

Modifying Pong to meet new specifications

By Simon Guo and Jesse Zhao



Description:

Pong is a really famous and original game. In the samples of Pippy on OLPC, there is also a Pong game in the graphics categories. In order to understand python better, we started a project to modify pong through coding in python.

What is Python?

Python is a **widely used, general-purpose, high-level programming language**. Its **design emphasizes code readability**, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. Python is an **object-oriented programming language**. It has gained popularity because of its **clear syntax and readability**.

What is Pong?

Pong is one of the first computer games that was ever created, this simple "tennis like" game usually features two paddles and a ball, or one paddle and a few lives and a close edged wall for the ball to bounce off of. The goal is to **prevent the ball from going on the line that your paddle is moving on**. Once you lose for a certain amount of times, it's game over!

What is Pygame?

Pygame is a **cross-platform set of Python modules designed for writing video games**. It includes computer graphics and sound libraries designed to be used with the Python programming language. Pygame adds functionality on top of the excellent SDL library. This allows you to **create fully featured games and multimedia programs in the python language**. Pygame is **highly portable and runs on nearly every platform and operating system**. Pygame itself has been downloaded millions of times, and has had millions of visits to its website.

Where do I get the Pong from?

We need to get Pong from the OLPC, using linux commands to find them and copy to the usb device.

Linux/Unix commands

These are linux and unix basic commands to access file, switch directory, and run program.

pwd outputs the name of the current working directory.

ls lists all files and directories in the working directory.

cd switches you into the directory you specify.

cd.. move up one directory

mkdir creates a new directory in the working directory.

cp sourcefile/ targetfile copies files or directories.

rm remove files or directories

cat Concatenate files and print on the standard output

mount list directory for system hardware and outside devices

program name + file name use program to run a file

How to transfer a file from OLPC to a computer

1. Determine the path name to the USB stick using the mount command

```
[olpc@xo-4f-ff-dc graphics]$ mount
```

Mount is the command you can use to find outside device connected to the computer.

after typing mount, we can find a directory of our usb stick ("TRANSFER")

```
tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755)
tmpfs on /sys/fs/cgroup type tmpfs (rw,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,release_agent=/usr/lib/systemd/systemd-cgroups-agent,name=systemd)
cgroup on /sys/fs/cgroup/cpu type cgroup (rw,nosuid,nodev,noexec,relatime,cpu)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
vartmp on /var/tmp type tmpfs (rw,relatime,size=51200k)
/tmp on /tmp type tmpfs (rw,relatime,size=51200k)
varlog on /var/log type tmpfs (rw,relatime,size=20480k)
none on /var/lib/stateless/writable type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/cache/man type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/lib/xbk type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/lib/dhclient type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /etc/adjtime type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/lib/logrotate.status type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/spool type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
/dev/mmcblk0p1 on /etc/ssh type ext4 (rw,relatime)
/dev/mmcblk0p1 on /etc/hosts type ext4 (rw,relatime)
/dev/mmcblk0p1 on /var/lib/dbus type ext4 (rw,relatime)
/dev/mmcblk0p1 on /var/lib/random-seed type ext4 (rw,relatime)
/dev/mmcblk0p1 on /etc/NetworkManager/system-connections type ext4 (rw,relatime)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
/dev/sda1 on /run/media/olpc/TRANSFER type vfat (rw,nosuid,nodev,relatime,uid=1000,gid=1000,mask=0022,dmask=0077,codepage=cp437,iocharset=ascii,shortname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks2)
```

2. Find the directory of Pong

The Pong program is in the Pippy data directory, we need to navigate to /home/olpc/Activities/Pippy.activity/data/graphics using the cd Linux command

```
[olpc@xo-4f-ff-dc ~]$ cd /home/olpc/Activities/Pippy.activity/data/graphics
[olpc@xo-4f-ff-dc graphics]$ pwd
/home/olpc/Activities/Pippy.activity/data/graphics
```

3. Copy pong to the USB as pong using the Linux cp command

```
[olpc@xo-4f-ff-dc graphics]$ cp pong /run/media/olpc/TRANSFER/.
```

Use cp command to copy pong from current directory to new directory with the same name.

4. Check if the file is in the USB

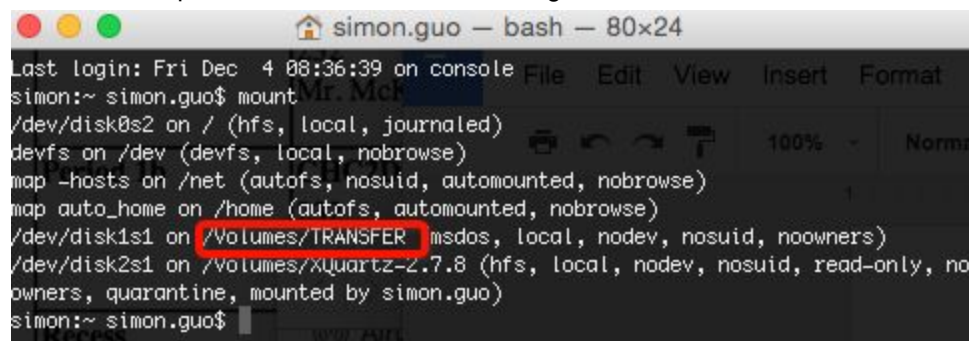
```
[olpc@xo-4f-ff-dc graphics]$ cd /run/media/olpc/TRANSFER/
[olpc@xo-4f-ff-dc TRANSFER]$ ls
pascal
pong
pygame-1.9.2pre-py2.7-macosx10.7.mpkg.zip
Screenshot of _Pippy Activity_1.png
Screenshot of _Pippy Activity_.png
Screenshot of _Snow_1.png
Screenshot of _Snow_2.png
Screenshot of _Snow_.png
Screenshot of _Terminal Activity_1.png
Screenshot of _Terminal Activity_2.png
Screenshot of _Terminal Activity_3.png
Screenshot of _Terminal Activity_.png
XQuartz-2.7.8.dmg
[olpc@xo-4f-ff-dc TRANSFER]$
```

Use cd command to go to USB directory, and then use ls to list all file to see the file is in or not.

How to transfer a file from USB to a computer

1. Find the path of USB by mount command

Determine the path name to the USB stick using the mount command



```
simon.guo — bash — 80x24
Last login: Fri Dec 4 08:36:39 on console
simon:~ simon.guo$ mount
/dev/disk0s2 on / (hfs, local, journaled)
devfs on /dev (devfs, local, nobrowse)
map -hosts on /net (autofs, nosuid, automounted, nobrowse)
map auto_home on /home (autofs, automounted, nobrowse)
/dev/disk1s1 on /Volumes/TRANSFER msdos, local, nodev, nosuid, noowners
/dev/disk2s1 on /Volumes/XQuartz-2.7.8 (hfs, local, nodev, nosuid, read-only, no
owners, quarantine, mounted by simon.guo)
simon:~ simon.guo$
```

2. Navigate to the USB using the cd command

```
simon:~ simon.guo$ cd /Volumes/TRANSFER
simon:TRANSFER simon.guo$ pwd
/Volumes/TRANSFER
```

We find TRANSFER under Volumes directory.

