tkinter Tutorial for Designing Graphical User Interface in Data Visualization Tools like ComVisMD

Shridhar B. Dandin*, Megha Sinha

Department of Computer Science Engineering
Sarala Birla University, Ranchi, India

Abstract

An end user always intends that every application should be easy-to-handle, attractive and self-driven to offer its functionalities. The user interface decides the quality of an application, its sustainability as well as its productivity. Today's trend and need is to develop each and every application with easily understandable Graphical User Interface (GUI) to enhance the user interactivity, and directly or indirectly leading to visual analysis of data. A cross-platform framework like *tkinter* of Python's standard library, eases the job of programmer in designing GUI for many data visualization tools like ComVisMD (Compact Visualization of Multidimensional Data). The *tkinter* framework supports multiple platforms and provides many easy-to-use, quickly buildable system components. The purpose of this paper is to introduce such components through a tutorial and to show how can be they used to build a GUI-based application, and to further utilize such applications to analyze the data through visualization.

Index Terms- GUI, Widget, Framework, Window, WIMP, Event-driven Programming, Data Visualization.

1. INTRODUCTION

An interface can be used to establish an interaction and communication between machine (or software or application) and its users. This interface argument or parameter or point can be a dashboard, a menu, a simple display screen, even a mouse or a keyboard. *The designers of user interface are increasingly realizing that it is important to provide a high degree of end-user customization* [6]. In the very beginning the interface was only character-based or command-driven. In the present era, voice-controlled interfaces, gesture-based interfaces have also become pleasurable in comparison with Graphical User Interface (GUI), popular successor of character-based interface (or CUI - Character User Interface or Command-line User Interface) other than the devices. CUIs were in general for designing system programs like operating systems. The CUIs only accept alpha-characters, numeric-characters or some special-characters. These CUIs still maintain their space in the Computer Science. The next best successor of CUIs is GUI because of the cost-effective hardware advancements in terms of graphical terminals.

In today's world of Artificial Intelligence bringing automation to the concepts, Python is a widely used programming language which brings commendable, easy-to-use features. It is being used to develop many Machine Language and Data Science applications. Web applications, communications applications are also being developed using Python. GUI application development feature of the language is also popular among not only the software developers but also the researchers. The *tkinter is the only Graphical User Interface (GUI) framework that is built into the Python standard library* [1]. It is popular because of its cross-platform nature. It works with Linux, macOS and Windows through rendered system components. It helps to build the system quickly giving more weightage or preference to functionality rather than just spangled framework. The applications based on GUI are common because of the ease of interaction. Very easily, many options may be provided at a glance to make further exploration of the application. Python, the programming language, provides several choices for the programmers to design GUI based application including *wxPython*, *Tkinter* and *PyQt*.

Every GUI application starts with building of basic component *window*. The window looks like a container to hold other GUI components. These are widgets or window gadgets, having a powerful functionality. Interestingly only one or two lines of code bring the components to action in the window. *The tkinter package (Tk interface) is the standard Python interface to the Tk GUI toolkit* [2]. As mentioned by Beniz and Epsindola in [4] *tkinter* arose as a good start for their GUI application, as it is the standard GUI package of Python. An visualization tool, ComVisMD (Compact Visualization of Multidimensional Data) developed using graphics package of Python is being transformed into its latest version using *tkinter*. In this paper we tried to introduce *tkinter* as an instrumental asset for the GUI developers in the form of a tutorial. The components of *tkinter*, additionally support in a way for the programmers to implement Event-driven programming also. Event-driven programming is a way of writing programs in which the actions (leading to events) of the user, decides the flow of execution. Event is a user action, like a mouse click or key press that causes a GUI application to respond. An association between 'a widget, an event and an event handler' is called binding. The event handler is executed when the event occurs in the widget.

2. DEFINITIONS

Irrespective of the programming language used, the GUI elements or components are very much common to all the programming languages like java, C++ or Python. These components are used specify the appearance of the user interface in graphical form. Such visual conventions represent information to be conveyed to the users of the GUI application. In general, these conventional components in the form of most common style are nicknamed to WIMP [8]: window, icon, menu, pointer. With reference to Shipman in [3], a GUI application element, *Window*, is a primary component and it is further augmented to contain many widgets (window gadgets or elements).

The table, Tab.1 lists few of such elements and their definition (meaning or purpose).

Element	definition (meaning or purpose)
Window	a rectangular area on a display screen
Top-level window	a window existing independently on screen, one can move it, minimize it, resize it and close it
Widget	a basic component (window gadget or object) of GUI application
Frame	a dedicated rectangular area of window, invisible container acts as a repository to hold other widgets child
Parent	parent-child relationship happens whenever widgets are created. A frame created will become parent to hold a label widget (a child)
Label	widget used to give a textual reference to other widget for identification
Button	widget containing text or an image that commands or triggers an event when pressed
Canvas	an area to display lines, rectangles, circles like shapes
Entry	a region where users can type message or text
Menu	a set of options available for the users is displayed on a display screen
Icon	a small graphical image to represent a closed window

Tab.1. Elements of GUI with definitions

3. LINES OF CODE FOR FEW WIDGETS

In this section we will show how few lines of code of Python language can be used to build GUI application.

In order to use any of the widgets of *tkinter* it is necessary to import the package as used in the line of code:

```
import tkinter as tk     #tk is the alias name for tkinter, in rest of the program 'tk' is used
```

Two lines used to create a window named 'win' with title 'ComVisMD':

```
win = tk.Tk()
win.title("ComVisMD")
```

The window created using these two lines is shown in figure, Fig.1.

