting through your whole code.

Below, is an image of the original background colours, set to black, before the ball hits the walls and are turned into background colours:

```
# ball constants
ball_color = (250, 250, 250)
ball_radius = 25

Background_Red = 0
Background_Green = 0
Background_Blue = 0

# game constants
fsize = 48
msg = 'Press \'s\' to start game'
```

Functions

A function is reusable code that can be performed multiple times. Functions allow you to use code reusing. I did not use a function in my code, but I will give an example of a function

```
def get_RGB()
        r = randint(0,255)
        g = randint(0,255)
        b = randint(0,255
        screen.fill((r,g,b))
```

Errors

There are 3 main types of errors. They are - syntax, logic, and run-time. While programming my pong, I found 2 errors. They were syntax errors and run time errors. I did not run into any logic errors.

syntax - Python cannot understand your commands. Most likely caused by spelling mistake, or missing characters.
logic - A logic error is a bug in the program that causes it to operate incorrectly but not crash the program. It can also produce unwanted output or other behaviour.
run time - A runtime error occurs while the program is running. This is usually when a file is called but it does not exist.

Below is a syntax error I ran into, because I forgot to put a colon.

```
Michaels-MacBook-Air:pong_test michael.bateman$ python pong.py
  File "pong.py", line 188
    if balls == 0
                ^
SyntaxError: invalid syntax
Michaels-MacBook-Air:pong_test michael.bateman$
```

RGB - Red Green Blue Color

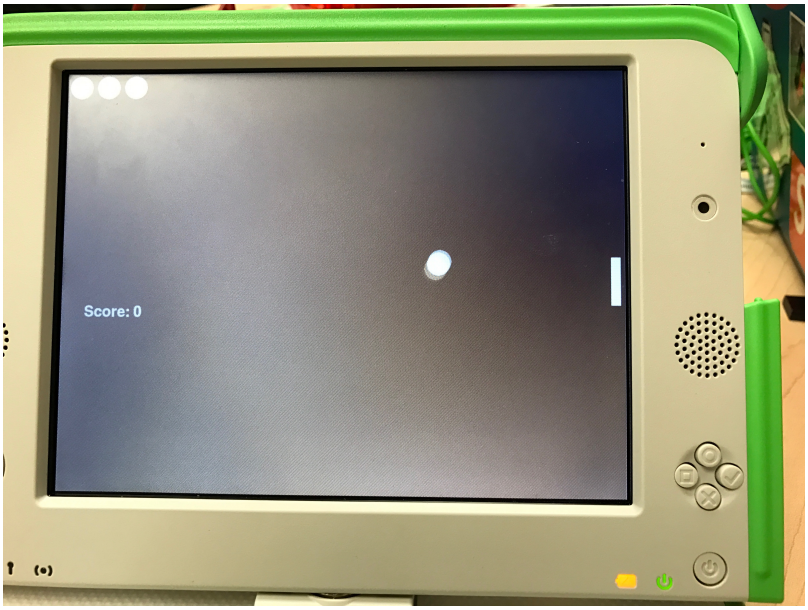
RGB stands for red, green, and blue. RGB is the colour that are used on a computer screen. They can be combined to make any colour that is in the visible spectrum.

I used RGB to make the screen background random. In the image below, you will see that I made a random colour every time the ball hit a wall.

```
# bounce on bottom
elif i == 1 \
    and ball_position[i] >= ball_limit[i] - ball_radius:
    ball_position[i] = ball_limit[i] - ball_radius - 1
    ball_mvect[i] = -1 * ball_mvect[i]
    Background_Red = randint(0,254)
    Background_Green = randint(0,254)
    Background_Blue = randint(0,254)
```

Final Product - Pictures

Below, I am running my code on the OLPC. Note, every time the ball hit's the wall, the colour changes.