3. Copy pong.py from the USB to ~/pong.py using the Unix cp command

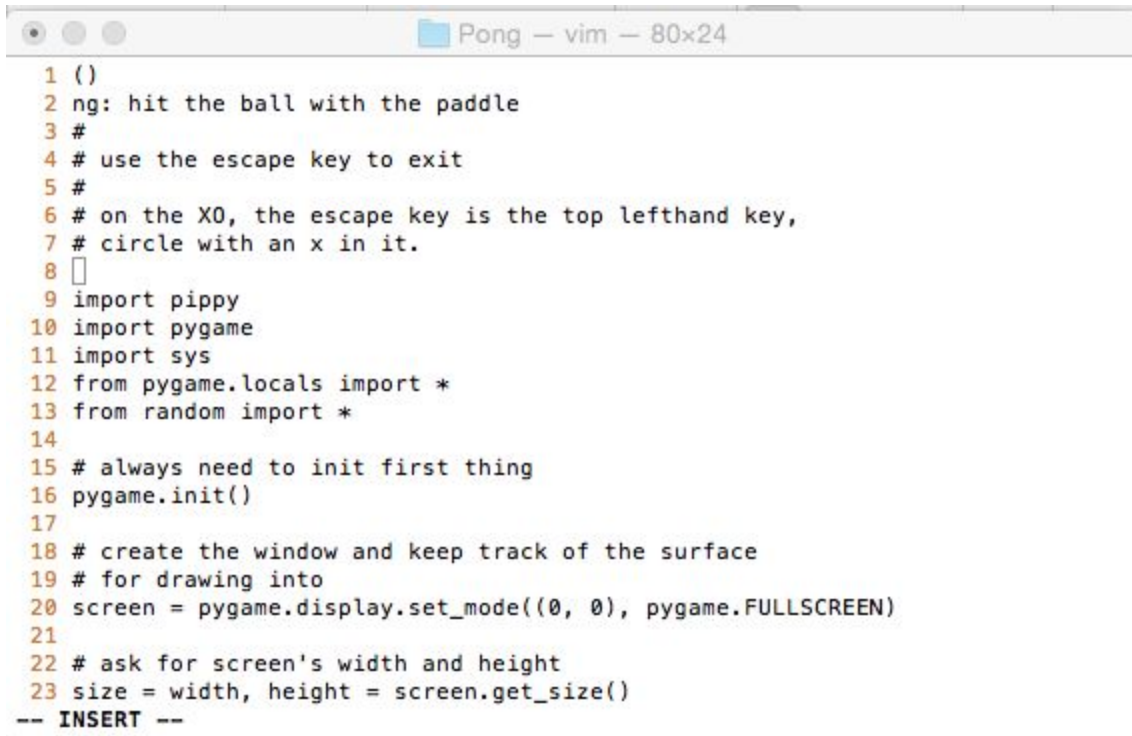
```
simon:TRANSFER simon.guo$ cp pong ~/pong.py
```

instead of /., we change the name of pong to pong.py, with .py file type, it is easier to recognize and run. We will copy that to the compsci folder later, so that we can organize our programs more easily.

How do I edit the code, debug, and run?

Through terminal, we can use vi editor to edit it.

```
simon:pong simon.guo$ vi PONG 0.0.py
```



```
1 ()
2 ng: hit the ball with the paddle
3 #
4 # use the escape key to exit
5 #
6 # on the X0, the escape key is the top lefthand key,
7 # circle with an x in it.
8
9 import pippy
10 import pygame
11 import sys
12 from pygame.locals import *
13 from random import *
14
15 # always need to init first thing
16 pygame.init()
17
18 # create the window and keep track of the surface
19 # for drawing into
20 screen = pygame.display.set_mode((0, 0), pygame.FULLSCREEN)
21
22 # ask for screen's width and height
23 size = width, height = screen.get_size()
-- INSERT --
```

Also, we can run it by type python PONG 0.0.py

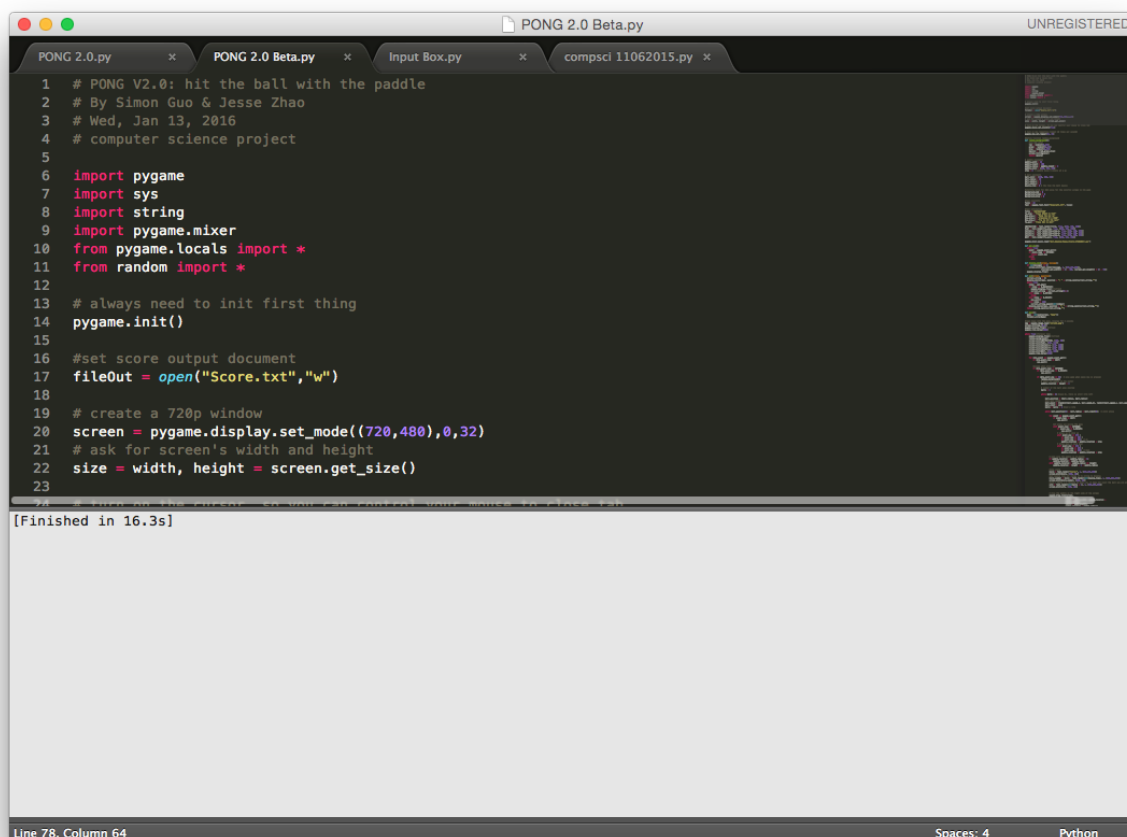
```
simon:pong simon.guo$ python PONG 0.0.py
```

VI editor is the basic editor in the Linux system. It is really basic. It is really complicated when you switch between writing and running. It takes a lot of extra time. Also, it is hard to edit, because you cannot just use normal command for example command+c to copy. It is not really efficient when you do a lot of programming.

We can solve these problems by downloading a programming IDE.

IDE stands for integrated development environment, it is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger.

Sublime Text Editor is a cross-platform source code editor with a Python application programming interface (API). It natively supports many programming languages and markup languages, and its functionality can be extended by users with plugins, typically community-built and maintained under free-software licenses.



The top half is programming space, you can write code there. The bottom half (can be hidden) is running and compile space, which shows error and run time. When you need to run the program, you just need to press command + B to start building it. It is really easy and useful.

Error type

There are three errors that we experienced when modifying the program:

1. Syntax In linguistics, syntax is the set of rules, principles, and processes that govern the structure of sentences in a given language, specifically word order.

A syntax error is a character or string incorrectly placed in a command or instruction that causes a failure in execution.

This is an easy error to fix, some IDE will show you the syntax error, you just need to fix it.

2. Logic is the branch of philosophy concerned with the use and study of valid reasoning. The study of logic also features prominently in mathematics and computer science.

A logic error is a bug in a program that causes it to operate incorrectly, but not to terminate abnormally or crash. A logic error produces unintended or undesired output or other behaviour, although it may not immediately be recognised as such.

This is harder to solve, you need to show yourself the program algorithm again to figure out what is wrong.

3. Run time is the time during which a program is running (executing), in contrast to other program lifecycle phases such as compile time, link time and load time.

A runtime error is a problem that prevents a program from working correctly. Runtime errors might cause you to lose information in the file you're working on, cause errors in the file (corrupt the file) so you can't work with it, or prevent you from using a feature.

It is more unlikely to get a Runtime error than the first two, it has many problems related to it. It may be an extra forever loop, or perhaps logic error causing a runtime error which might make the program delay for a long time.



Specific experience with debugging will be showed later

Before Writing the code

SSRGMIO

Python is a Sequential, Selective(if statements), Repetitive, Graphics, Modular(function, actions), Input, Output function.

We try to demonstrate these seven key elements in our programs.

