**LABORATORY EXERCISE 2**

**DRAWING 2D SHAPES WITH D SHAPE DRAWER**

# Learning Objectives

* Understand how to create interactive interfaces using Tkinter or similar libraries.
* Learn to capture mouse input for shape drawing operations.
* Knowledge of points, lines, circles, and rectangles for shape manipulation.
* Apply OOP principles to organize shape drawing functionalities efficiently.
* Develop skills to identify and fix shape drawing errors and user interaction errors.
* Exercise creativity to design an intuitive and visually appealing user interface.

# Prerequisite student experiences and knowledge

Students undertaking the D Shape Drawer laboratory exercise should have basic proficiency in Python programming, including knowledge of syntax, data types, loops, conditionals, and functions. Familiarity with GUI programming concepts, such as event-driven programming and GUI libraries like Tkinter, is beneficial but not mandatory. Understanding basic 2D geometry concepts like points, lines, circles, and rectangles is essential for implementing shape drawing functionalities. Problem-solving skills, including the ability to break down complex problems and debug code, are necessary, as is the capability to document code effectively and test programs thoroughly. A positive attitude towards learning and willingness to seek assistance are valuable for completing the exercise.

# Background

Students should have basic Python skills covering syntax, data types, loops, and functions. Familiarity with GUI concepts like event-driven programming and basic geometry (points, lines, etc.) is helpful but not mandatory. Problem-solving abilities, debugging skills, and a positive attitude toward learning are also important for success in the exercise.

# Materials/Resources

* Basic understanding of Python programming.
* Familiarity with GUI programming concepts (e.g., using Tkinter or another GUI library).
* Understanding of basic geometry concepts (e.g., points, lines, circles, rectangles).
* Git is installed on your system.
* GitHub account (for version control and collaboration).

# Laboratory Activity

**Instructions:**

* Ensure that Python is installed on your system.
* Install the Tkinter library if not already available. You can install it using pip: ***pip install tk***
* Install Git on your system if it is not already available. Follow the instructions provided on the [official Git website](https://git-scm.com/) for installation.
* Create a GitHub account if you don't have one already
* Create a new repository on GitHub named "d-shape-drawer" (or any other desired name).
* Clone the repository to your local machine using the following command: ***git clone <repository-url>*** or other methods.
* Open your preferred code editor or IDE to implement the D Shape Drawer program.
* Create a graphical user interface (GUI) window using Tkinter.
* Add buttons, dropdown menus, and canvas for drawing shapes.
* Design an intuitive layout that allows users to select the type of shape to draw and specify its parameters.
* Define functions for shape drawing operations (e.g., points, lines, circles, rectangles).
* Utilize Tkinter canvas to draw shapes based on user input.
* Ensure that users can specify each shape's position, size, and color.
* Implement event handlers to capture mouse input from the canvas.
* Detect mouse clicks and movements to determine shapes' starting and ending points.
* Test the program thoroughly to ensure that users can draw various shapes accurately.
* Debug any issues related to shape drawing, parameter specification, or user interaction.
* Add additional features such as shape resizing, rotation, or deletion.
* Improve the user interface by adding tooltips or status indicators.
* Implement saving and loading functionality to preserve drawn shapes.
* Document your code thoroughly, including comments and explanations for each function and component.
* Test your program extensively to ensure its functionality.
* Submit your completed D Shape Drawer program along with any necessary documentation.
* Create a README.md file in your repository describing the project, its features, and how to run it.
* Submit the link to your GitHub repository along with any necessary documentation.

**Note:** Explore and experiment with different aspects of GUI programming and shape manipulation to enhance their understanding and skills.

# Question

1. Why does good interface design matter in drawing apps like D Shape Drawer? How does it make the app easier to use? Give simple examples of design principles that help.

* Good interface design matters in drawing apps like D Shape Drawer because it helps users to easily understand the concept and use of the app without any confusion. For some instances, clear labels and icons for different tools that has been added make it simple and easier to find what the user really needs. Furthermore, using some other gestures while considering usage familiarities like resizing or rotating shapes enhances the user experience by making the interactions feel natural, smooth, and intuitive. In short, good interface design in drawing apps improves usability and makes creating artwork more enjoyable for users.