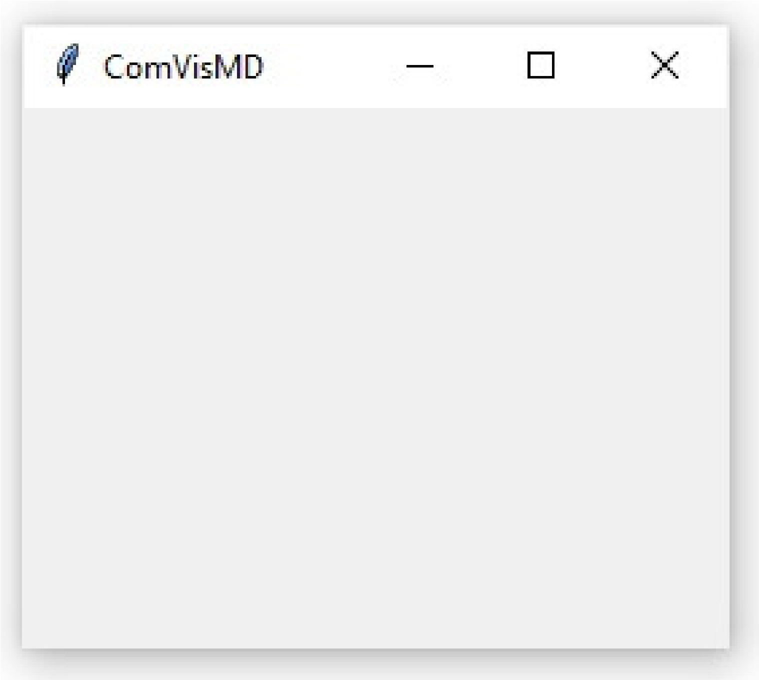


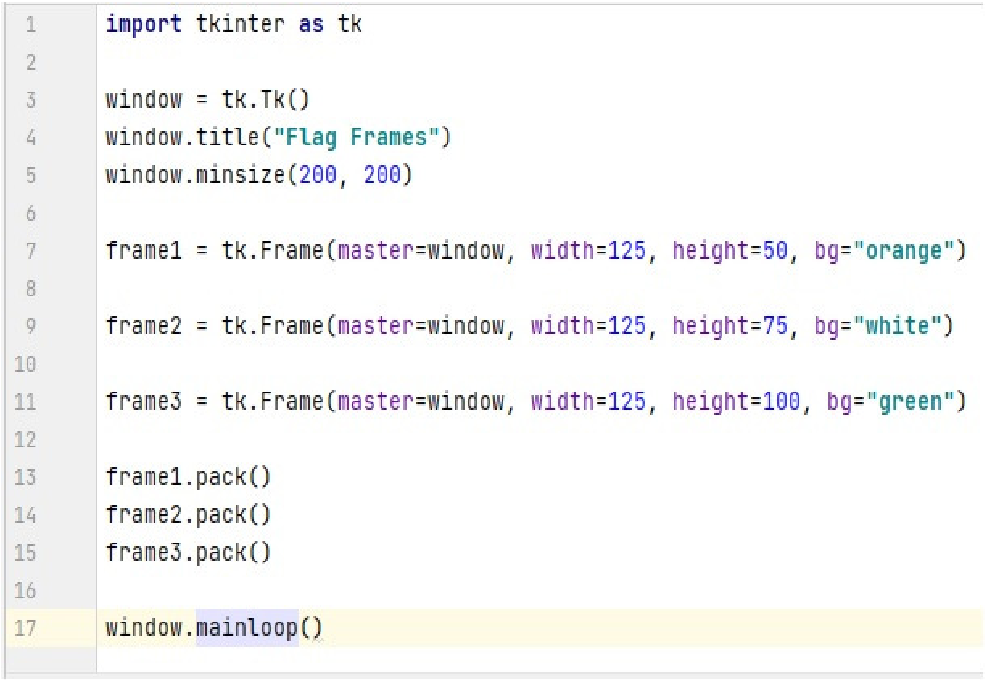
Fig.1. Creation of a window titled "ComVisMD"

```
label = tk.Label(win, text="Label") #code line used to add a label widget to a window, 'win':
button = tk.Button(win, text="Button", width=15, height=5) #A line used to add a button
                                                            #widget to a window
entry = tk.Entry(win)                  #used to add an entry widget to a window, 'win':
```

Note that `entry.get()` can be used to retrieve the entered value for `entry` widget.

```
frame = tk.Frame(win) #to add a frame widget to a window, 'win':
```

The Python code (script) to create three frames in a window is shown in figure, Fig.2. The figure, Fig.3. shows the output.



```
import tkinter as tk

window = tk.Tk()
window.title("Flag Frames")
window.minsize(200, 200)

frame1 = tk.Frame(master=window, width=125, height=50, bg="orange")

frame2 = tk.Frame(master=window, width=125, height=75, bg="white")

frame3 = tk.Frame(master=window, width=125, height=100, bg="green")

frame1.pack()
frame2.pack()
frame3.pack()

window.mainloop()
```

Fig.2. Python code to create 3 frames

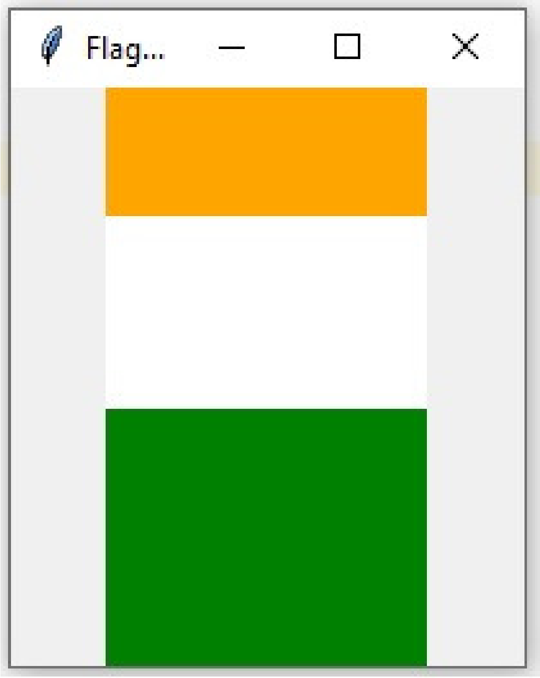
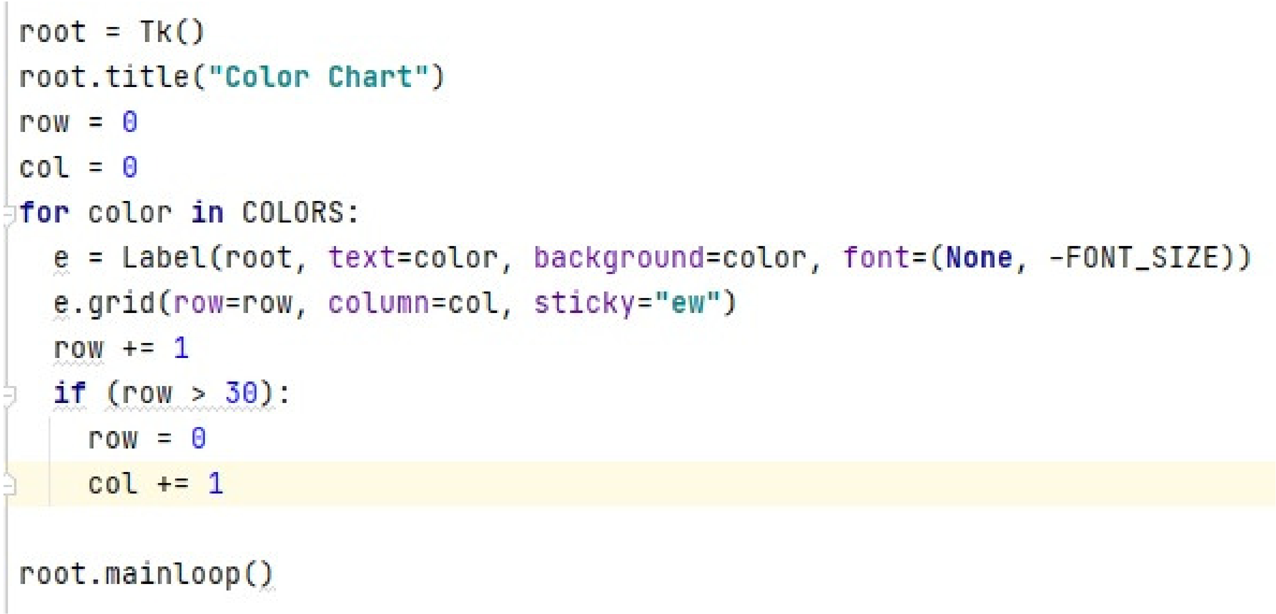


Fig.3. Output of the code of figure Fig.2.

