OSL MINI-PROJECT

“TYPING SPEED TEST”

CLASS: S.E (SEM-4)

DEPARTMENT: COMPUTER

DIV: A BATCH: 02

SUBJECT: OPEN SOURCE LABORATORY (OSL)

COLLEGE NAME: RGIT

SUBJECT HEAD: PROF.DILIP DALGADE

|  |  |  |
| --- | --- | --- |
| TEAM MEMBERS | DIV/BATCH | ROLL NO |
| 1)SAKET DHAMAPURKAR | A-2 | 426 |
| 2)UJJWAL DIXIT | A-2 | 429 |

INDEX

|  |  |  |
| --- | --- | --- |
| SR.NO | TOPIC | PAGE NO. |
| 1 | ABSTRACT |  |
| 2 | WORKING |  |
| 3 | CODE |  |
| 4 | OUTPUT |  |
| 5 | REFRENCE |  |

ABSTRACT

Have you played a typing speed game? It’s a very useful game to track your typing speed and improve it with regular practice. Now, you will be able to find your own typing speed using this game built Python.

In this Python project idea, we are going to build an exciting project through which you can **check** and even **improve** your typing speed. For a graphical user interface, we are going to use the **pygame** library which is used for working with graphics. We will draw the images and text to be displayed on the screen.

WORKING

1. Import the libraries

For this project based on Python, we are using the pygame library. So we need to import the library along with some built-in modules of Python like time and random library.

1. import pygamefrom pygame.locals import \*
2. import sys
3. import time
4. import random

### 2. Create the game class

Now we create the game class which will involve many functions responsible for starting the game, reset the game and few helper functions to perform calculations that are required for our project in Python.

### 3. draw text () method

The draw text () method of Game class is a **helper function** that will draw the text on the screen. The argument it takes is the screen, the message we want to draw, the y coordinate of the screen to position our text, the size of the font and colour of the font. We will draw everything in the centre of the screen. After drawing anything on the screen, pygame requires you to update the screen.

### 4. get sentence() method

Remember that we have a list of sentences in our sentences.txt file? The get sentence () method will open up the file and **return a random sentence** from the list. We split the whole string with a newline character.

### 5. show results() method

The show results () method is where we **calculate the speed** of the user’s typing. The time starts when the user clicks on the input box and when the user hits return key “Enter” then we perform the difference and calculate time in seconds.

To calculate accuracy, we did a little bit of math. We counted the correct typed characters by comparing input text with the display text which the user had to type.

**The formula for accuracy is:**

**Correct characters x 100**

**Total characters**

The WPM is the words per minute. A typical word consists of around 5 characters, so we calculate the words per minute by dividing the total number of words with five and then the result is again divided that with the total time it took in minutes. Since our total time was in seconds, we had to convert it into minutes by dividing total time with 60.

At last, we have drawn the typing icon image at the bottom of the screen which we will use as a reset button. When the user clicks it, our game would reset. We will see the reset game () method later in this article.

### 6. run () method

This is the main method of our class that will**handle all the events**. We call the reset game() method at the starting of this method which resets all the variables. Next, we run an infinite loop which will capture all the mouse and keyboard events. Then, we draw the heading and the input box on the screen.

We then use another loop that will look for the mouse and keyboard events. When the mouse button is pressed, we check the position of the mouse if it is on the input box then we start the time and set the active to True. If it is on the reset button, then we reset the game.

When the active is true and typing has not ended then we look for keyboard events. If the user presses any key then we need to update the message on our input box. The enter key will end typing and we will calculate the scores to display it. Another event of a backspace is used to trim the input text by removing the last character.