Mindstorm

Before we start our project, we discussed what kind of functions we want the program to have. These are some of them:

- Game's instruction give a brief introduction about how to play the game
- Game's information give some information for the game and author
- Smaller window make a smaller window, easier to play and debug
- Colorful background make the game look fancy
- Score system add score in it to make the game competitive
- Pixel font make pixel time to let it look old-timed
- Increasing difficulty make the game more difficult
- Better game interface change constants of balls and paddle to make look more friendly
- ball sound but sound when the ball bounce on the paddle
- Game over display Game Over when the game is over
- Output High Score put your high score to a document
- Output Your name put your name to a document

Technology Requirements of Python Pong

Before running the game, we need to install XQuartz and Pygame.

XQuartz

The XQuartz project is an open-source effort to develop a version of the X.Org X Window System that runs on OS X. Together with supporting libraries and applications, it forms the X11.app that Apple shipped with OS X versions 10.5 through 10.7.

<http://www.xquartz.org/>

Pygame

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. This is based on the assumption that the most expensive functions inside games (mainly the graphics part) can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.

<http://pygame.org/hifi.html>

PONG

After that, you are ready to go!

You can either use linux command in terminal or build shortcut in sublime text.(mentioned in vi editor and sublime editor)

Below are the different versions we modified.produced except for the original one.

Version 0.0

Description: a fresh version of Pong, copy from olpc. Cannot run on Mac OS, because pippy IDE is not installed in OLPC

pong: hit the ball with the paddle

use the escape key to exit

on the XO, the escape key is the top left-hand key,

circle with an x in it.

#import files

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) #This makes the screen all black (0, 0, 0) is black.

screen.fill(bgcolor) #This fills the screen black

paddle constants

paddle_width = 20 #this is setting the paddle width on the game to be 20 units

paddle_length = 100 #this is setting the paddle length on the game to be 100 units

paddle_radius = paddle_length / 2 #

paddle_color = (250, 250, 250) # This colors the paddle by implementing the RGB color code (in this case the color would be white)

step = 6 #paddle moves 3 pixels at a go

```

# ball constants
ball_color = (250, 250, 250) #defining the ball's color which is white
ball_radius = 25 #defining the ball's radius which is 25 units

# game constants
fsize = 48 #this is the font size for the words displayed
msg = 'Press \'g\' to start game' #displays this message when you open the game
#set font information
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
#launch pygame in pippy
while pippy.pygame.next_frame():

    # 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 == 103: # 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: #you are still alive or have a few lives

```

```

ball_position = [ball_radius, ball_radius]
ball_mvect = [randint(3, 5), randint(3, 5)] #set ball speed
ball_limit = size #paddle did not meet the ball
balls = balls - 1 #lost one ball

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

    for event in pygame.event.get():
        if event.type == QUIT:
            sys.exit()
            #press esc to quit
        elif event.type == KEYDOWN:
            if event.key == K_ESCAPE:
                sys.exit()
            elif event.key == 273 \
                or event.key == 265 \
                or event.key == 264: # press up to go up
                paddle_location = paddle_location - step
            elif event.key == 274 \
                or event.key == 259 \
                or event.key == 258: # press down to go 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)

    # 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()

# 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]
# 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]

# bounce on paddle
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]

```

How the code works?

The code works by scanning and calculating the ball position, with different special situation in it, like esc keys. It have major two parts, the wait part and play part. The wait part and play part runs sequentially in a forever while loop. Inside the while loop, there are several nested selective loop, which give ball different action.

Version 1.0

Description: a version that stops using pippy library and could run on OS X.

```
8 import pippy -----> 8 #import pippy
```

do not import pippy anymore, we can get library from pygame

```
55 while pippy.pygame.next_frame(): 55 while True: #pippy.pygame.next_frame():
```

Change while pippy is available to while True, which both runs forever loop, but while True is recognized on OS X

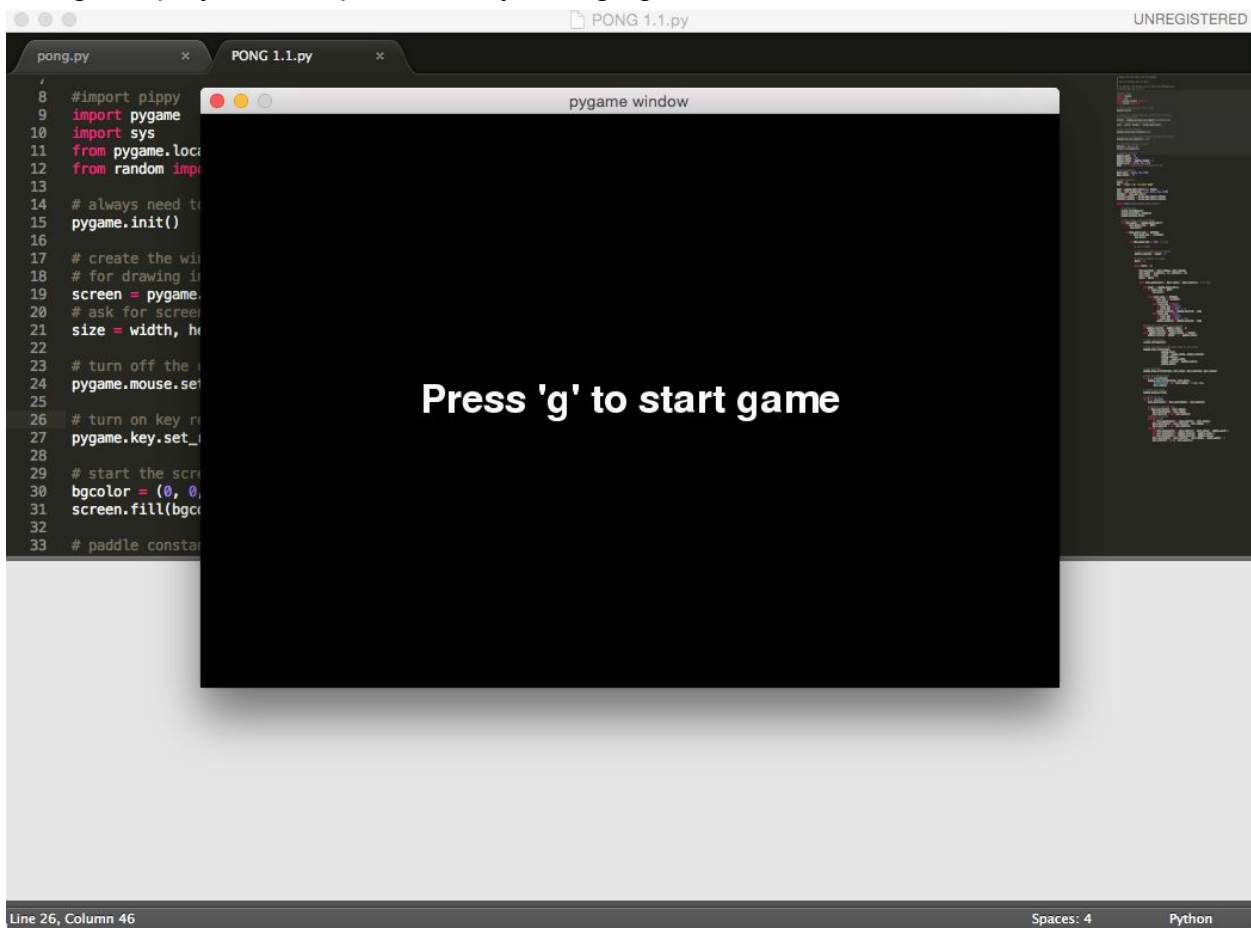
Benefits: We can play on OS X

Version 1.1 smaller playing window

Description: a version that has a smaller window and visible mouse to play

```
19 screen = pygame.display.set_mode((0, 0), pygame.FULLSCREEN)
19 screen = pygame.display.set_mode((720,480),0,32)
```

change display to a 720p window by changing set mode.



Also, I make the mouse visible, by changing the boolean.

```
25 pygame.mouse.set_visible(False)
```

```
25 pygame.mouse.set_visible(True)
```

Benefits: A smaller window will make the program launch faster, easier to play and debug. Visible mouse help us to close the window easily.