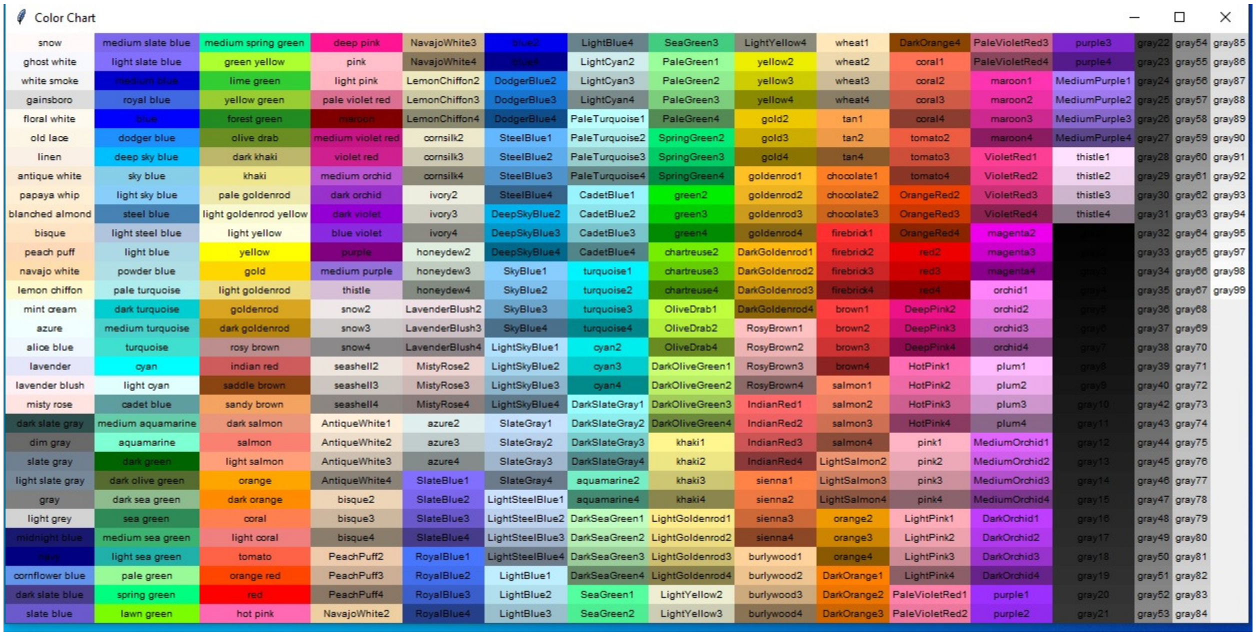
In the figure, Fig.4. the widget ‘Label’ is used in the Python script to display all possible color names, from the color ‘snow’ to color ‘gray99’. The background of the label is colored with respective color name with option, ‘background’. The labels are created in a window named ‘root’ with title ‘Color Chart’. The grid method used with label is a geometry manager of Python used to place the widget at proper row and column position of the display screen. The output of the script is shown in the figure, Fig.5.



```
root = Tk()
root.title("Color Chart")
row = 0
col = 0
for color in COLORS:
  e = Label(root, text=color, background=color, font=(None, -FONT_SIZE))
  e.grid(row=row, column=col, sticky="ew")
  row += 1
  if (row > 30):
    row = 0
    col += 1

root.mainloop()
```

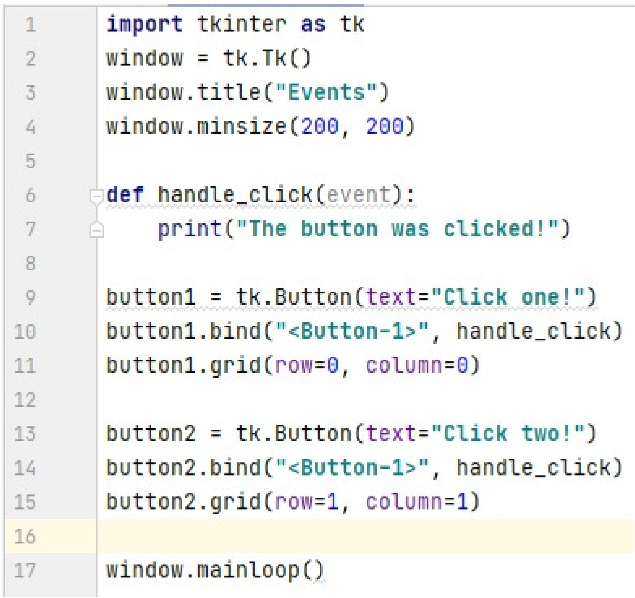
Fig.4. Python script to use label widget to display all possible colors



snow	medium slate blue	medium spring green	deep pink	NavajoWhite3	blue2	LightBlue4	SeaGreen3	LightYellow4	wheat1	DarkOrange4	PaleVioletRed3	purple3	gray22	gray54	gray85
ghost white	light slate blue	green yellow	pink	NavajoWhite4	blue4	LightCyan2	PaleGreen1	yellow2	wheat2	coral1	PaleVioletRed4	purple4	gray23	gray55	gray86
white smoke	medium blue	lime green	light pink	LemonChiffon2	DodgerBlue2	LightCyan3	PaleGreen2	yellow3	wheat3	coral2	maroon1	MediumPurple1	gray24	gray56	gray87
gainsboro	royal blue	yellow green	pale violet red	LemonChiffon3	DodgerBlue3	LightCyan4	PaleGreen3	yellow4	wheat4	coral3	maroon2	MediumPurple2	gray25	gray57	gray88
floral white	blue	forest green	maroon	LemonChiffon4	DodgerBlue4	PaleTurquoise1	PaleGreen4	gold2	tan1	coral4	maroon3	MediumPurple3	gray26	gray58	gray89
old lace	dodger blue	olive drab	medium violet red	cornsilk2	SteelBlue1	PaleTurquoise2	SpringGreen2	gold3	tan2	tomato2	maroon4	MediumPurple4	gray27	gray59	gray90
linen	deep sky blue	dark khaki	violet red	cornsilk3	SteelBlue2	PaleTurquoise3	SpringGreen3	gold4	tan4	tomato3	VioletRed1	thistle1	gray28	gray60	gray91
antique white	sky blue	khaki	medium orchid	cornsilk4	SteelBlue3	PaleTurquoise4	SpringGreen4	goldenrod1	chocolate1	tomato4	VioletRed2	thistle2	gray29	gray61	gray92
papaya whip	light sky blue	pale goldenrod	dark orchid	ivory2	SteelBlue4	CadetBlue1	green2	goldenrod2	chocolate2	OrangeRed2	VioletRed3	thistle3	gray30	gray62	gray93
blanched almond	steel blue	light goldenrod yellow	dark violet	ivory3	DeepSkyBlue2	CadetBlue2	green3	goldenrod3	chocolate3	OrangeRed3	VioletRed4	thistle4	gray31	gray63	gray94
bisque	light steel blue	light yellow	blue violet	ivory4	DeepSkyBlue3	CadetBlue3	green4	goldenrod4	firebrick1	OrangeRed4	magenta2	[illegible]	gray32	gray64	gray95
peach puff	light blue	yellow	purple	honeydew2	DeepSkyBlue4	CadetBlue4	chartreuse2	DarkGoldenrod1	firebrick2	red2	magenta3	gray2	gray33	gray65	gray97
navajo white	powder blue	gold	medium purple	honeydew3	SkyBlue1	turquoise1	chartreuse3	DarkGoldenrod2	firebrick3	red3	magenta4	gray3	gray34	gray66	gray98
lemon chiffon	pale turquoise	light goldenrod	thistle	honeydew4	SkyBlue2	turquoise2	chartreuse4	DarkGoldenrod3	firebrick4	red4	orchid1	gray4	gray35	gray67	gray99
mint cream	dark turquoise	goldenrod	snow2	LavenderBlush2	SkyBlue3	turquoise3	OliveDrab1	DarkGoldenrod4	brown1	DeepPink2	orchid2	gray5	gray36	gray68	
azure	medium turquoise	dark goldenrod	snow3	LavenderBlush3	SkyBlue4	turquoise4	OliveDrab2	RosyBrown1	brown2	DeepPink3	orchid3	gray6	gray37	gray69	
alice blue	turquoise	rosy brown	snow4	LavenderBlush4	LightSkyBlue1	cyan2	OliveDrab4	RosyBrown2	brown3	DeepPink4	orchid4	gray7	gray38	gray70	
lavender	cyan	indian red	seashell2	MistyRose2	LightSkyBlue2	cyan3	DarkOliveGreen1	RosyBrown3	brown4	HotPink1	plum1	gray8	gray39	gray71	
lavender blush	light cyan	saddle brown	seashell3	MistyRose3	LightSkyBlue3	cyan4	DarkOliveGreen2	RosyBrown4	salmon1	HotPink2	plum2	gray9	gray40	gray72	
misty rose	cadet blue	sandy brown	seashell4	MistyRose4	LightSkyBlue4	DarkSlateGray1	DarkOliveGreen3	IndianRed1	salmon2	HotPink3	plum3	gray10	gray42	gray73	
dark slate gray	medium aquamarine	dark salmon	AntiqueWhite1	azure2	SlateGray1	DarkSlateGray2	DarkOliveGreen4	IndianRed2	salmon3	HotPink4	plum4	gray11	gray43	gray74	
dim gray	aquamarine	salmon	AntiqueWhite2	azure3	SlateGray2	DarkSlateGray3	khaki1	IndianRed3	salmon4	pink1	MediumOrchid1	gray12	gray44	gray75	
slate gray	dark green	light salmon	AntiqueWhite3	azure4	SlateGray3	DarkSlateGray4	khaki2	IndianRed4	LightSalmon2	pink2	MediumOrchid2	gray13	gray45	gray76	
light slate gray	dark olive green	orange	AntiqueWhite4	SlateBlue1	SlateGray4	aquamarine2	khaki3	sienna1	LightSalmon3	pink3	MediumOrchid3	gray14	gray46	gray77	
gray	dark sea green	dark orange	bisque2	SlateBlue2	LightSteelBlue1	aquamarine4	khaki4	sienna2	LightSalmon4	pink4	MediumOrchid4	gray15	gray47	gray78	
light grey	sea green	coral	bisque3	SlateBlue3	LightSteelBlue2	DarkSeaGreen1	LightGoldenrod1	sienna3	orange2	LightPink1	DarkOrchid1	gray16	gray48	gray79	
midnight blue	medium sea green	light coral	bisque4	SlateBlue4	LightSteelBlue3	DarkSeaGreen2	LightGoldenrod2	sienna4	orange3	LightPink2	DarkOrchid2	gray17	gray49	gray80	
navy	light sea green	tomato	PeachPuff2	RoyalBlue1	LightSteelBlue4	DarkSeaGreen3	LightGoldenrod3	burlywood1	orange4	LightPink3	DarkOrchid3	gray18	gray50	gray81	
cornflower blue	pale green	orange red	PeachPuff3	RoyalBlue2	LightBlue1	DarkSeaGreen4	LightGoldenrod4	burlywood2	DarkOrange1	LightPink4	DarkOrchid4	gray19	gray51	gray82	
dark slate blue	spring green	red	PeachPuff4	RoyalBlue3	LightBlue2	SeaGreen1	LightYellow2	burlywood3	DarkOrange2	PaleVioletRed1	purple1	gray20	gray52	gray83	
slate blue	lawn green	hot pink	NavajoWhite2	RoyalBlue4	LightBlue3	SeaGreen2	LightYellow3	burlywood4	DarkOrange3	PaleVioletRed2	purple2	gray21	gray53	gray84	