1. How does event-driven programming help in drawing shapes in the D Shape Drawer? Explain how mouse events work and the challenges in handling user actions effectively.

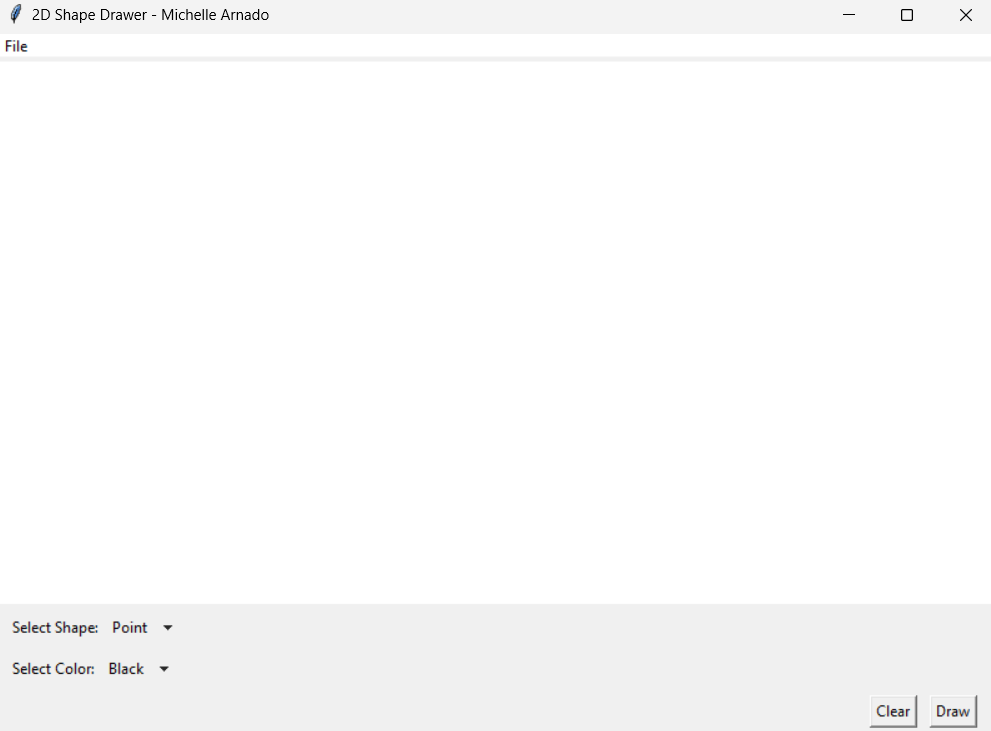
* Event-driven programming in D Shape Drawer helps in drawing shapes by allowing the program to respond user actions, such as mouse clicks movements that the users will do. When you click or move the mouse, it triggers events that the program can detect and respond to accordingly, like drawing a line or a circle on the screen. However, handling these user actions effectively can be challenging because the program needs to accurately interpret various mouse inputs and translate them into meaningful actions on the canvas, ensuring smooth interaction without errors.