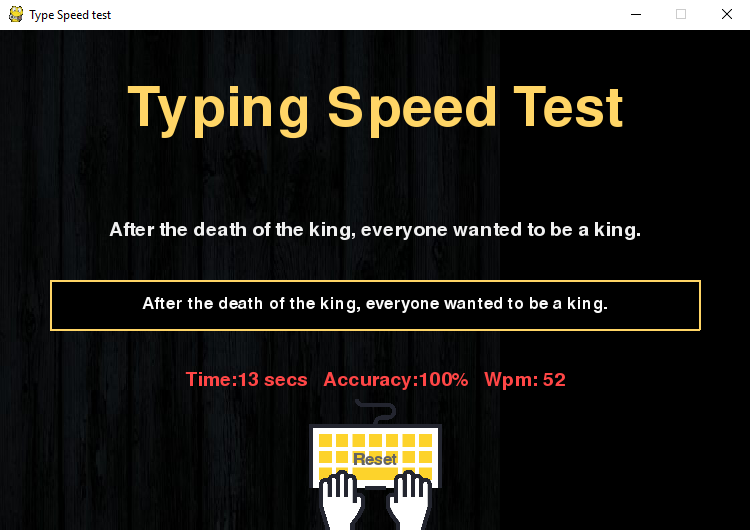
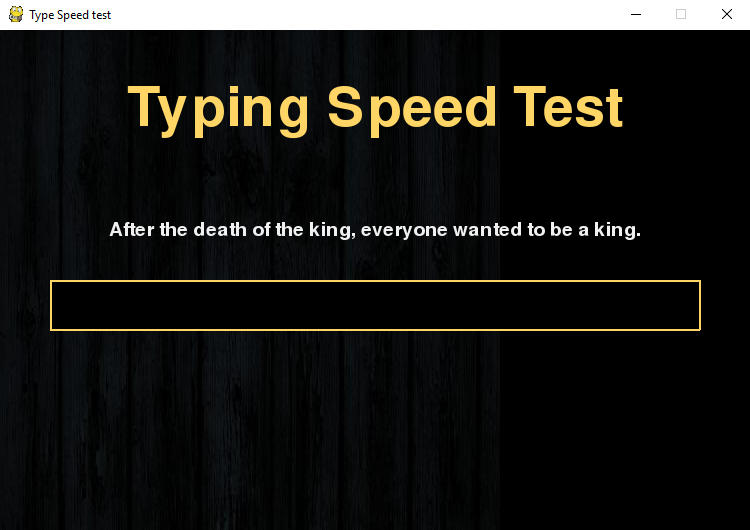
### 7. reset game() method

The reset game() method **resets all variables** so that we can start testing our typing speed again. We also select a random sentence by calling the get sentence() method. In the end, we have closed the class definition and created the object of Game class to run the program.

Code

import pygame  
from pygame.locals import \*  
import sys  
import time  
import random  
  
  
# 750 x 500  
  
class Game:  
  
 def \_\_init\_\_(self):  
 self.w = 750  
 self.h = 500  
 self.reset = True  
 self.active = False  
 self.input\_text = ''  
 self.word = ''  
 self.time\_start = 0  
 self.total\_time = 0  
 self.accuracy = '0%'  
 self.results = 'Time:0 Accuracy:0 % Wpm:0 '  
 self.wpm = 0  
 self.end = False  
 self.HEAD\_C = (255, 213, 102)  
 self.TEXT\_C = (240, 240, 240)  
 self.RESULT\_C = (255, 70, 70)  
  
 pygame.init()  
 self.open\_img = pygame.image.load('type-speed-open.png')  
 self.open\_img = pygame.transform.scale(self.open\_img, (self.w, self.h))  
  
 self.bg = pygame.image.load('background.jpg')  
 self.bg = pygame.transform.scale(self.bg, (500, 750))  
  
 self.screen = pygame.display.set\_mode((self.w, self.h))  
 pygame.display.set\_caption('Type Speed test')  
  
 def draw\_text(self, screen, msg, y, fsize, color):  
 font = pygame.font.Font(None, fsize)  
 text = font.render(msg, 1, color)  
 text\_rect = text.get\_rect(center=(self.w / 2, y))  
 screen.blit(text, text\_rect)  
 pygame.display.update()  
  
 def get\_sentence(self):  
 f = open('sentences.txt').read()  
 sentences = f.split('\n')  
 sentence = random.choice(sentences)  
 return sentence  
  
 def show\_results(self, screen):  
 if (not self.end):  
 # Calculate time  
 self.total\_time = time.time() - self.time\_start  
  