Fig.5. Output of the script of figure, Fig.4.

The script of the figure, Fig.6. shows the use of the widget ‘Button’ for event handling.



```python
import tkinter as tk
window = tk.Tk()
window.title("Events")
window.minsize(200, 200)

def handle_click(event):
    print("The button was clicked!")

button1 = tk.Button(text="Click one!")
button1.bind("<Button-1>", handle_click)
button1.grid(row=0, column=0)

button2 = tk.Button(text="Click two!")
button2.bind("<Button-1>", handle_click)
button2.grid(row=1, column=1)

window.mainloop()
```

Fig.6. Python script to use button widgets to handle the simple events

A method ‘bind’ is used to connect the button and the functionality of handler. When user presses the button then the respective event is happening to display the connecting message. The output of the script given in the figure, Fig.6. is shown in the figures Fig.7. and Fig.8.

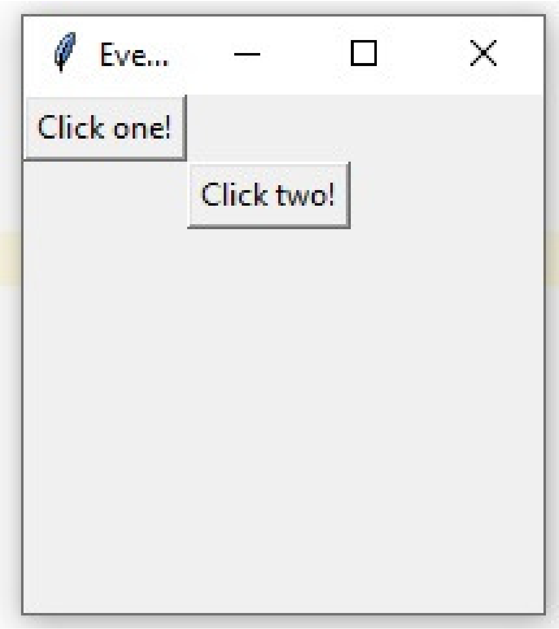
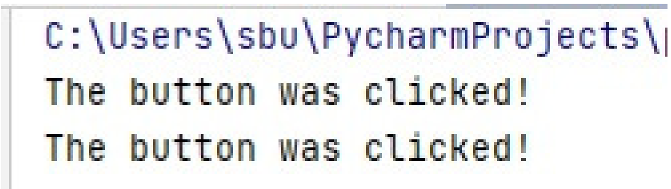


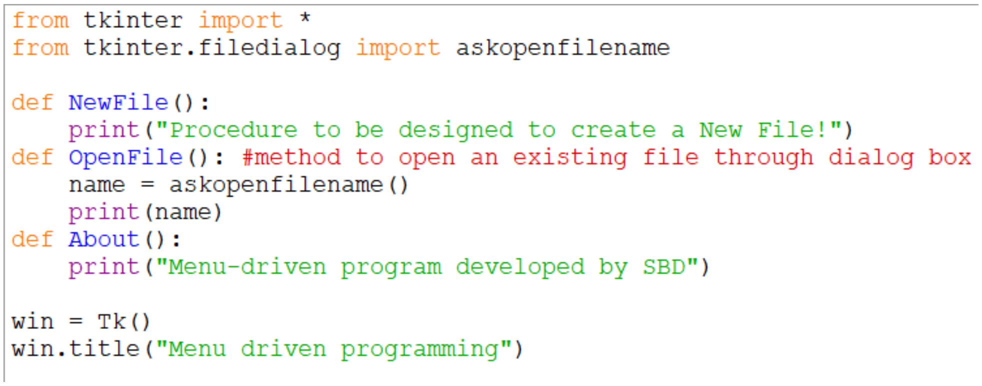
Fig.7. The buttons ‘Click one!’ and ‘Click two!’ created in a window named ‘root’



```
C:\Users\sbu\PycharmProjects\
The button was clicked!
The button was clicked!
```

Fig.8. The events handled after pressing the buttons ‘Click one!’ and ‘Click two!’

The code shown in the figures Fig.9. and Fig.10 can be used to design a menu driven program. The menu is designed with three options, **File**, **Help** and **Exit**. A function can be linked with an option through 'command' option. At the click of the option, the linked function is executed. The output of the program is shown in the figures Fig.11. and Fig.12.