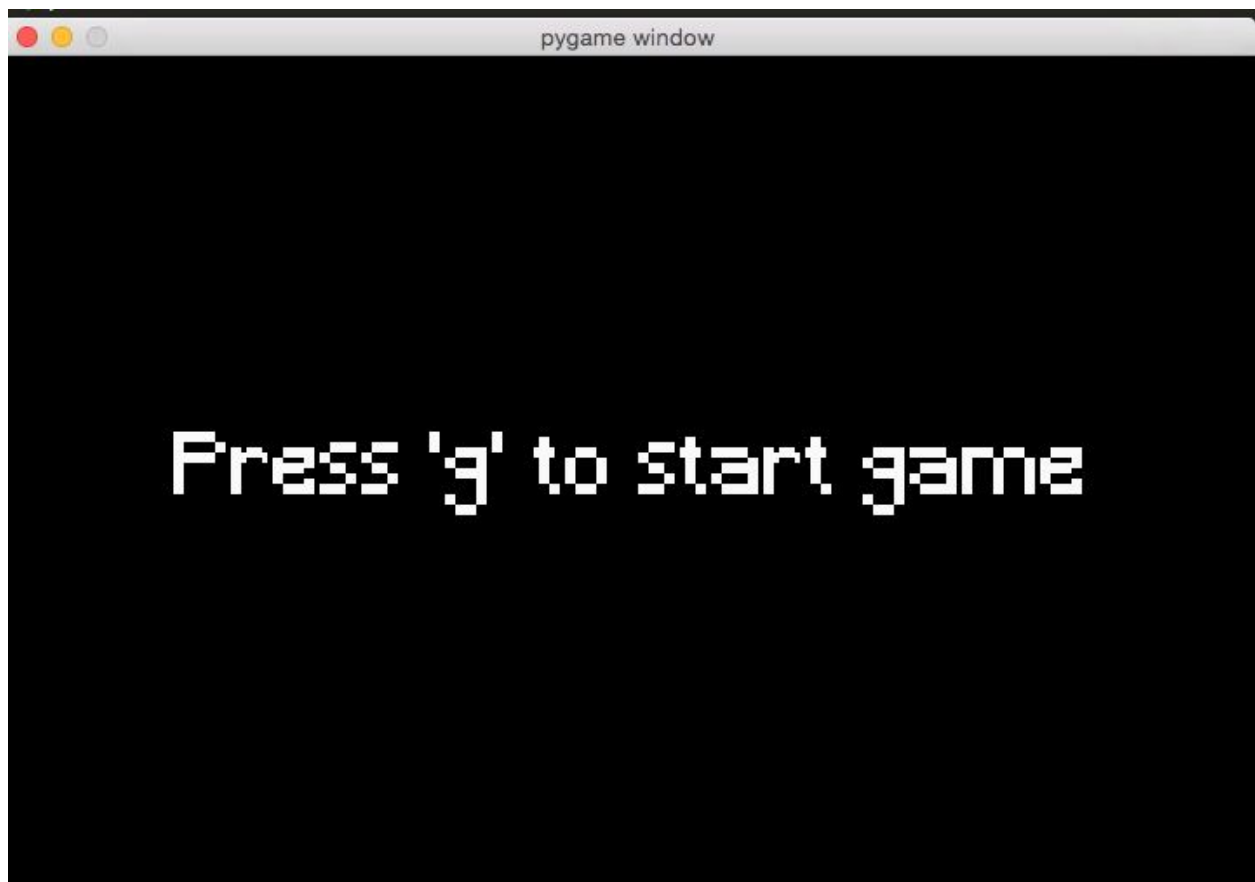
Version 1.2 Font

Description: a version that use pixel font, also improves moving speed and size.

```
48 font = pygame.font.Font(None, fsize)
```

```
48 font = pygame.font.Font("Minecraft.ttf", fsize)
```

By downloading Minecraft.ttf, we can make the font have pixel look.



Benefits: Changing ball speed and size make the game more friendly. The font make it looks a like an old-time game(it actually is).

Version 1.3 Improved game difficulty

Description: a version that increases the game difficulty. The ball will increase speed as the more life is lost.

```
83 ball_mvect = [randint(3, 5), randint(3, 5)]
```

```

152 elif i == 0 \
153     and ball_position[i] >= ball_limit[i] - ball_radius - paddle_width \
154     and ball_position[1] > paddle_location - paddle_radius \
155     and ball_position[1] < paddle_location + paddle_radius:
156     ball_position[i] = ball_limit[i] - ball_radius - paddle_width - 1
157     ball_mvect[i] = (-1) * ball_mvect[i]
158     step = step + 0.2
159     ball_speed_1 = ball_speed_1 + 10
160     ball_speed_2 = ball_speed_2 + 10
161

```

The ball speed will increase as the more life is lost.

The paddle will also increase speed to make the game better for playing.

Benefit: The game will become more challenging.

Version 1.5 Score Display

Description: It will display the score on the top of the window, and the life left beside the balls.

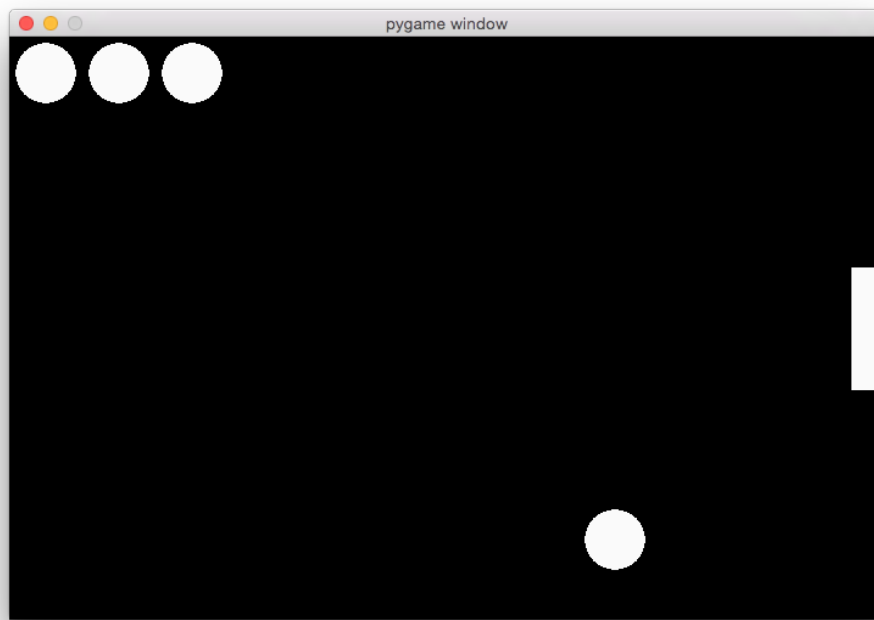
These codes are for displaying score and life. Because the numbers are integers, we need to change them to string to display.

```

120 Score = font.render("Score:", 1, (255,255,255))
121 screen.blit(Score, (280, 10))
122
123
124
125 Score_number = Score = font.render(str(bounce_time), 1, (255,255,255))
126 screen.blit(Score_number, (440, 10))
127
128
129 Life = font.render(str(balls + 1), 1, (255,255,255))
130 screen.blit(Life, (190, 10))
131
132 screen.fill(bgcolor)
133 # clear the screen
134

```

I met a problem. The Score does not show up. It is a **Logic Error**.



I think the screen fill function has some relation to it. So I try to disable this function and see what happened.

```
132 #screen.fill(bgcolor)
133 # clear the screen
```



I noticed that the screen fill function cover the graphics were drawn.

I changed the order of screen fill and display score. And it works.

```
118 screen.fill(bgcolor)
119 # clear the screen
120
121 Score = font.render("Score:", 1, (255,255,255))
122 screen.blit(Score, (280, 10))
123
124 Score_number = Score = font.render(str(bounce_time), 1, (255,255,255))
125 screen.blit(Score_number, (440, 10))
126
127 Life = font.render(str(balls + 1), 1, (255,255,255))
128 screen.blit(Life, (190, 10))
129
```

On line 127, I put a balls +1, because the balls in the code does not count the ball that is being played by you at that time.

```
46 bounce_time = 0
```

We set bounce_time as a variable. It will begin from 0. When the ball hit the paddle, it will add one.

```
bounce_time = bounce_time + 1
```

So this is a sequential problem. Now the screen will cover the path right now and display the score later.



Version 1.6 colorful introduction background

Description: This version has a colorful intro page instead of the black background. It will change the color every 1.5 seconds, which make the game look more fancy.