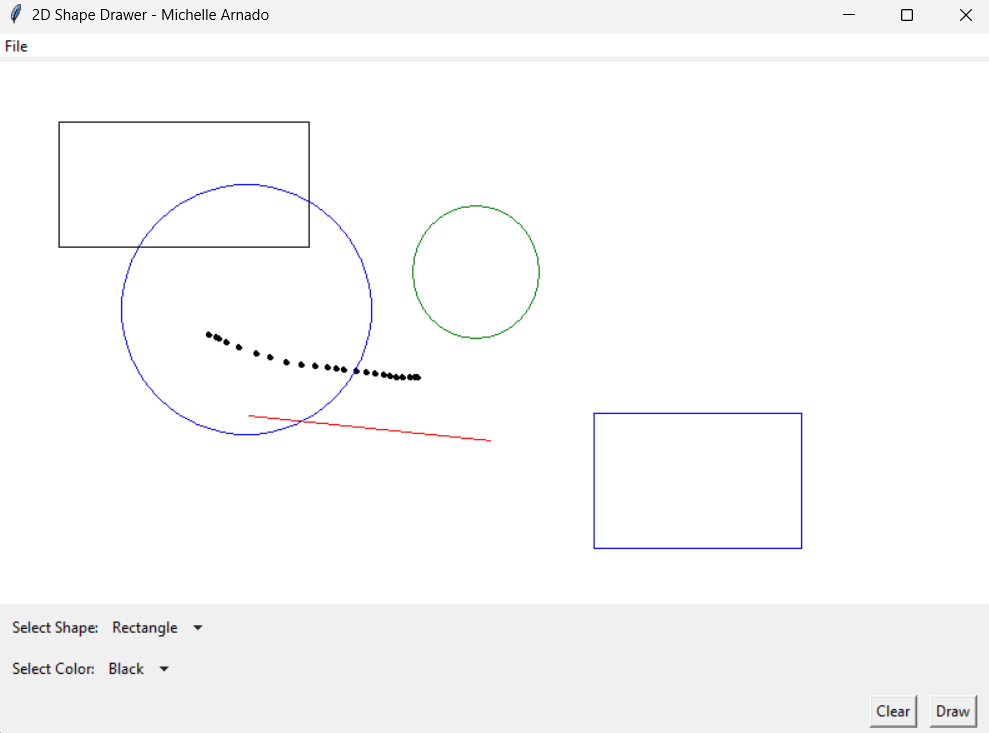
1. What methods are used to handle shapes in apps like D Shape Drawer? Discuss how geometric algorithms and data structures help in drawing and transforming shapes. Provide simple examples.

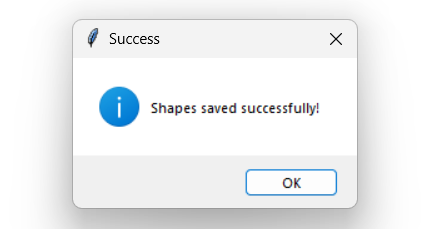
* In apps like D Shape Drawer, shapes are handled using geometric algorithms and data structures. These algorithms help in drawing shapes by calculating their coordinates and properties, while data structures organize the shape's information efficiently. For example, a simple algorithm might calculate the vertices of a square based on its center and size, while a data structure like a linked list could store these vertices in an organized manner. These tools enable the app to draw and transform shapes accurately, allowing users to create various designs easily.

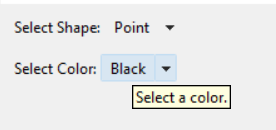
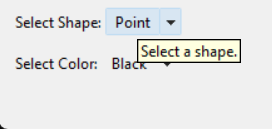
**Output / Results**

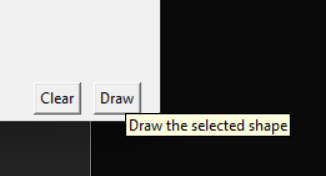
*(Indicate the Result and complete source code)*