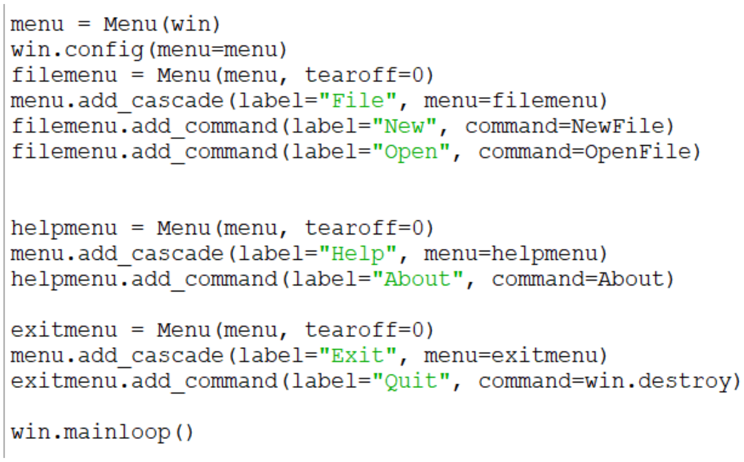


```
from tkinter import *
from tkinter.filedialog import askopenfilename

def NewFile():
    print("Procedure to be designed to create a New File!")
def OpenFile(): #method to open an existing file through dialog box
    name = askopenfilename()
    print(name)
def About():
    print("Menu-driven program developed by SBD")

win = Tk()
win.title("Menu driven programming")
```

Fig.9. First part of a python program to define functions for 'menu options' of a menu-driven program



```
menu = Menu(win)
win.config(menu=menu)
filemenu = Menu(menu, tearoff=0)
menu.add_cascade(label="File", menu=filemenu)
filemenu.add_command(label="New", command=NewFile)
filemenu.add_command(label="Open", command=OpenFile)


helpmenu = Menu(menu, tearoff=0)
menu.add_cascade(label="Help", menu=helpmenu)
helpmenu.add_command(label="About", command=About)

exitmenu = Menu(menu, tearoff=0)
menu.add_cascade(label="Exit", menu=exitmenu)
exitmenu.add_command(label="Quit", command=win.destroy)

win.mainloop()
```

Fig.10. Second part of the python program to define menu of a menu-driven program

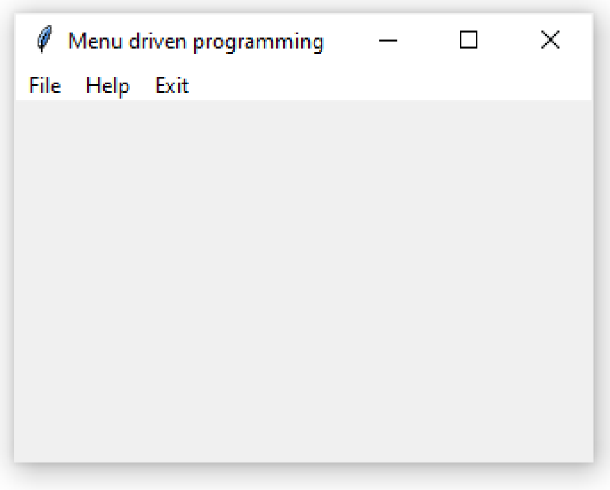


Fig.11. Output showing the display of menu

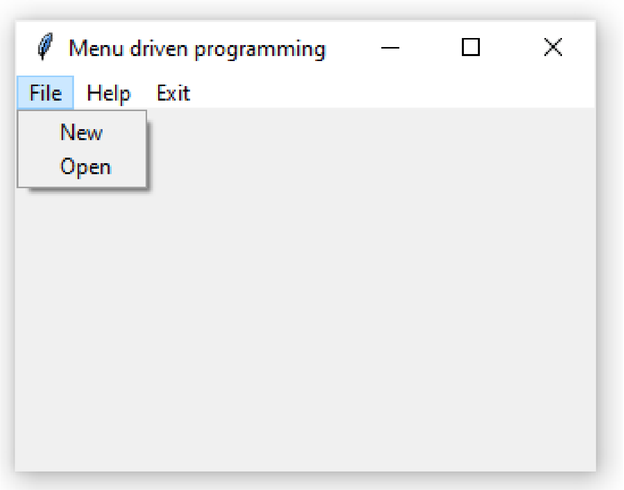
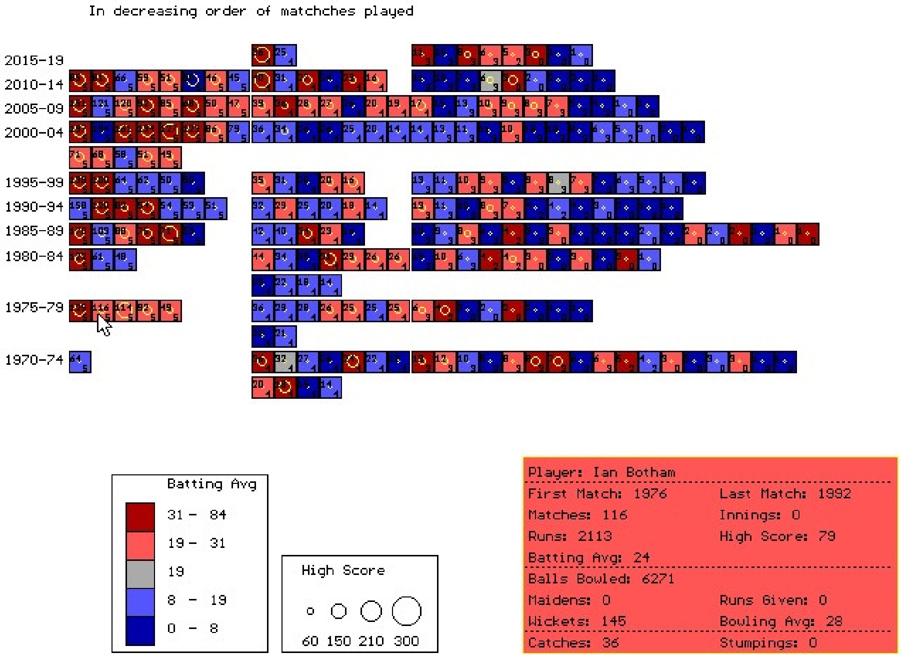


Fig.12. Output after clicking menu option

4. GUI FOR ComVisMD

ComVisMD is the name given to the system developed for **Com**pact **Vis**ualization of **M**ultidimensional **D**ata. Multi-featured data present in two-dimensional format motivated us to design such a system to project data-features in different dimensions of a mapped display like: **column** (vertical dimension), **row** (horizontal dimension), **color** dimension for classification of displayed cells, **hole** dimension to inscribe proportional value on the cell, **shape** dimension to decide the shape of a cell of categorical value.



Player: Ian Botham
First Match: 1976 Last Match: 1992
Matches: 116 Innings: 0
Runs: 2113 High Score: 79
Batting Avg: 24
Balls Bowled: 6271
Maidens: 0 Runs Given: 0
Wickets: 145 Bowling Avg: 28
Catches: 36 Stumpings: 0

Fig.13. Map of the 1st version of ComVisMD

The map shown in the figure, Fig.13. is the first version of the map project using C language [9]. In the latest version of ComVisMD, the interface is developed using *tkinter*. A main widget, 'Menu' is used to provide different options for the user to move on. The designed menu as GUI is shown in the following figure, Fig 14. It has options 'File', 'Dimension' and 'Quit'. The option 'File' is used to chose a CSV (Comma Separated Value) file for visual analysis by selecting the 'OpenCSV' option from the drop-down menu. The drop-down menu has second option to view the raw content of the CSV file by chosing the 'Display' option.

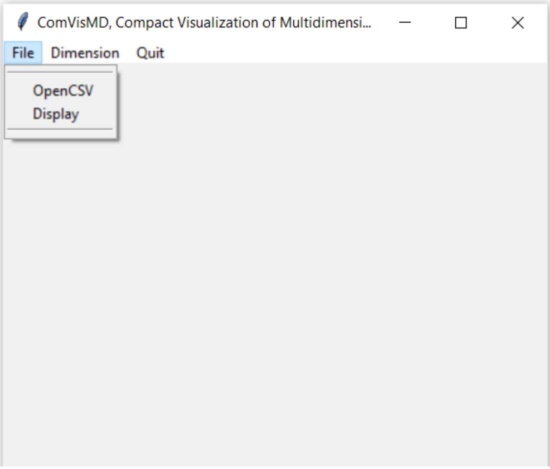


Fig.14. 'Main Menu' and 'File Menu' of ComVisMD

When the user chose the ‘OpenCSV’ option a dialogue window is displayed to choose the file from the current directory, as shown in the figures, Fig 15 and Fig.16. After the selection of a CSV file like ‘Auto’, user can now choose ‘Display’ option to see the contents in raw form, as shown in the figure, Fig 17.