RGB: color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue. Each of red, green, and blue color is from 0 to 255. Different combination can make different colors together.

```
29 def random_background():
30
31     red = randint(0,255)
32     green = randint(0,255)
33     blue = randint(0,255)
34     bgcolor = (red,green,blue)
35     screen.fill(bgcolor)
36     return bgcolor
37
```

I defined a function called random background. It will make a random color and fill it on screen.

```
56 Title = 'INSTRUCTION'
57 To_play = 'Press SPACE to PLAY'
58 How_play_1 = 'Move the bar by'
59 How_play_2 = 'pressing UP & DOWN'
60 How_play_3 = 'Try to hit the ball'
61 To_quit = 'Press ESC to QUIT'
62
63 font = pygame.font.Font("Minecraft.ttf", fsize)
64
65 INSTRUCTION = font.render(Title, True, (250, 250, 250))
66 PLAY = font.render(To_play, True, (250, 250, 250))
67 Content_1 = font.render(How_play_1, True, (250, 250, 250))
68 Content_2 = font.render(How_play_2, True, (250, 250, 250))
69 Content_3 = font.render(How_play_3, True, (250, 250, 250))
70 QUIT = font.render(To_quit, True, (250, 250, 250))
71
```

I put several sentences on the colorful background, and change the start key to space.

```
72 while True: #pippy.pygame.next_frame():
73
74     pygame.display.flip()
75     random_background()
76     screen.blit(INSTRUCTION, (185, 60))
77     screen.blit(PLAY, (120, 145))
78     screen.blit(Content_1, (170, 210))
79     screen.blit(Content_2, (140, 260))
80     screen.blit(Content_3, (170, 310))
81     screen.blit(QUIT, (150, 380))
82     pygame.time.delay(1500)
```

It will keep showing them until the space key or an esc key is pressed.



Benefit: The space key is more common than the g key. Also, we made a fancy instruction page.

Version 1.6 Beta colorful game background

Description: The gaming background will change color when the ball hits the paddle

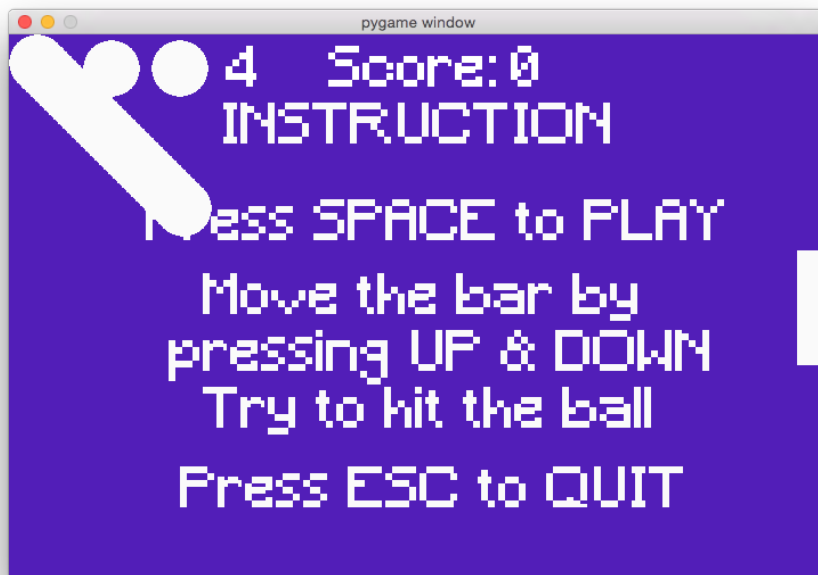
I met a logic error again. put the random_background inside the program. But it seems like it is always flashing or not refreshing the screen.

```
29 # start the screen all black
30 bgcolor = (0, 0, 0)
31 screen.fill(bgcolor)
```

```
29 def random_background():
30
31     red = randint(0,255)
32     green = randint(0,255)
33     blue = randint(0,255)
34     bgcolor = (red,green,blue)
35     screen.fill(bgcolor)
36     return bgcolor
```

```
123
124
125     screen.fill(bgcolor)
126     # clear the screen
127
```

```
136     random_background()
137     # clear the screen
138
```



I think this is happening because two subprograms are using one value, which can lead to a logic error that the color changes really fast(no delay).

The two sub programs are taking one set of random numbers and the variable cannot pass data to both of them without will defined.

I struggled on this for a long time. Finally, i come with an idea. I set another set of variables, which begin with black, then input the value when it hits the paddle. In that way, the program can get its own data and decide what to do.

```
53 Background_Red = 0
54 Background_Green = 0
55 Background_Blue = 0

174 screen.fill((Background_Red,Background_Green,Background_Blue))
175 # clear the screen

196
197 elif i == 0 \
198     and ball_position[i] >= ball_limit[i] - ball_radius - paddle_width \
199     and ball_position[1] > paddle_location - paddle_radius \
200     and ball_position[1] < paddle_location + paddle_radius:
201     ball_position[i] = ball_limit[i] - ball_radius - paddle_width - 1
202     ball_mvect[i] = (-1) * ball_mvect[i]
203     step = step + 0.2
204     bounce_time = bounce_time + 1
205     Background_Red = randint(0,254)
206     Background_Green = randint(0,254)
207     Background_Blue = randint(0,254)
208
```

It works and look like that.



Version 1.7 Game over

Description: In this version, it will show game over when you use up your life.

I first try to use a while loop, inside the big forever loop. Although, I put a break, it does not break out the loop after it shows game over.

```
206         while balls < 1:
207             #break
208             GameOver = 'Game Over'
209             text = font.render('GameOver', True, (250, 250, 250))
210             screen.blit(text, (170, 210))
211             random_background()
212             pygame.time.delay(2000)
213             break
214
```

And then I try the if loop, which works perfectly.

```
206         if balls == 0:
207             while idle_event.key != 32: # space key
208                 while True:
209                     random_background()
210                     GameOver = 'Game Over'
211                     text = font.render('GameOver', True, (250, 250, 250))
212                     screen.blit(text, (220, 210))
213                     print ("GameOver")
214                     pygame.time.delay(1500)
215
```



The differences between if and while loop.

That is actually both a syntax and a logic error.

The program inside if statement runs **once** when the if statement is true.

The program in while statement runs **forever** when the while statement is true. I cannot let while statement run once here, because add one to the balls inside the balls < 1 statement, that will make a logic error.

So, the if statement is better here.

Thanks for Mr. Olds. He help me solve the problem.

Benefits: the game look systemically.

Version 1.8 Output high score to a document

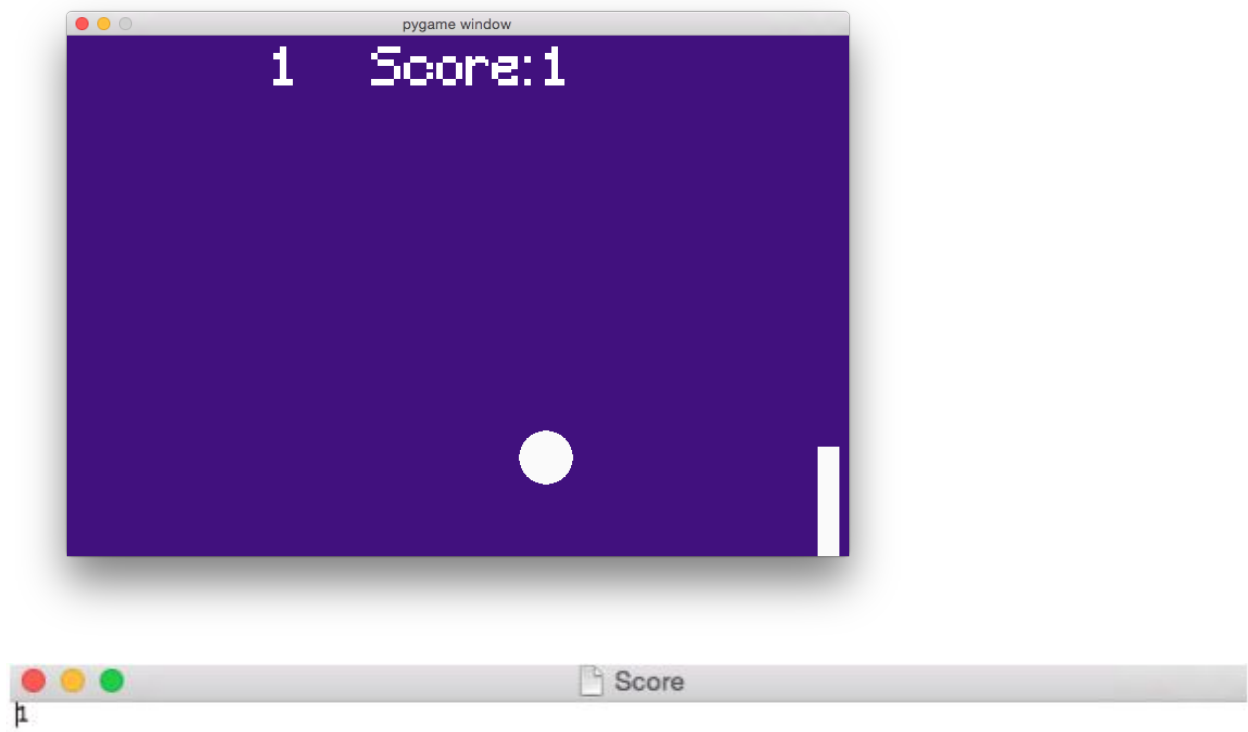
Description: In this version, it will write score into a document.

```
17 fileOut = open("Score.txt","w")
```