 # Calculate accuracy  
 count = 0  
 for i, c in enumerate(self.word):  
 try:  
 if self.input\_text[i] == c:  
 count += 1  
 except:  
 pass  
 self.accuracy = count / len(self.word) \* 100  
  
 # Calculate words per minute  
 self.wpm = len(self.input\_text) \* 60 / (5 \* self.total\_time)  
 self.end = True  
 print(self.total\_time)  
  
 self.results = 'Time:' + str(round(self.total\_time)) + " secs Accuracy:" + str(  
 round(self.accuracy)) + "%" + ' Wpm: ' + str(round(self.wpm))  
  
 # draw icon image  
 self.time\_img = pygame.image.load('icon.png')  
 self.time\_img = pygame.transform.scale(self.time\_img, (150, 150))  
 # screen.blit(self.time\_img, (80,320))  
 screen.blit(self.time\_img, (self.w / 2 - 75, self.h - 140))  
 self.draw\_text(screen, "Reset", self.h - 70, 26, (100, 100, 100))  
  
 print(self.results)  
 pygame.display.update()  
  
 def run(self):  
 self.reset\_game()  
  
 self.running = True  
 while (self.running):  
 clock = pygame.time.Clock()  
 self.screen.fill((0, 0, 0), (50, 250, 650, 50))  
 pygame.draw.rect(self.screen, self.HEAD\_C, (50, 250, 650, 50), 2)  
 # update the text of user input  
 self.draw\_text(self.screen, self.input\_text, 274, 26, (250, 250, 250))  
 pygame.display.update()  
 for event in pygame.event.get():  
 if event.type == QUIT:  
 self.running = False  
 sys.exit()  
 elif event.type == pygame.MOUSEBUTTONUP:  
 x, y = pygame.mouse.get\_pos()  
 # position of input box  
 if (x >= 50 and x <= 650 and y >= 250 and y <= 300):  
 self.active = True  
 self.input\_text = ''  
 self.time\_start = time.time()  
 # position of reset box  
 if (x >= 310 and x <= 510 and y >= 390 and self.end):  
 self.reset\_game()  
 x, y = pygame.mouse.get\_pos()  
  
  
 elif event.type == pygame.KEYDOWN:  
 if self.active and not self.end:  
 if event.key == pygame.K\_RETURN:  
 print(self.input\_text)  
 self.show\_results(self.screen)  
 print(self.results)  
 self.draw\_text(self.screen, self.results, 350, 28, self.RESULT\_C)  
 self.end = True  
  
 elif event.key == pygame.K\_BACKSPACE:  
 self.input\_text = self.input\_text[:-1]  
 else:  
 try:  
 self.input\_text += event.unicode  
 except:  
 pass  
  
 pygame.display.update()  
  
 clock.tick(60)  
  
 def reset\_game(self):  
 self.screen.blit(self.open\_img, (0, 0))  
  
 pygame.display.update()  
 time.sleep(1)  
  
 self.reset = False  
 self.end = False  
  
 self.input\_text = ''  
 self.word = ''  
 self.time\_start = 0  
 self.total\_time = 0  
 self.wpm = 0  
  
 # Get random sentence  
 self.word = self.get\_sentence()  
 if (not self.word): self.reset\_game()  
 # drawing heading  
 self.screen.fill((0, 0, 0))  
 self.screen.blit(self.bg, (0, 0))  
 msg = "Typing Speed Test"  
 self.draw\_text(self.screen, msg, 80, 80, self.HEAD\_C)  
 # draw the rectangle for input box  
 pygame.draw.rect(self.screen, (255, 192, 25), (50, 250, 650, 50), 2)  
  
 # draw the sentence string  
 self.draw\_text(self.screen, self.word, 200, 28, self.TEXT\_C)  
  
 pygame.display.update()  
  
  
Game().run()

Output



Reference

1)[www.geeksforgeeks.com](http://www.geeksforgeeks.com)

2)[www.stackoverflow.com](http://www.stackoverflow.com)

3)[www.github.com](http://www.github.com)