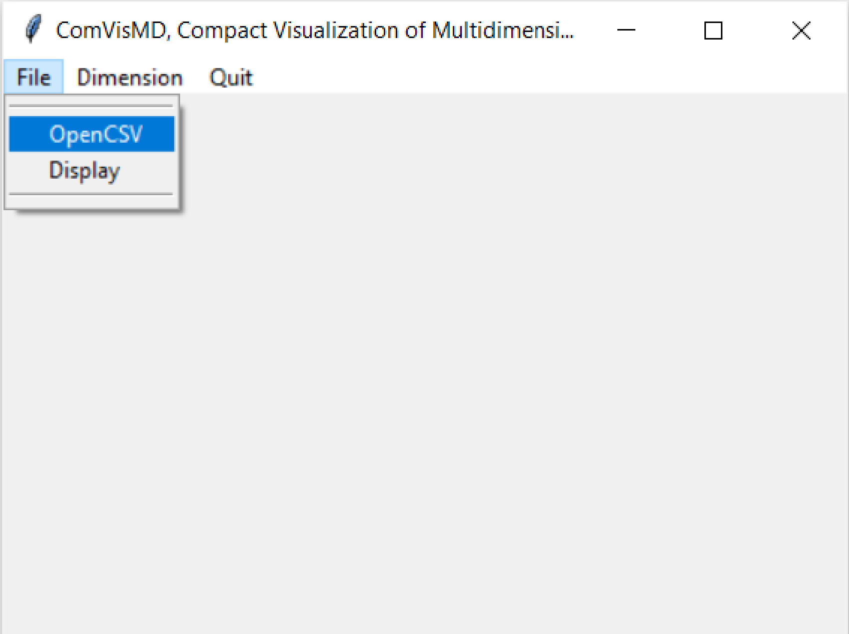
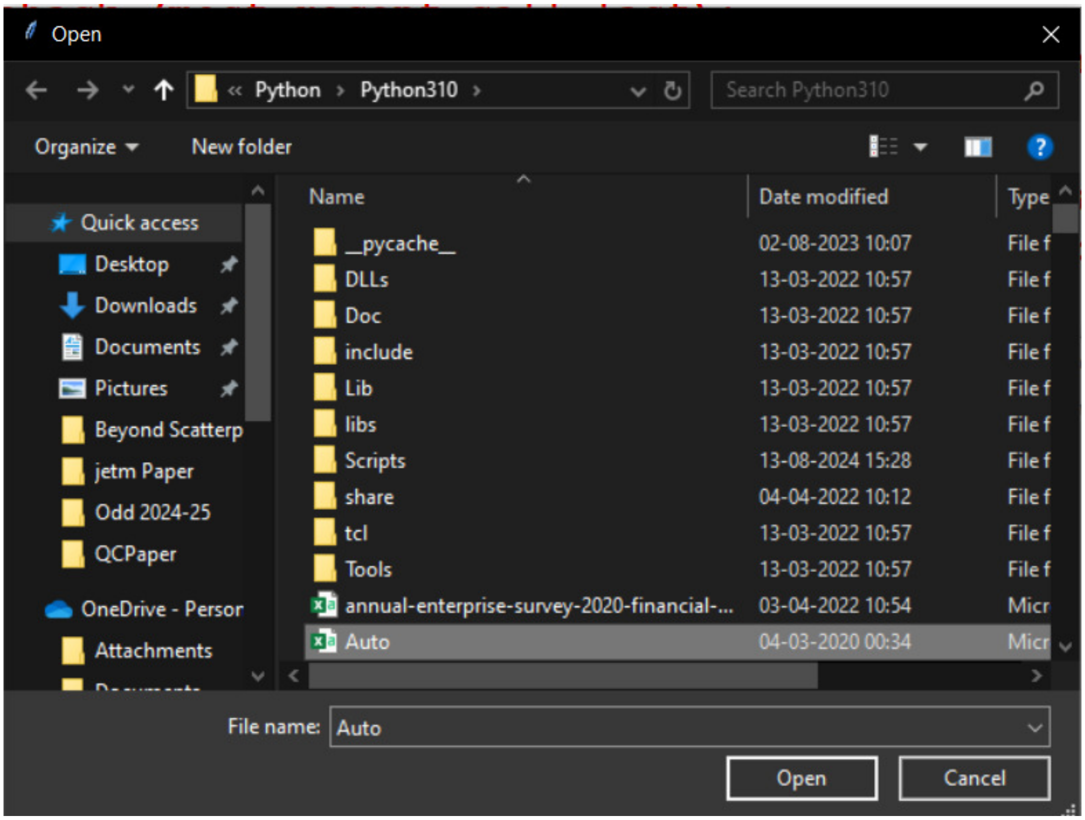


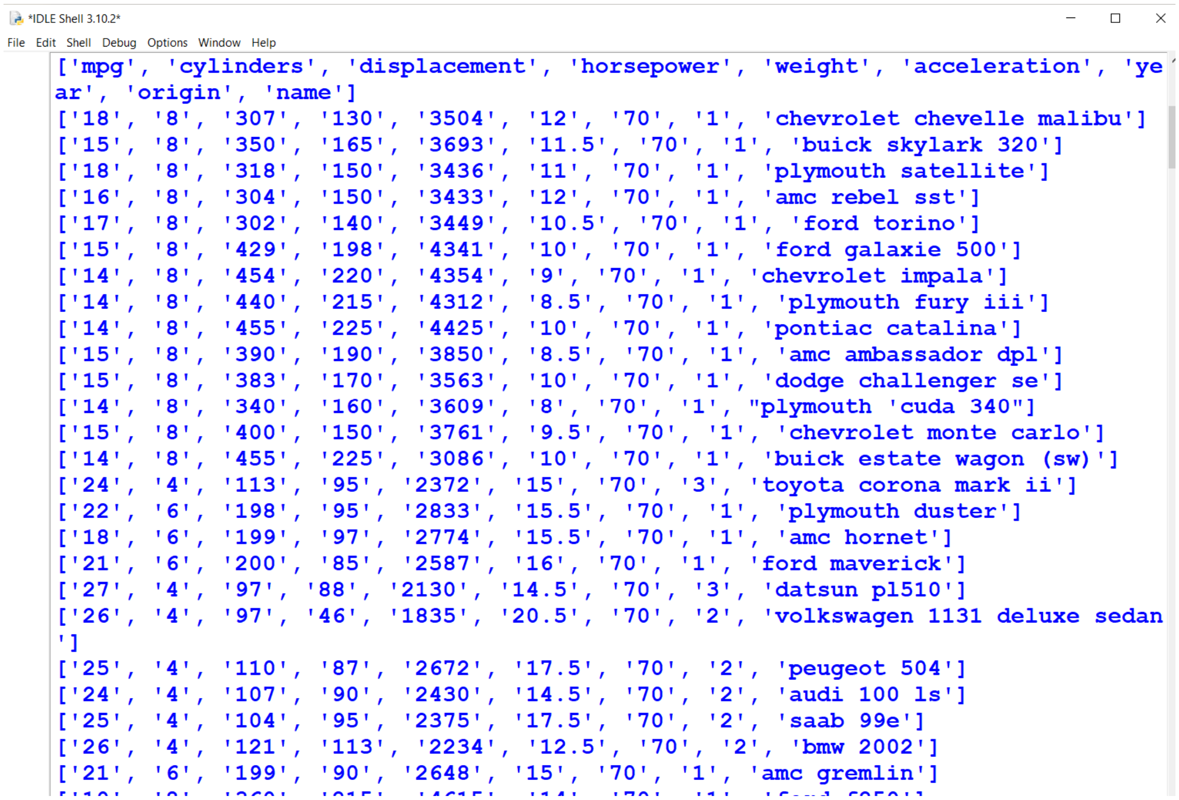
Fig.15. ‘OpenCSV’ option chosen from the ‘File Menu’ of ComVisMD