I identify a file to output called Score.txt.

```
209 if balls == 0:
210     screen.fill((0,0,0))
211     fileOut.write(str(bounce_time))
212     bounce_time = 0
213     pygame.time.wait(3000)
214     #GameOver = 'Game Over'
215     Game_Over = font.render("Game Over",True,(250,250,250))
216     screen.blit(Game_Over,(220,160))
217     #YourScore = 'Your Score is in Score.txt'
218     Your_Score = font.render("'Your Score is in Score.txt'",True,(250,250,250))
219     screen.blit(Your_Score,(70,250))
220
```

I write the score as a string time into the output document when the ball hits the paddle.



The score in the game is later output to the Score document.

Version 1.9

Description: I import a image I made as an introduction, which contains game name, version, author and running engine.

I import a picture in the same folder as the game folder, and delay for 3 seconds.

```
79 img = pygame.image.load('IntroV2.png')
80 screen.blit(img,(0,0))
81 pygame.time.delay(3000)
```



Benefits: It will shows a nice introduction image before instruction starts.

Version 2.0

Description: Fix bugs in the previous versions. That is our **latest stable program**. With all these functions demonstrated before, and fixed bugs like delay for no reason, unstable time for showing image.

In Version 1.9, when I show image or the game over page, it will delay for 2 or 3 seconds and then show the picture or game over for about half a second. I asked Mr. hoel, and fixed it.

I just need a simple line of code to totally refresh the screen.

It is `pygame.display.flip()`.

I previously use `screen.blit` which only display tings.

The flip function refresh the screen.

That is a logic error, or i can call it a run time error(it do not have a clear instruction to delay the picture that long).

```
77 img = pygame.image.load('IntroV2.png')
78 screen.blit(img,(0,0))
79 pygame.display.flip() #refresh
80 pygame.time.delay(3000)
```

```
211
212
213     if balls == 0: #case when player have no life anymore
214         #fill screen with black color
215         screen.fill((0,0,0))
216         #change bounce time to string, in order to write it to output
217         fileOut.write(str(bounce_time))
218         fileOut.write(" ")
219         #reset the bounce_time(score) to zero
220         bounce_time = 0
221         #show game over on screen
222         Game_Over = font.render("Game Over",True,(250,250,250))
223         screen.blit(Game_Over,(220,160))
224         #show your score is in Score.txt
225         Your_Score = font.render("'Your Score is in Score.txt'",True,(250,250,250))
226         screen.blit(Your_Score,(70,250))
227         #wait for 3 seconds
228         pygame.display.flip() #refresh
229         pygame.time.wait(3000)
230
```

Benefits: More reliable and stable.

Below it is all the code of the V 2.0. Explanations are after # on the picture.

```

1 # PONG V2.0: hit the ball with the paddle
2 # By Simon Guo & Jesse Zhao
3 # Wed, Jan 13, 2016
4 # computer science project
5
6 import pygame
7 import sys
8 from pygame.locals import *
9 from random import *
10
11 # always need to init first thing
12 pygame.init()
13
14 #set score output document
15 fileOut = open("Score.txt","w")
16
17 # create a 720p window
18 screen = pygame.display.set_mode((720,480),0,32)
19 # ask for screen's width and height
20 size = width, height = screen.get_size()
21
22 # turn on the cursor, so you can control your mouse to close tab
23 pygame.mouse.set_visible(True)
24
25 # turn on key repeating (repeat 40 times per second)
26 pygame.key.set_repeat(25, 25)
27
28 #define function random_background
29 def random_background():
30     #use RGB color
31     red = randint(0,255)
32     green = randint(0,255)
33     blue = randint(0,255)
34     bgcolor = (red,green,blue)
35     screen.fill(bgcolor)
36     return bgcolor
37
38 # paddle constants
39 paddle_width = 22
40 paddle_length = 100

```

```

41 paddle_radius = paddle_length / 2
42 paddle_color = (250, 250, 250)
43 step = 8 # paddle moves 8 pixels at a go
44
45 # ball constants
46 ball_color = (250, 250, 250)
47 ball_radius = 25
48 ball_speed_1 = 3
49 ball_speed_2 = 4
50 bounce_time = 0 # the time the ball bounce
51
52 #starting value for RGB value for the colorful screen in the game
53 Background_Red = 0
54 Background_Green = 0
55 Background_Blue = 0
56
57 #font constants
58 fsize = 48
59 font = pygame.font.Font("Minecraft.ttf", fsize)
60
61 #text information
62 Title = 'INSTRUCTION'
63 To_play = 'Press SPACE to PLAY'
64 How_play_1 = 'Move the bar by'
65 How_play_2 = 'pressing UP & DOWN'
66 How_play_3 = 'Try to hit the ball'
67 To_quit = 'Press ESC to QUIT'
68
69 INSTRUCTION = font.render(Title, True, (250, 250, 250))
70 PLAY = font.render(To_play, True, (250, 250, 250))
71 Content_1 = font.render(How_play_1, True, (250, 250, 250))
72 Content_2 = font.render(How_play_2, True, (250, 250, 250))
73 Content_3 = font.render(How_play_3, True, (250, 250, 250))
74 QUIT = font.render(To_quit, True, (250, 250, 250))
75
76 #load image into the game, display for 3 seconds
77 img = pygame.image.load('IntroV2.png')
78 screen.blit(img,(0,0))
79 pygame.display.flip() #refresh
80 pygame.time.delay(3000)

```



```

162         paddle_location + paddle_radius),
163         paddle_width)
164
165     # draw the ball
166     pygame.draw.circle(screen, ball_color, ball_position, ball_radius)
167
168     # draw the unused balls
169     for i in range(balls):
170         pygame.draw.circle(screen, ball_color,
171             (int(round(30 + i * ball_radius * 2.4)), 30),
172             ball_radius)
173
174     # update the display
175     pygame.display.flip()
176
177     #keep the screen in a colorful background until the ball hit the paddle
178     screen.fill((Background_Red,Background_Green,Background_Blue))
179
180     # update the ball
181     for i in range(2):
182         ball_position[i] = ball_position[i] + ball_mvect[i]
183
184         # bounce on top and left
185         if ball_position[i] < ball_radius:
186             ball_position[i] = ball_radius
187             ball_mvect[i] = -1 * ball_mvect[i]
188
189         # bounce on bottom
190         elif i == 1 \
191             and ball_position[i] >= ball_limit[i] - ball_radius:
192             ball_position[i] = ball_limit[i] - ball_radius - 1
193             ball_mvect[i] = -1 * ball_mvect[i]
194             #ball become faster
195             ball_speed_1 = ball_speed_1 + 1
196             ball_speed_2 = ball_speed_2 + 1
197
198         #bounce on the paddle
199         elif i == 0 \
200             and ball_position[i] >= ball_limit[i] - ball_radius - paddle_width \
201             and ball_position[1] > paddle_location - paddle_radius \
202
203             and ball_position[1] < paddle_location + paddle_radius:
204             ball_position[i] = ball_limit[i] - ball_radius - paddle_width - 1
205             ball_mvect[i] = (-1) * ball_mvect[i]
206             step = step + 0.2 # the paddle will be faster
207             bounce_time = bounce_time + 1 #bounce_time + 1, score + 1
208             #choose background color again after it hits the paddle
209             Background_Red = randint(0,254)
210             Background_Green = randint(0,254)
211             Background_Blue = randint(0,254)
212
213     if balls == 0: #case when player have no life anymore
214         #fill screen with black color
215         screen.fill((0,0,0))
216         #change bounce time to string, in order to write it to output
217         fileOut.write(str(bounce_time))
218         fileOut.write(" ")
219         #reset the bounce_time(score) to zero
220         bounce_time = 0
221         #show game over on screen
222         Game_Over = font.render("Game Over",True,(250,250,250))
223         screen.blit(Game_Over,(220,160))
224         #show your score is in Score.txt
225         Your_Score = font.render("'Your Score is in Score.txt'",True,(250,250,250))
226         screen.blit(Your_Score,(70,250))
227         #wait for 3 seconds
228         pygame.display.flip() #refresh
229         pygame.time.wait(3000)

