Tic Tac Toe Project

Tic Tac Toe is one of the most played games and is the best time killer game that you can play anywhere with just a pen and paper. The game is played by two individuals. First, I draw a board with a 3×3 square grid. The first player chooses ‘X’ and draws it on any of the square grid, then it’s the chance of the second player to draw ‘O’ on the available spaces. Like this, the players draw ‘X’ and ‘O’ alternatively on the empty spaces until a player succeeds in drawing 3 consecutive marks either in the horizontal, vertical or diagonal way. Then the player wins the game otherwise the game draws when all spots are filled.

This interesting Python project will be built using the pygame library. Pygame is a great library that will allow me to create the window and draw images and shapes on the window. This way I will capture mouse coordinates and identify the block where I need to mark ‘X’ or ‘O’. Then I will check if the user wins the game or not. The pygame library is an open-source module for the Python programming language specifically intended to help me make games and other multimedia applications. Built on top of the highly portable SDL (Simple DirectMedia Layer) development library, pygame can run across many platforms and operating systems.

The pygame library is composed of a number of Python constructs, which include several different modules. These modules provide abstract access to specific hardware on my system, as well as uniform methods to work with that hardware. For example, display allows uniform access to your video display, while joystick allows abstract control of your joystick.

After importing the pygame library in the example above, the first thing I did was initialize PyGame using pygame.init(). This function calls the separate init() functions of all the included pygame modules. Since these modules are abstractions for specific hardware, this initialization step is required so that I can work with the same code on Linux, Windows, and Mac.

In addition to the modules, pygame also includes several Python classes, which encapsulate non-hardware dependent concepts. One of these is the Surface which, at its most basic, defines a rectangular area. Surface objects are used in many contexts in pygame.

**Surfaces, Images, and Transformations**

“Surface” is the name of the class, so you’d use the name you assigned when you created the surface. For example, if your main display Surface was called “screen” (as it is above), you’d use screen.blit(), not Surface.blit()

**Surface.blit(sourceSurface, destinationRect, optionalSourceRect)-** Copies pixels from one Surface to another. Used to draw images to the screen. If you omit the third argument, the entire source Surface is copied to the area of the destination Surface specified by the Rect in the second argument

**Surface.fill(color)-** Fills surface with a solid color. Argument is a tuple of RGB values. e.g. (255,0,255) for Magenta (maximum red and blue, no green)

**Surface.convert()-** Changes pixel format of the Surface’s image to the format used by the main display. Makes things faster. Use it.

**Surface.convert\_alpha()-** Same as above, but when the Surface’s image has alpha transparency values to deal with.

**Surface.get\_rect()-** Returns a Rect that will tell you the dimensions and location of the surface.

**pygame.image.load(filename)**- Loads image from disk and returns a Surface.

**pygame.transform.rotate(Surface, angle)-** Rotates Surface counterclockwise by degrees.

**pygame.transform.scale(Surface, (width, height))**- Resizes Surface to new resolution.

Link: <https://python101.readthedocs.io/pl/latest/_downloads/pygame192.pdf>

Link: <https://devdocs.io/pygame/>