**Display Text and Count Score in PyGame**

In PyGame, there is a built-in function used to display text on the screen. We can simply develop our own functions (as shown below) which include that built-in function. For example,

|  |
| --- |
| def text\_objects(text, font): # Our new function to display text  textSurface = font.render(text, True, RED) # Define fonts color  return textSurface, textSurface.get\_rect() # Define a rectangular surface to # display font |

|  |
| --- |
| def message\_display(text):  largeText = pygame.font.Font('freesansbold.ttf',115) # Define font type and size  TextSurf, TextRect = text\_objects(text, largeText) # Create text as object  TextRect.center = ((displayWidth/2),(displayHeight/2)) # Define where texts will appear  gameDisplay.blit(TextSurf, TextRect) # Display text  pygame.display.update() # Refresh the screen  time.sleep(2) # Pause for 2 seconds |

You need to put the above two functions in your Python code. To call these two functions, simply call

message\_display(‘Welcome)

message\_display(‘Game Over’)

**1) Write a code to display a text “Welcome” before the start of the game. You can, at this time, put the code in gameLoop( ). Also write a code to display a text “Game Over” when objects collision is detected.**

As your code grows bigger, it becomes more manageable to develop functions to handle game introduction and game ending (or game pausing). Hence, the following two functions are developed to call message\_display( )

|  |
| --- |
| def crash():  message\_display(‘You Crashed!!!’)  def game\_intro():  message\_display(‘Welcome’) |

**2) Replace the codes you have written in 1) with the above two functions.**

In the case that we want to display a text that is active (text is changed when something is happening). For example, when we want to show a score, level, life, etc., we can write a simple function such as

|  |
| --- |
| def scoreCount(count):  font = pygame.font.SysFont(None, 25) # Font type = None, size = 25  text = font.render("Score: "+str(count), True, BLACK) # Create text from “Score”+ count  gameDisplay.blit(text,(0,0)) # Display text at 0,0 |

**3) To ensure scoreCount(count) works correctly, you need to modify fallingFruit( ) to take one more input argument (which is variable *count*) and return that variable out. In the if-else condition to detect collision (in fallingFruit( )), a value of *count* is increased by 1. After that (outside if-else condition), scoreCount(count) is called to display score(s) on the screen. Note that since variable *count* needs to be use inside and outside many functions, it has to be defined before use. Hence in gameLoop( ), we need to define this variable and initialize its value to 0. For example, count = 0.**

**4) Write a new function, namely def levelCount(level), to display the level with a condition that the level is increased by 1 every 10 points.**

**5) Write a code to have a new falling durian (with different color or size). If the new falling durian is collected, the score is increased by 3.**

**Hint: You may write a new function, namely fallingFruit2( ) to handle the new Durian. New image must be loaded for this function. Make sure that you change the variable name that contains an image. After that in gameLoop( ), you need to call this new function similar to what you did when the fallingFruit( ) is called. , for example**

If you have successfully completed the above instructions, you should see two different fruits falling in the game. However if you want these two fruits to alternately drop (after it is collected), you need to manage this part by:

1. **You need a new variable to determine the state of the fruits. Let’s define this variable’s name as *flag*. When the collision is detected (in both fallingFruit( ) and fallingFruit2( )), the value of *flag* is changed alternately (between 0 and 1).**
2. **The change of *flag*’s value must be changed from 1 to 0 in one function and from 0 to 1 in another function.**
3. **Variable *flag* must be passed into the functions and be returned from the functions.**
4. **Variable *flag* must be defined and initialized in gameLoop( ) (same as you do with variable *count*).**
5. **To make it work, the following part of the code needs to be slightly modified.**

|  |
| --- |
| if not gameExit:  if flag == 0: # for fallingFruit( )  # ==== Call fallingFruit( ) here ====#    if not gameExit: # for fallingFruit2( )  if flag == 1:  #==== call fallingFruit2( ) here ====# |

If you have successfully managed to complete the above codes, especially in gameLoop( ), you will notice that fruits are alternately dropped if you can collect one or the other.