Lab 11: Pygame

- Please check the Prelab11, it will be helpful.
- Make sure you understand this lab well, because it will help you with the upcoming project.

In this lab we will use Pygame, which is a Python library used specifically to make games. First thing to do is to install Pygame to you machine.

Pygame installation

Please check the prelab for installation steps.

Task1

In this task you are required to use Pygame to create a very simple game. (It's recommended to check the pre-lab.) You will implement a program using Pygame similar to the first example in the pre-lab, you will move a rectangle around on the screen using the arrow keys in the keyboard. The user should be able to move your rectangle continuously on the screen without having to keep pressing a certain direction, for example when you press the down arrow, the rectangle should keep moving down even when you stop pressing the down key, it's just one press and the rectangle should be moving contentiously in that direction.

Use this template for this task:

```python
import pygame

pygame.init()
window = pygame.display.set_mode((800,600))
pygame.display.set_caption('Rectangle continuous move')
clock = pygame.time.Clock()
black = (0,0,0)
white = (255,255,255)

#TODO

pygame.quit()
```
Task2

In this task you are required to use Pygame to create a very simple game. (It's recommended to check the pre-lab.) You will implement a program using Pygame similar to the second example in the pre-lab. Your job is to draw a formation of squares on the screen and to use the keyboard to move this formation of squares contentiously like task1 such that the relative positions and distances between the squares are preserved during the motion. In other words, your keyboard buttons should move the whole formation of squares together continuously as a single unit.

Use this template for this task:

```python
import pygame

WIDTH = 600  # width and height always the same for simplicity
n = 3  # the number of squares on the same line.

# some initializations
pygame.init()
window = pygame.display.set_mode((WIDTH, WIDTH))
pygame.display.set_caption('Rectangle matrix continuous move')

clock = pygame.time.Clock()
black = (0,0,0)
white = (255,255,255)
length = WIDTH / (2 * n + 1)  # the side length of the square, which equals to the length of the margin

# TODO

pygame.quit()
```

Task3

This task is a tutorial task (not graded). Your job here is to implement collision detection in Pygame. Collision detection in games is the technique used to detect if two entities collide with each other. For example, recall project2, bouncing the ball off bob or off the windows boundaries is collision detection, also preventing bob from crossing the windows boundaries is using collision detection. In this task we will draw two rectangles, the first one is controlled by the keyboard and the second rectangle is fixed. Whenever the moving rectangle collides with the fixed rectangle, you should change the color of the moving rectangle to red. The moving rectangle should stay red as long as it's in collision with the fixed rectangle, and it should change its color back to white when there is no collision anymore.

Here is a template for this task:

```python
import pygame

pygame.init()
window = pygame.display.set_mode((800,600))
```
pygame.display.set_caption('Collision detection using Pygame')
clock = pygame.time.Clock()

black = (0,0,0)
white = (255,255,255)
red = (255,0,0)
blue = (0,0,255) #color of the fixed rectangle

# initial center position for two squares
# we have a small square and a big square
small_x, small_y = 50, 50
big_x, big_y = 400, 300

game_loop = True
while game_loop:
    # moving the white rectangle
    # check collision
    # if collision detected, change color of the white rectangle to red
    # draw and update screen
	pygame.quit()

---

**Turnin Instructions**

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

```
$ cd cs177
$ turnin -c cs177=COMMON -p lab11 lab11
```

To verify that you did not submit a wrong file or an empty one, run the following command:

```
$ turnin -v -p lab11
```

**Grading Criteria**

<table>
<thead>
<tr>
<th>Task</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>50%</td>
</tr>
<tr>
<td>Task 2</td>
<td>50%</td>
</tr>
<tr>
<td>Task 3</td>
<td>Not graded</td>
</tr>
</tbody>
</table>

---

From:
http://courses.cs.purdue.edu/ - Computer Science Courses

Permanent link:
http://courses.cs.purdue.edu/cs17700:fall15:labs:lab11

Last update: 2015/11/02 23:10