```


Version 2.0 Beta

Description: a version with some new functions like input name and write in score.txt. Also, it will play a bounce sound when it hits the paddle.

```
80 def get_key():
81     while True:
82         event = pygame.event.poll()
83         if event.type == KEYDOWN:
84             return event.key
85         else:
86             pass
87
88 def display_box(screen, message):
89     if len(message) != 0:
90         screen.blit(font.render(message, 1, (255,255,255)),
91                     ((screen.get_width() / 2) - 300, (screen.get_height() / 2) -
92                     pygame.display.flip()
93
94 def ask(screen, question):
95     current_string = []
96     display_box(screen, question + ": " + string.join(current_string, ""))
97     while True:
98         inkey = get_key()
99         if inkey == K_BACKSPACE:
100             pygame.display.flip() #refresh
101             current_string = current_string[0:-1]
102         elif inkey == K_RETURN:
103             break
104         elif inkey == K_ESCAPE:
105             sys.exit()
106         elif inkey < 128:
107             current_string.append(chr(inkey))
108             display_box(screen, question + ": " + string.join(current_string, ""))
109     return string.join(current_string, "")
110
```

I add these code to make a input line, which you can put your name in and submit with pressing return button.

```
111 def main():
112     Name = str(ask(screen, "Name"))
113     fileOut.write(Name)
114
```

```

263 #show your score is in score.txt
264 screen.fill((0,0,0))
265
266 Your_Score = font.render("Your Score is in Score.txt",True,(250,250,250))
267 screen.blit(Your_Score,(70,300))
268 Put_Name = font.render("Please type your Name",True,(250,250,250))
269 screen.blit(Put_Name,(70,100))
270 pygame.display.flip() #refresh
271 if __name__ == '__main__': main()
272 pygame.time.wait(2000)
273
274 #change bounce time to string, in order to write it to output
275 fileOut.write(" ")
276 fileOut.write(str(bounce_time))

```

I added the main loop after showing game over.

```

78 pygame.mixer.music.load('Ball_Bounce-Popup_Pixels-172648817.wav')
79

```

I downloaded a ball bounce music from

<http://soundbible.com/1626-Ball-Bounce.html> pygame only supports .wav

I loaded it before the while loop starts.

```

pygame.mixer.music.play(-1, 0.0)

```

I added this under when the ball bounce on the paddle. I have not really test it because my laptop voice output is broken: it works on the OLPC.



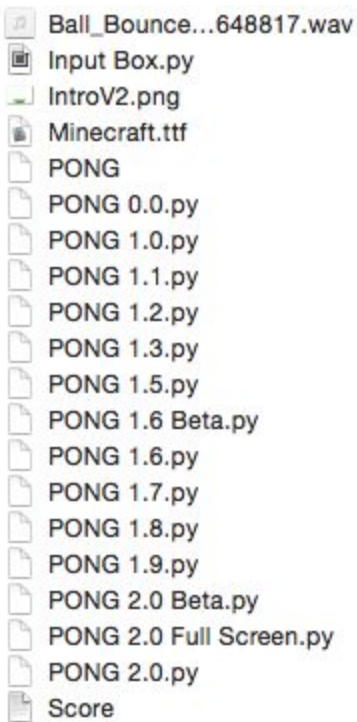
The player's name and score are shown on the screen. There is a bug that you cannot delete your input and then input it. So we did not use this version on OLPC

Variable type:

We use a lot of variables, most of them are integers, some of them are strings. We listed them in the picture of V 2.0

How we manage our programs:

We put all our versions and input, output files in one folder



Making different version made debugging really easy, and we always have something done and working.

How we used SSRGMIO and what we learned:

S: Sequential

We learned how python runs sequentially from logic errors experienced in the coding process

S: Select a path (if statements)

We learned the differences between while and if loops, and used a lot of nested conditional loops.

R: Repeat (for and while loops) for (repeat 8 times) while (repeat while a condition is true)

We learn how these loop works, and also how to break them.

G: Graphics

We learn how to draw graphics in different situation.

M: Modular prog. "Actions", functions, methods, sub prog."byte sized modules"

We learned how to make sub functions, but while using them, we have to be careful about whether the data they used is defined clearly.

I: Input

We learn how to input pictures, fonts and text and sound into a program.

O: Output

We learn how to output string to a document.

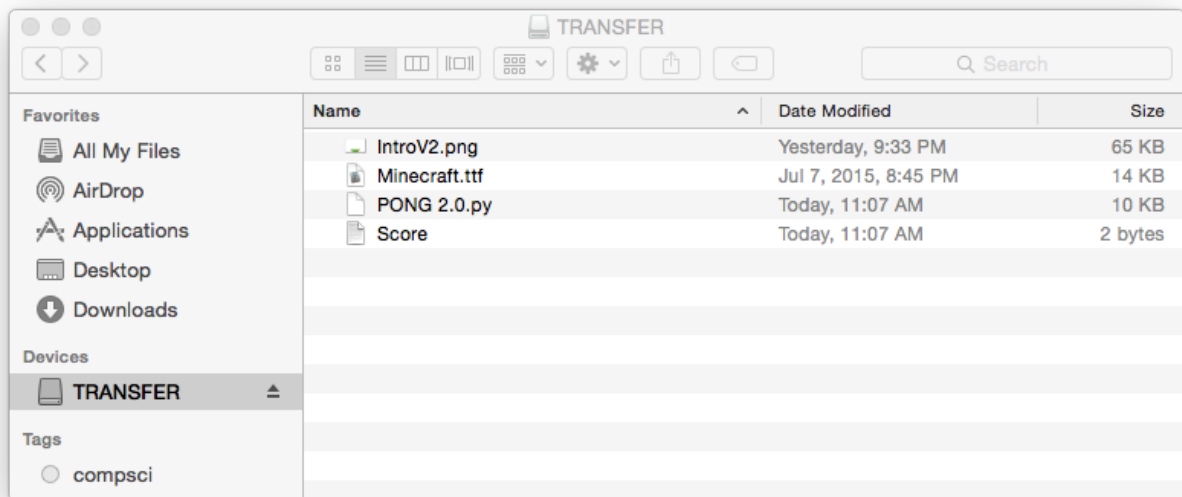
SSRGMIO was fully considered in our programming project.