Name	Date modified	Type
__pycache__	02-08-2023 10:07	File f
DLLs	13-03-2022 10:57	File f
Doc	13-03-2022 10:57	File f
include	13-03-2022 10:57	File f
Lib	13-03-2022 10:57	File f
libs	13-03-2022 10:57	File f
Scripts	13-08-2024 15:28	File f
share	04-04-2022 10:12	File f
tcl	13-03-2022 10:57	File f
Tools	13-03-2022 10:57	File f
annual-enterprise-survey-2020-financial-...	03-04-2022 10:54	Micr
Auto	04-03-2020 00:34	Micr

File name: Auto

Fig.16. Dialogue window to show the selection of a file, ‘Auto’



```
['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'ye
ar', 'origin', 'name']
['18', '8', '307', '130', '3504', '12', '70', '1', 'chevrolet chevelle malibu']
['15', '8', '350', '165', '3693', '11.5', '70', '1', 'buick skylark 320']
['18', '8', '318', '150', '3436', '11', '70', '1', 'plymouth satellite']
['16', '8', '304', '150', '3433', '12', '70', '1', 'amc rebel sst']
['17', '8', '302', '140', '3449', '10.5', '70', '1', 'ford torino']
['15', '8', '429', '198', '4341', '10', '70', '1', 'ford galaxie 500']
['14', '8', '454', '220', '4354', '9', '70', '1', 'chevrolet impala']
['14', '8', '440', '215', '4312', '8.5', '70', '1', 'plymouth fury iii']
['14', '8', '455', '225', '4425', '10', '70', '1', 'pontiac catalina']
['15', '8', '390', '190', '3850', '8.5', '70', '1', 'amc ambassador dpl']
['15', '8', '383', '170', '3563', '10', '70', '1', 'dodge challenger se']
['14', '8', '340', '160', '3609', '8', '70', '1', "plymouth 'cuda 340"]
['15', '8', '400', '150', '3761', '9.5', '70', '1', 'chevrolet monte carlo']
['14', '8', '455', '225', '3086', '10', '70', '1', 'buick estate wagon (sw)']
['24', '4', '113', '95', '2372', '15', '70', '3', 'toyota corona mark ii']
['22', '6', '198', '95', '2833', '15.5', '70', '1', 'plymouth duster']
['18', '6', '199', '97', '2774', '15.5', '70', '1', 'amc hornet']
['21', '6', '200', '85', '2587', '16', '70', '1', 'ford maverick']
['27', '4', '97', '88', '2130', '14.5', '70', '3', 'datsun pl510']
['26', '4', '97', '46', '1835', '20.5', '70', '2', 'volkswagen 1131 deluxe sedan
']
['25', '4', '110', '87', '2672', '17.5', '70', '2', 'peugeot 504']
['24', '4', '107', '90', '2430', '14.5', '70', '2', 'audi 100 ls']
['25', '4', '104', '95', '2375', '17.5', '70', '2', 'saab 99e']
['26', '4', '121', '113', '2234', '12.5', '70', '2', 'bmw 2002']
['21', '6', '199', '90', '2648', '15', '70', '1', 'amc gremlin']
['10', '8', '360', '215', '4615', '14', '70', '1', 'ford f250']
```

Fig.17. Raw contents of 'Auto' file

The next option, 'Dimension' has two groups of options. In the first group, there are five options for our Compact Display's map selection and the second group option 'Map View' is used to display the compact map to display the cells (objects or records) of selected CSV file for example, 'Auto'. The 'Horizontal' option of first group, is used to select one of the attributes to map the objects in horizontal direction. 'Vertical' for vertical direction, 'Color' to color the respective 'cell', 'Hole' to inscribe a scaled circle in the 'cell' and 'Shape' to decide the cell shape like, square, triangle or hexagon. Figure, Fig 18 shows the options of 'Dimension' option of main menu.

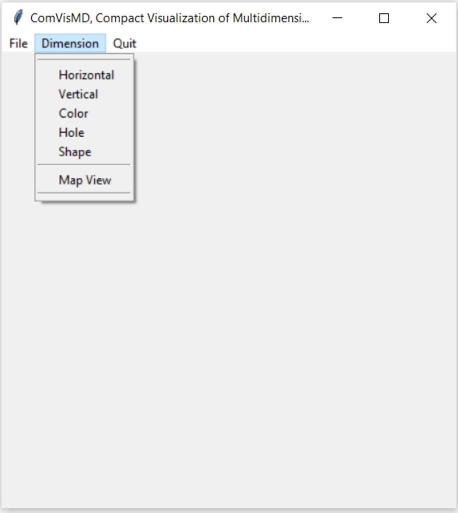


Fig.18. Two groups of options of 'Dimension' option

The following figures, Fig.19, Fig.20, Fig.21, Fig.22, Fig.23, Fig.24 reflects the change in colors of respective options when they are chosen for ‘Horizontal’, ‘Vertical’, ‘Color’, ‘Hole’, and ‘Shape’ respectively.

