Lab13: Pygame library

In this lab we will use Pygame, which is a Python library used specifically to make games.

So far in this class, we have used Graphics Library by John M. Zelle. Graphics library is a “easy-to-use library to introduce object-oriented concepts”. It other words, it is designed to teach people how to program. It is not designed to actually program games.

In the next Project and in this Lab, we will use pygame. Pygame is a library for Python, just like graphics and matplotlib. Pygame is particularly designed to make games. Here are some cool projects written with pygame: http://www.pygame.org/hifi.html#awesome.

A very useful pygame introduction: https://www.cs.ucsb.edu/~pconrad/cs5nm/topics/pygame/drawing/

Here is a video series that might be useful: https://www.youtube.com/playlist?list=PL6gx4Cwl9DGAjkwJocj7vlc_mFU-4wXjQ.

- Make sure you understand this lab well, because it will help you with the upcoming Project 4.

Prelab 13

We encourage you to work together on the Pre Lab. The Pre Lab is not graded but will help you prepare for your lab session. If you have any questions on the material of the Pre lab, first check the book and recitation slides. If you continue to have any doubt about it, please email your recitation TA or the course instructor.

This prelab covers some basic usage of pygame library and introduces some simple operations.

Here is PreLab13

Environment Setup

Go to your working directory in “data.cs.purdue.edu” and create a directory “cs177/lab13”. Refer to the first lab if you need to remember the steps to do so. Then start the IDLE Python Interpreter.

To test if pygame works in your machine, you can run this code:

```python
import pygame
import sys
```
pygame.init()
window = pygame.display.set_mode((600,600))

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

    window.fill((0,0,0))  # black background
    pygame.draw.rect(window, (255,255,255), (250,250,100,100))  # white square
    pygame.display.update()

pygame.quit()

Or similarly, https://www.pygame.org/docs/ref/examples.html

If the above runs without any errors, you should be ready to proceed.

If not, go to the PreLab13 and install Matplotlib on your machine.

**Task 1: Collision detection using pygame**

Make sure you finished the PreLab13 before you start this task.

We will do collision detection using pygame in this task. Here are the details:

1. The window is 800×600.
2. We will have two squares.
3. Big square is blue, has width = 200, height = 200, and it is always perfectly centered on the window.
4. Small square has width = 100, height = 100.
5. Small square is white when there is no collision with big square, and is red when there is collision.
6. Small square will start in the top-left corner.
7. We will use arrow keys to navigate the small square.

Here is the annotated visualization of the initial position of the program:
Your final program should look like: https://youtu.be/eq9O1r2wITM

Save the program as collision.py in your lab13 directory. Make sure that it runs correctly.

**Task 2: Move 3x3 square matrix**

Make sure you finished the PreLab13 before you start this task.

This task is very similar to the last task in the PreLab. Here are the details:

1. The window size is 500×500
2. There will be 3×3 = 9 squares in the window
3. Each square will be of size 20×20 and the distance between them will be 20.
4. The initial coordinate of the square-matrix will be the center of the screen.

Here is the annotated initial positioning of the rectangles:
Note that here are the left-top corners of the squares in the beginning, however, you cannot hardcopy these values. You should generate them using a nested for or while loop.

```
[[200, 200], [200, 240], [200, 280],
[240, 200], [240, 240], [240, 280],
[280, 200], [280, 240], [280, 280]]
```

Your output should look like: https://youtu.be/BW0N2F9g57g

Save the program as `move_matrix.py` in your lab13 directory. Make sure that it runs correctly.

**Task 3: Draw grid**

We draw grid using lines in this task. We can draw lines on the screen using the following command in Pygame:
import pygame
import sys

# some initializations
pygame.init()
window = pygame.display.set_mode((500, 500))
pygame.display.set_caption('Grid')
clock = pygame.time.Clock()

# color definitions, see RGB color model if this does not make sense:
# https://en.wikipedia.org/wiki/RGB_color_model
black = (0, 0, 0)
white = (255, 255, 255)

m = 3  # number of rows
n = 4  # number of columns

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
sysexit()

    # draw and update screen
    window.fill(black)

    # Your code starts here...
    # ------------------------
    # ------------------------

    # ------------------------
    # Your code ends here

    pygame.display.update()

    # fps stuff:
    clock.tick(10)

pygame.quit

Here are the details:

1. The window size is 500×500
2. You need to use two separate loops (not-nested) which calls pygame.draw.line() function
3. m is number of rows and n is number of columns
4. Your program should work when we change m or n in the code.

Documentation for draw.line() function:
http://www.pygame.org/docs/ref/draw.html#pygame.draw.line
Example: `pygame.draw.line( window, (255, 255, 255), (0, 0), (100, 100) )` will draw a line to the window from (0,0) to (100,100) with white color.

Here are some examples.

m=4, n=4:

![Grid Image](image)

m=3, n=5:
m=2, n=6:
m=1, n=5:
Save the program as `grid.py` in your lab13 directory. Make sure that it runs correctly.

**Turnin Instructions**

Run putty and login to `data.cs.purdue.edu`. Turn in your lab by typing:

```
$ cd cs177
$ turnin -v -c cs177=COMMON -p lab13 lab13
```

**Grading Criteria**

<table>
<thead>
<tr>
<th>Task</th>
<th>Percentage (%)</th>
<th>Breakdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>30%</td>
<td>Plotting the initial objects using pygame(10%) Using arrow keys to move the small square correctly(10%) Changing the small square's color when a collision happens(10%)</td>
</tr>
<tr>
<td>Task</td>
<td>Percentage (%)</td>
<td>Breakdown</td>
</tr>
<tr>
<td>------</td>
<td>----------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Task 2</td>
<td>30%</td>
<td>Plotting the initial objects using pygame (15%) Using arrow keys to move all squares synchronously (15%)</td>
</tr>
<tr>
<td>Task 3</td>
<td>40%</td>
<td>Plots the grid correctly when $m=3$, $n=4$ (20%), Plots correctly for all $m,n$ values.(20%)</td>
</tr>
</tbody>
</table>

From: [http://courses.cs.purdue.edu/](http://courses.cs.purdue.edu/) - Computer Science Courses


Last update: **2016/04/02 17:31**