How to copy file from Mac to OLPC

We need to test pong on the OLPC

1.Copy file to TRANSFER (USB DEVICE)

We need the image, the font, the score document and the version 2.0 game



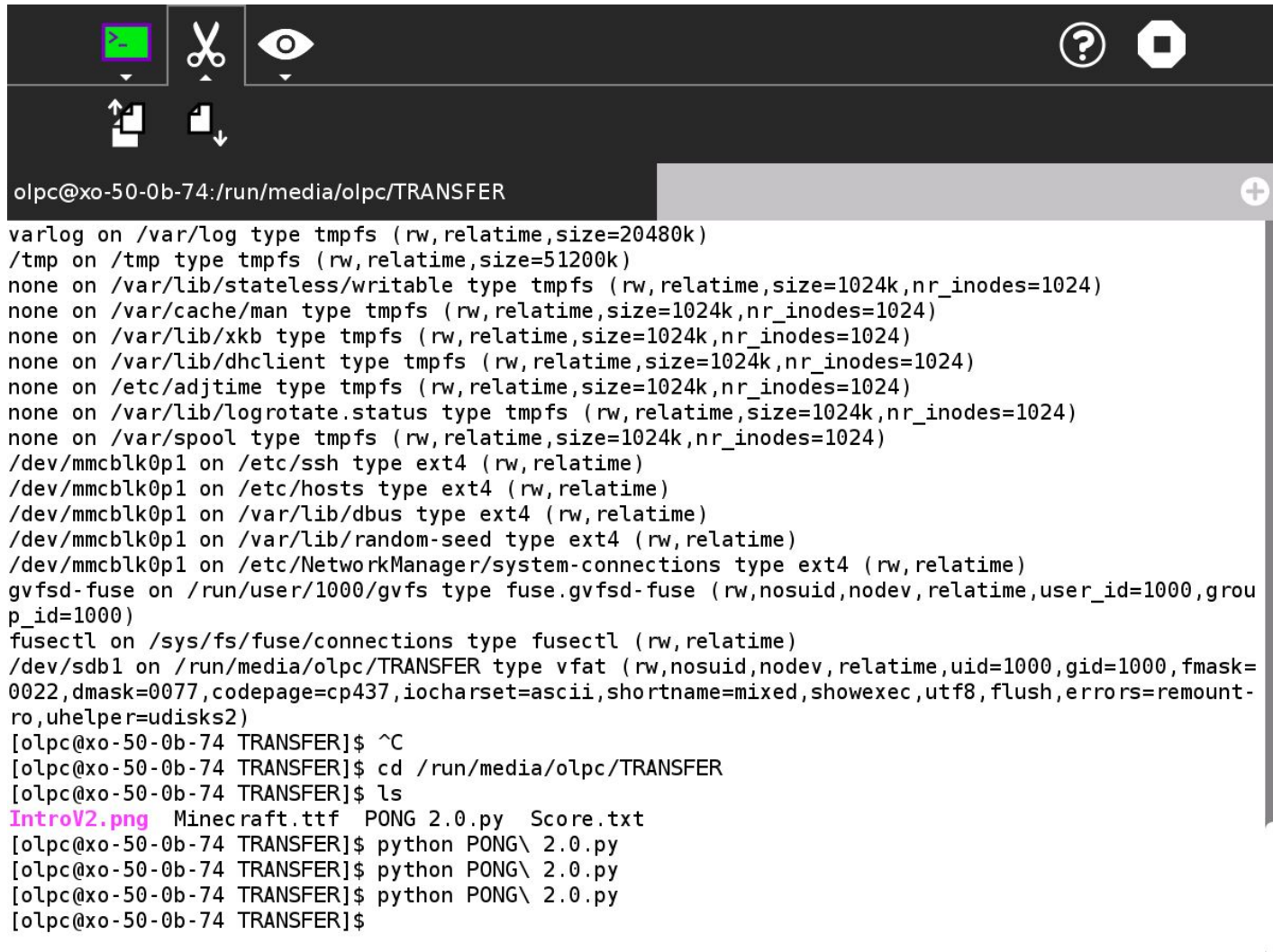
2. copy these file to OLPC

put mount command

```
olpc@xo-50-0b-74 TRANSFER:~$ mount
/dev/mmcblk0p1 on /boot type ext4 (rw,relatime)
/dev/mmcblk0p2 on / type ext4 (rw,noatime)
devtmpfs on /dev type devtmpfs (rw,nosuid,relatime,size=377536k,nr_inodes=94384,mode=755)
/dev/mmcblk0p2 on /home type ext4 (rw,relatime)
/dev/mmcblk0p2 on /versions type ext4 (rw,relatime)
/dev/mmcblk0p1 on /security type ext4 (rw,relatime)
proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
tmpfs on /dev/shm type tmpfs (rw,relatime,size=51200k)
devpts on /dev/pts type devpts (rw,relatime,gid=5,mode=620)
tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755)
tmpfs on /sys/fs/cgroup type tmpfs (rw,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,release_agent=/usr/lib/systemd/systemd-cgroups-agent,name=systemd)
cgroup on /sys/fs/cgroup/cpu type cgroup (rw,nosuid,nodev,noexec,relatime,cpu)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
squashfs on /dev/mqueue type mqueue (rw,relatime)
vartmp on /var/tmp type tmpfs (rw,relatime,size=51200k)
varlog on /var/log type tmpfs (rw,relatime,size=20480k)
/tmp on /tmp type tmpfs (rw,relatime,size=51200k)
none on /var/lib/stateless/writable type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/cache/man type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/lib/xb type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/lib/dhclient type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /etc/adjtime type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/lib/logrotate.status type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/spool type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
```

change directory to USB

cp all file to an OLPC directory, and then use python to run it.



```
olpc@xo-50-0b-74:/run/media/olpc/TRANSFER
varlog on /var/log type tmpfs (rw,relatime,size=20480k)
/tmp on /tmp type tmpfs (rw,relatime,size=51200k)
none on /var/lib/stateless/writable type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/cache/man type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/lib/xkb type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/lib/dhclient type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /etc/adjtime type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/lib/logrotate.status type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
none on /var/spool type tmpfs (rw,relatime,size=1024k,nr_inodes=1024)
/dev/mmcblk0p1 on /etc/ssh type ext4 (rw,relatime)
/dev/mmcblk0p1 on /etc/hosts type ext4 (rw,relatime)
/dev/mmcblk0p1 on /var/lib/dbus type ext4 (rw,relatime)
/dev/mmcblk0p1 on /var/lib/random-seed type ext4 (rw,relatime)
/dev/mmcblk0p1 on /etc/NetworkManager/system-connections type ext4 (rw,relatime)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,grou
p_id=1000)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
/dev/sdb1 on /run/media/olpc/TRANSFER type vfat (rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=
0022,dmask=0077,codepage=cp437,iocharset=ascii,shortname=mixed,showexec,utf8,flush,errors=remount-
ro,uhelper=udisks2)
[olpc@xo-50-0b-74 TRANSFER]$ ^C
[olpc@xo-50-0b-74 TRANSFER]$ cd /run/media/olpc/TRANSFER
[olpc@xo-50-0b-74 TRANSFER]$ ls
IntroV2.png  Minecraft.ttf  PONG 2.0.py  Score.txt
[olpc@xo-50-0b-74 TRANSFER]$ python PONG\ 2.0.py
[olpc@xo-50-0b-74 TRANSFER]$ python PONG\ 2.0.py
[olpc@xo-50-0b-74 TRANSFER]$ python PONG\ 2.0.py
[olpc@xo-50-0b-74 TRANSFER]$
```

here is a link to the video of it.

https://drive.google.com/open?id=0B_MW2YLgRt45NnIPaFd6MFRBaGs

How Pong run on olpc

Contributions to this project:

Simon Guo

I did most of the programming and debugging. I keep update the version of my pong. I give what I change and different version to my partner, so he can do the documentation. Also, I am doing some graphing design for our pong and document.

Jesse Zhao

I did most of the tutorial/documentation including taking some pictures. I designed most of the google document and explained some of simon's modifications as well as the other requirements on the document. I also gave a lot of ideas as to what modifications our program should have.

References:

1. <http://pygame.org/docs/ref/display.html#pygame.display.set+mode>

This is list of Pygame functions and describes how to use them and what they are used for.

2. <http://trevorappleton.blogspot.ca/2014/04/writing-pong-using-python-and-pygame.html>

Tutorials, Hints and Tips about the Raspberry Pi and Python.

3. <http://www.theasciicode.com.ar/>

ASCII code table.

4. <http://pygame.org/docs/tut/intro/intro.html>

An introduction about how to load an image into pygame.

5. <http://pygame.org/hifi.html>

Pygame news.

6. <http://www.xquartz.org/>

Information about xquartz and has the download link for xquartz on the website

7. <http://soundbible.com/1626-Ball-Bounce.html>

A website that has many different ball sounds that can be listened and downloaded. We implemented some in our project.

Thanks!