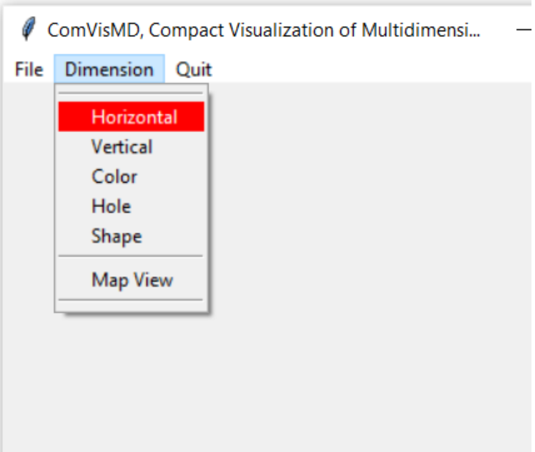


Fig.19. ‘Horizontal’ option selected from ‘Dimension’ option

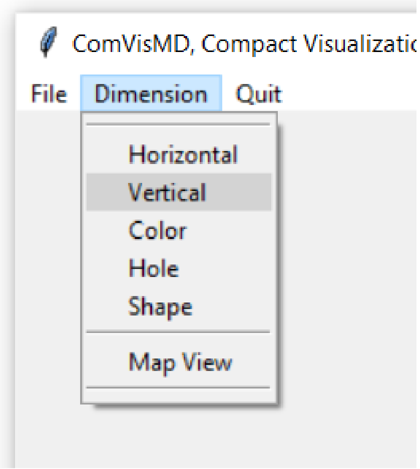


Fig.20. ‘Vertical’ option selected

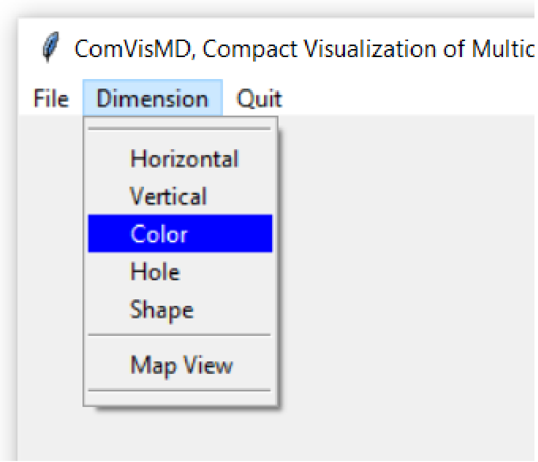


Fig.21. ‘Color’ option selected

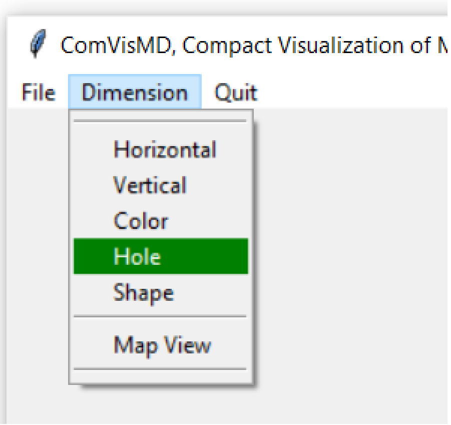


Fig.22. ‘Hole’ option selected

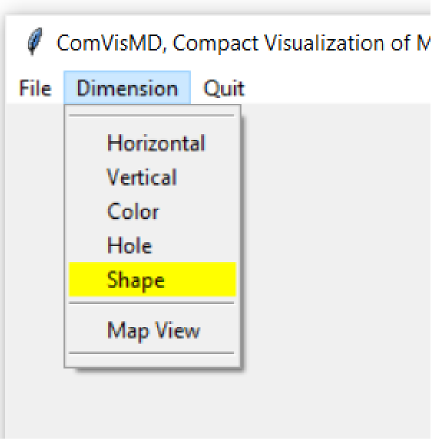
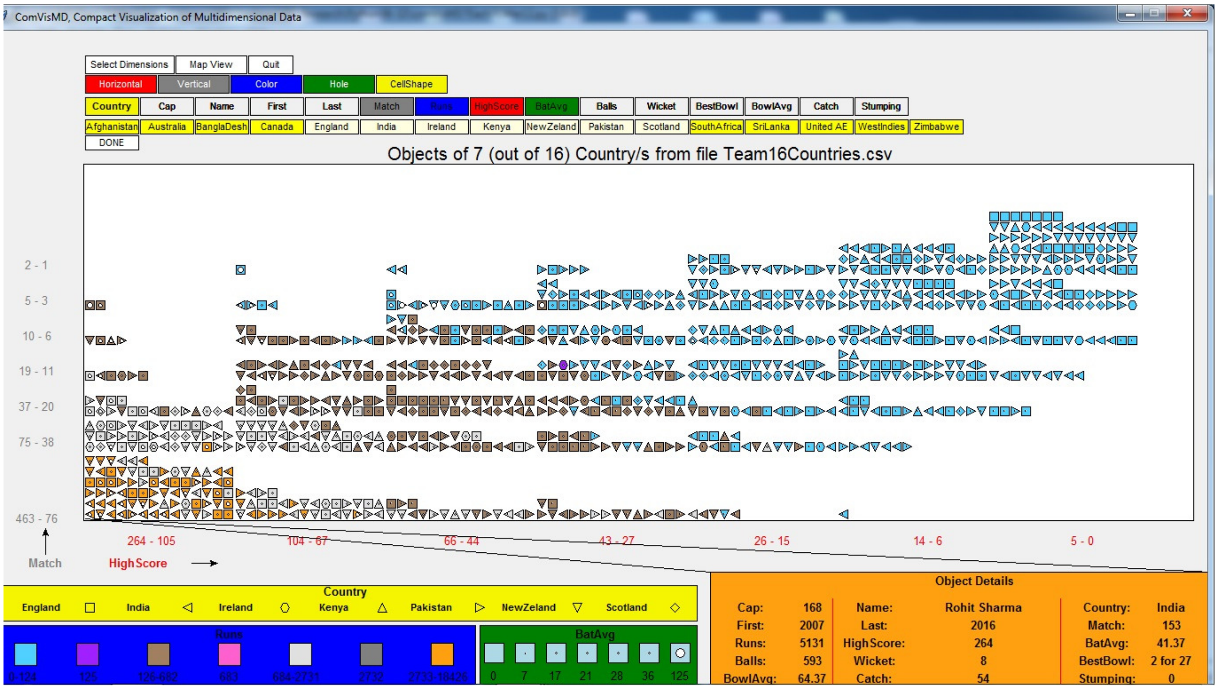


Fig.23. ‘Shape’ option selected

The second version of ComVisMD map with reference to [10] is shown in the figure, Fig.24. projected using graphics package of Python. The map displayed in compact form contains the objects in the form of respective shaped cells from the CSV file ‘Team16Countries.csv’. The attribute (dimension) ‘HighScore’ is chosen as ‘Horizontal’ option, ‘Matches’ is chosen as ‘Vertical’ option, ‘Runs’ is chosen as ‘Color’ option, ‘BatAvg’ is chosen as ‘Hole’ option. These all four dimensions (attributes) are ‘numerical’ attributes. The fifth option, ‘Country’ dimension is chosen for ‘Shape’ option and it is a categorical attribute. In this way the user can make the choice among the attributes of the CSV file and bring these dimensions for display of the objects in the form of cells in a compact display format. The user of the tool can automatically derive correlation among the objects and also observe that most interesting objects with dark-orange-colored cells are at one corner (lower-left) of the map. This helps in visual analysis of the data. The user can click on any cell to display whole characteristics of the objects, as shown in a large rectangle at the right bottom corner of the figure. In the latest version of ComVisMD, the *tkinter* is used for the development of user interface as discussed earlier in this section.



Object Details

Cap:	168	Name:	Rohit Sharma	Country:	India
First:	2007	Last:	2016	Match:	153
Runs:	5131	HighScore:	264	BatAvg:	41.37
Balls:	593	Wicket:	8	BestBowl:	2 for 27
BowlAvg:	64.37	Catch:	54	Stumping:	0

Fig.24. Map of the 2nd version of ComVisMD

5. CONCLUSION

An effort is made in this paper to make the reader familiar with a GUI framework in the form of easily understandable tutorial with self-explanatory lines of code. Few of the commands can be used directly without even knowing detailed programming. In a simpler way a GUI framework helps a programmer to design an application without writing much of the complex code. The developers can also make use of *tkinter* to develop rapid GUI applications, in turn using them for designing tools for data visualization and analysis.

REFERENCES

[1] https://realpython.com/python-gui-tkinter.

[2] https://docs.python.org/3/library.html.

[3] John W. Shipman, "Tkinter 8.5 reference: a GUI for Python", New Mexico Tech.

[4] D.B. Beniz and A.M. Epsindola, "Using Tkinter of Python to Create Graphical User Interface (GUI) for Scripts in LNLS", *Proceedings of PCaPAC2016*, ISBN 978-3-95450-189-2, 28-Oct-2016.

[5] Ishan Banerjee, Bao Nguyen, Vahid Garousi and Atif Memona, "Graphical user interface (GUI) testing: Systematic mapping and repository", *ELSEVIER- Information and Software Technology*", 55 (2013), 1679-1694, http://dx.doi.org/10.1016/j.infsof.2013.03.004.

[6] Brad A. Myers, ed. "Languages for Developing User Interfaces", Boston: Jones and Bartlett, 1992. 480 pages. ISBN 0-86720-450-8.

[7] Reed, Phillip Kevin, "A Comparison of Programming Languages for Graphical User Interface Programming" (2002). *Chancellor's Honors Program Projects.* https://trace.tennessee.edu/utk_chanhonoproj/590.

[8] D. Alan, F. Janet, A. Gregory D and B. Russel, "Human-Computer Interaction", Pearson Education, III edition, 2004.

[9] Shridhar B. Dandin and Mireille Ducassé, "ComVisMD - compact visualization of multidimensional data: experimenting with cricket players data", 2018, IOP Conf. Ser.: Mater. Sci. Eng. 331 012006, DOI 10.1088/1757-899X/331/1/012006.

[10] Dandin, S.B., Ducassé, M. (2021), "ComVisMD - Compact 2D Visualization of Multidimensional Data: Experimenting with Two Different Datasets", In: Sharma, H., Saraswat, M., Kumar, S., Bansal, J.C. (eds) Intelligent Learning for Computer Vision, CIS 2020, Lecture Notes on Data Engineering and Communications Technologies, vol 61. Springer, Singapore. https://doi.org/10.1007/978-981-33-4582-9_37.