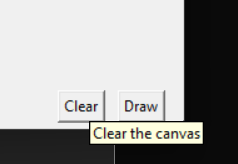


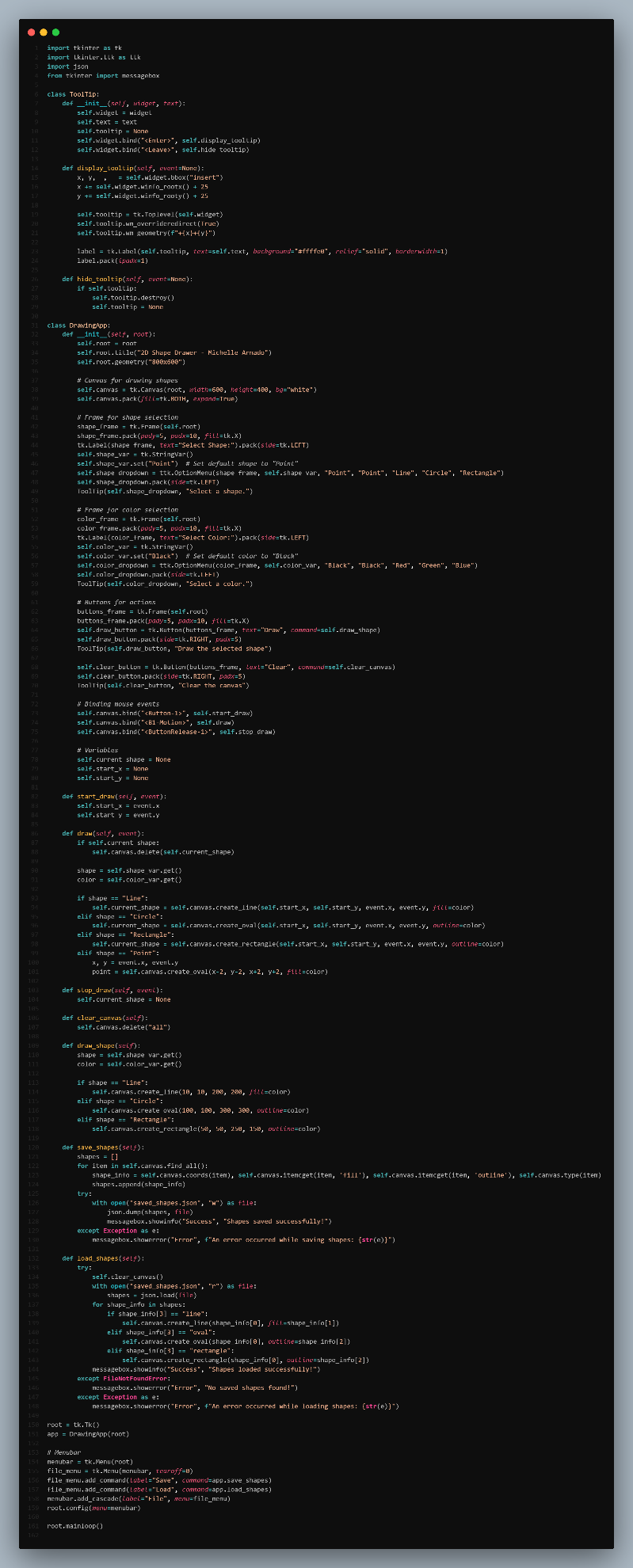


****

****

****

****

****

**GitHub Link: Paste it here**

<https://github.com/mzshil>

**Conclusion**

Creating the D Shape Drawer program was a great learning experience for me as I am new to this kind of work. I delved into the world of GUI programming using Tkinter. I learned how to make computer programs that have buttons, menus, and shapes you can draw. It's like playing with digital building blocks. By the suggestions of our professor, I used a tool called Tkinter and learned how to make things happen when you click on buttons or move the mouse. It helped me get to know new programming language specifically python, I slowly learn this language to somehow honed my python skills and problem-solving abilities which are needed in this kind of field I am in.

Through this laboratory task, I realized the importance of good interface design in enhancing user experience. Clear labels, intuitive layouts, and familiar gestures like resizing and rotating shapes make the app easier to use and more enjoyable for users. Event-driven programming is like listening for someone to clap their hands. When they do, you react. For example, when you click the mouse, the program does something. Testing and debugging are like practicing to make sure you react correctly every time someone claps their hands, so everything works smoothly. Handling shapes in the app, I used methods and ways of organizing to work with shapes, by employing geometric algorithms and data structures. These methods help us figure out where to put the shapes, what they look like, and how to change them accurately.

Generally speaking, this laboratory task taught me a lot and gave me useful knowledge and abilities in making programs with graphical interfaces, manipulating shapes, and finding solutions to problems. By working to this kind of task I gained some practical experience which I somehow consider big and useful for future task in creating computer applications that I or anyone can use. These kind of skills need to be honed because these are important skills that a programmer should have, especially the beginner ones who dream to make a big project someday, for a reason that it allows the students like me who is studying computer science build programs that solve real-world problems and easy for people to understand. Thanks to the insights and another skills I gained from this project.