# Bubble Sort Visualizer using PyGame - ProjectGurukul

#### **About Bubble Sort Visualizer:**

Consider a telephone directory containing phone numbers in a random order. Since the listing is random, finding specific information is going to be time consuming and laborious. The same task can be easy to perform when the directory is alphabetically ordered. This is where Sorting comes to play. Sorting is a technique that aids searching for necessary information by ordering elements of consideration in increasing order of numbers, or alphabets.

#### **Python Bubble Sort Visualizer:**

The objective of this project is to visualize bubble sort using pygame.

#### **Project Prerequisites:**

The project uses pygame, a built-in module in python. It is helpful to understand loops and function calls. Since it is built-in, it is good to check if the module is installed using the following commands

import pygame

pygame.\_\_version\_\_

This command displays the pygame version. In case the module is missing, an error will pop up. Thus install pygame using pip3: pip3 install pygame

#### **Download Bubble Sort Visualizer code:**

You can download the source code for the Bubble Sort Visualizer in the given link: [Bubble Sort Using Pygame](https://drive.google.com/file/d/1YDmWhbmYo18FPX2le6NV6vUWPdcaZogi/view?usp=sharing)

#### **Project File Structure:**

Let’s have a look at the steps to build the project:

1. Importing necessary modules: pygame
2. Initialising pygame and creating a window with declaration of necessary variables
3. Defining functions to draw rectangles, show texts and implement bubble sort
4. Taking user input and visualization of bubble sort

Now for the implementation in detail.

Feel free to play with the values and change your input methods.

##### **Importing necessary modules:**

#ProjectGurukul's Guide for Visualization Of Bubble Sort

#Import necessary modules

import pygame

**Code Explanation:**

* **Import pygame:** Pygame is an open source library to create games and do other multimedia tasks. It provides functions to create windows, shapes, read keyboard and mouse events using which we can do functions or tasks.

##### **Initialising pygame and creating a window with declaration of necessary variables:**

#Initialise pygame window

pygame.init()

array1 = ""

array =[]

#Define screen dimensions, declare font for viewing text

screen = pygame.display.set\_mode((700,500))

font = pygame.font.SysFont('ubuntu mono', 20)

run = True

**Code explanation:**

* **pygame.init():**  Initialise pygame to create a window for viewing bubble sort
* **array=[], array1=”:** Declare a list, array, to contain the user input. Similarly array1 is an empty string
* **pygame.display.set\_mode((500,500)):**  Create a window with the dimensions(500,500)
* **pygame.font.Font(None, 25):** Initialise a font variable of the font type None, with size 25. Here None implies the default system font. If the ttf file for a font is present, it can substitute None. Alternatively, to provide with a predefined font, use the function **SysFont(font\_name, size)**
* **run:** A variable to terminate the loop and close the window

##### **Defining functions to draw rectangles, show texts and implement bubble sort:**

#Function to show text

def show\_text(array):

#Create a new screen

screen.fill((0,0,150,0))

#Use the font to display the array

block = font.render(str(array), True, (255,255,150))

#Display the array

screen.blit(block, (20,20))

#Function to draw rectangles

def draw\_rect():

for i in range(len(array)):

#Draw rectangles using array elements

#To maintain gaps between rectangles, mention the top coordinate more than the width

pygame.draw.rect(screen, (255, 125, 0),((50+i\*25,50, 20, array[i]\*2)))

pygame.display.update()

#Function to implement bubble sort

def bubble\_sort():

for i in range(len(array)):

for j in range(len(array)-1):

#Compare every element with every other element and switch places

if array[j] > array[j+1]:

array[j], array[j+1] = array[j+1], array[j]

#Display the array

array1 = [str(i) for i in array]

array1=",".join(array1)

show\_text(array1)

#Draw the rectangular boxes

draw\_rect()

#Keep a delay between changes

pygame.time.delay(500)

#Display the changes made

pygame.display.update()

**Code explanation:**

* **show\_text:** To receive user input, the user must be able to see what he types. Hence this function displays or renders the array on a screen. Fill the screen with color in the format (r,g,b,a) or by typing the color name. Next using the initialised font, we prepare to render it using render(text\_to\_display, True, (color)), following which screen.blit(rendered\_text, (position)) displays the text
* **draw\_rect():** This function draws rectangles on the screen. The length of rectangles correspond to the values in the user input. To maintain distance between the rectangles, a distance greater than the width is set for the top left coordinate. Here we add 5 as the distance between rectangles. To reflect the changes on the screen, call pygame.display.update()
* **bubble\_sort():** Implement bubble sort here. The array elements are compared with one another and swapped if the former element is greater than the latter. After sorting, convert the array to a string of numbers and join them using ‘,’.join(array1). show\_text() is called to display the array changes. To display the rectangles, call draw\_rect() and set a delay between changes using pygame.time.delay(500). Then view the changes using pygame.display.update()

##### **Taking user input and visualization of bubble sort:**

#Since changes keep happening, a loop is used

while run == True:

#Detect keyboard press

for event in pygame.event.get():

#Keyboard press condition

if event.type == pygame.KEYDOWN:

#Spacebar press

if event.key == pygame.K\_RETURN:

#Start visualising and sorting

#Convert string to array by splitting the string

array = array1.split(",")

array = [int(i) for i in array]

draw\_rect()

pygame.time.delay(3000)

bubble\_sort()

elif event.key == pygame.K\_BACKSPACE:

#Remove last element in case of any changes

array1 = array1[:-1]

else:

#Check if the keyboard press is a digit

array1+=event.unicode

show\_text(array1)

pygame.display.update()

#Check if pygame exit is selected

elif event.type == pygame.QUIT:

run= False

#Quit and close the window

pygame.quit()

**Code explanation:**

* **Run == True:** To continue viewing the visualization, run has to be set to True.
* **for event in pygame.event.get():** Detect user events such as mouse clicks, mouse movements, keyboard key clicks
* **event.type == pygame.KEYDOWN:** To check if the event is a keyboard key press.
* **event.key == pygame.K\_RETURN:** Checks if enter is pressed. The array (string) is split to a list using split() and converted to integer elements using int() and then sorting starts after displaying rectangles initially following which a delay happens.
* **event.key == pygame.K\_BACKSPACE:** To delete the last element in the list using array1[:-1]. To remove the last element, index is set to -1.
* **else condition:** The user input is taken here. Append the user input to the string array1. Call show\_text(array1) to display it to the user
* **event.type == pygame.QUIT:** If the user selects the close window option, the flag run is set to False and the loop terminates.
* **pygame.quit():** To close the window and quit the initialised pygame, pygame.quit() is used.

##### **Extras:**

#Initially fill the screen black

screen.fill((0,0,0,0))

block = font.render("ProjectGurukul's Guide for Visualization Of Bubble Sort", True, (255,0,150))

screen.blit(block, (0,20))

block1 = font.render("Enter Input and press ENTER to visualize", True, (255,255,150))

screen.blit(block1, (0,40))

block2 = font.render("Add comma to separate the integers and backspace to pop", True, (255,255,150))

screen.blit(block2, (0,60))

pygame.display.update()

**Code explanation:**

This snippet of code is optional

* **screen.fill((0,0,0,0)):** Fill the screen black whilst displaying initial texts

#### **Project output:**

Enter the inputs and view the output:



#### **Summary**

Thus using pygame, we created a simple Bubble Sort Visualizer. The project proves to be a great introduction and a fun opportunity to play with values and shapes creating amazing visualisers.