Final Product - Code

```
#-------------
#Michael Bateman
#Janurary 19, 2017
#-------------



# pong: hit the ball with the paddle
#
# use the escape key to exit
#
# on the XO, the escape key is the top lefthand key,
# circle with an x in it.

#import pippy
import pygame
import sys
from pygame.locals import *
from random import *

# always need to init first thing
pygame.init()

# create the window and keep track of the surface
# for drawing into
screen = pygame.display.set_mode((0, 0), pygame.FULLSCREEN)

# ask for screen's width and height
size = width, height = screen.get_size()

# turn off the cursor
pygame.mouse.set_visible(False)

# turn on key repeating (repeat 40 times per second)
pygame.key.set_repeat(25, 25)

# start the screen all black
bgcolor = (0, 0, 0)
screen.fill(bgcolor)

# paddle constants
paddle_width = 20
paddle_length = 100
paddle_radius = paddle_length / 2
paddle_color = (250, 250, 250)
step = 6  # paddle moves 3 pixels at a go

# ball constants
ball_color = (250, 250, 250)
ball_radius = 25

Background_Red = 0
Background_Green = 0
Background_Blue = 0
```

```
# game constants
fsize = 48
msg = 'Press \'s\' to start game'

font = pygame.font.Font(None, fsize)
text = font.render(msg, True, (250, 250, 250))
textRect = text.get_rect()
textRect.centerx = screen.get_rect().centerx
textRect.centery = screen.get_rect().centery

#scoreboard - MB
score = 0
scoretext = font.render("Score: " + str(score), True, (250, 250, 250))
scoreRect = scoretext.get_rect()
scoreRect.x = 50
scoreRect.y = 500

#while pippy.pygame.next_frame():
while True:

    # display msg
    screen.fill(bgcolor)
    screen.blit(text, textRect)
    pygame.display.flip()

    # chill until a key is pressed
    for idle_event in pygame.event.get():
        if idle_event.type == QUIT:
            sys.exit()

        if idle_event.type == KEYDOWN:
            if idle_event.key == K_ESCAPE:
                sys.exit()

            if idle_event.key == 115:  # g key

                # play a game!

                # start the paddle in the center
                paddle_location = height / 2

                # number of balls to a game
                balls = 4

                while balls > 0:

                    ball_position = [ball_radius, ball_radius]
                    ball_mvect = [randint(3, 5), randint(3, 5)]
                    ball_limit = size
                    balls = balls - 1

                    while ball_position[0] + ball_radius < ball_limit[0]:  # in play

                        for event in pygame.event.get():
```

```
for event in pygame.event.get():
    if event.type == QUIT:
        sys.exit()

    elif event.type == KEYDOWN:
        if event.key == K_ESCAPE:
            sys.exit()
        elif event.key == 273 \
            or event.key == 265 \
            or event.key == 264:  # up
            paddle_location = paddle_location - step
        elif event.key == 274 \
            or event.key == 259 \
            or event.key == 258:  # down
            paddle_location = paddle_location + step

# make sure the paddle is in-bounds
if paddle_location - paddle_radius < 0:
    paddle_location = paddle_radius
elif paddle_location + paddle_radius >= height:
    paddle_location = height - 1 - paddle_radius

# clear the screen

#screen.fill(bgcolor)

screen.fill((Background_Red,Background_Green,Background_Blue))

#show score on screen - MB
scoretext = font.render("Score: " + str(score), True, (250, 250, 250))
screen.blit(scoretext,scoreRect)

# draw the paddle on the right side of the screen
pygame.draw.line(screen,
                 paddle_color,
                 (width - paddle_width, paddle_location -
                 paddle_radius),
                 (width - paddle_width,
                 paddle_location + paddle_radius),
                 paddle_width)

# draw the ball
pygame.draw.circle(screen, ball_color, ball_position, ball_radius)

# draw the unused balls
for i in range(balls):
    pygame.draw.circle(screen, ball_color,
         (int(round(30 + i * ball_radius * 2.4)), 30),
         ball_radius)

# update the display
pygame.display.flip()

screen.fill((Background_Red,Background_Green,Background_Blue))
```

```
                # update the ball
                for i in range(2):
                    ball_position[i] = ball_position[i] + ball_mvect[i]

                    # bounce on top and left
                    if ball_position[i] < ball_radius:
                        ball_position[i] = ball_radius
                        ball_mvect[i] = -1 * ball_mvect[i]
                        Background_Red = randint(0,254)
                        Background_Green = randint(0,254)
                        Background_Blue = randint(0,254)

                    # bounce on bottom
                    elif i == 1 \
                        and ball_position[i] >= ball_limit[i] - ball_radius:
                        ball_position[i] = ball_limit[i] - ball_radius - 1
                        ball_mvect[i] = -1 * ball_mvect[i]
                        Background_Red = randint(0,254)
                        Background_Green = randint(0,254)
                        Background_Blue = randint(0,254)

                    elif i == 0 \
                        and ball_position[i] >= ball_limit[i] - ball_radius - paddle_width \
                        and ball_position[1] > paddle_location - paddle_radius \
                        and ball_position[1] < paddle_location + paddle_radius:
                        ball_position[i] = ball_limit[i] - ball_radius - paddle_width - 1
                        ball_mvect[i] = (-1) * ball_mvect[i]

                        #add score every time - MB
                        score = score + 1

            if balls == 0:
                screen.fill((0,0,0))
                score = 00
```

You can view the source code here: https://github.com/michael-bateman/pong-python