Two technologies I found for the main usage of Graphic User Interfaces (GUI) in Java are Swing and Java FX. Though Java’s first GUI was neither of these, instead what was developed first was something called Abstract Window Toolkit (AWT). This first rendition was a platform-dependent framework. Meaning that across platforms the AWT would need to tailor itself to those specific components. Which made AWT a burden across operating systems. It used operating system specific code in order for it to look natural on the operating system it was used on. Which is why it eventually got replaced by something called Swing.

Swing on the other hand is a Model View Controller (MVC) which means it is platform-independent, so it doesn’t suffer the same flaw as AWT. Making it more accessible across operating systems. With the new method of a MVC new terminology came about for it to help better understand the purpose. First the Model, which houses the raw data of an application. It is the stuff that gives meaning to your application. It is what the task is or the operation you need to complete. Now this might be the raw data portion, but that is all it is. We as users can not view this raw data, which is where the View part of MVC comes in. View is responsible for giving the user something to perceive. Without it, we most likely be looking at nothing. Okay now we have the raw data in the Model and a way to perceive it with View. That sounds like it, but think of them independently. The Model only deals with data as View only deals with what the user sees. There is nothing that connects the two, which is where Controller comes in. The Controller is the thing that when a user is Viewing something and sees a Model they wish to perform. The Controller acts as the communicator between the Model and View in order to get the desired output. With this understanding of the basics of a MVC methodology this is how Swing operates. Now what Swing does is give a easy menu that has a multitude of possibilities at it’s disposal. The reason for this is that it has been out for a long time and the libraries that are available for it is an extensive list. As well as documentation that defines how each of it’s functions work. It might be basic and a little crude in todays standards of GUI. Though it gets the job done, which is why some people keep their Swing frameworks rather than switch to JavaFX.

JavaFX is the successor of Swing. It might still be a new framework with a lack of experience and clear documentation. But that is quickly changing as time goes on. JavaFX is also an MVC GUI, but it also has a declarative language giving it something similar to Microsoft’s XML format. Now JavaFX utilizes their own rendition of the script, but this makes life easier in which to set up GUI elements on your scene, pane, window or whatever you want to call it. JavaFX being more modern, supports more of the dynamic GUI features that were missing in Swing like 3D graphics support, rich text support, and sensor support to name a few of them. JavaFX might be young, but as time goes on and how easy it is to switch over from Swing to JavaFX. Not to mention that Swing may have been around longer, but that gives JavaFX the backbone to work off of and develop a GUI system for Java that meets more with the times.

References

*Java Swing vs Java FX: Know the 6 most awesome differences*. EDUCBA. (2021, April 30). Retrieved November 11, 2021, from https://www.educba.com/java-swing-vs-java-fx/.

Larkin, P. (2021, January 12). *JavaFX vs swing: The key differences*. Career Karma. Retrieved November 11, 2021, from https://careerkarma.com/blog/javafx-vs-java-swing/.

*MVC: Model, view, Controller*. Codecademy. (n.d.). Retrieved November 11, 2021, from https://www.codecademy.com/articles/mvc.

*What is Java AWT (Abstract window toolkit)*. Software Testing Help. (2021, November 1). Retrieved November 11, 2021, from https://www.softwaretestinghelp.com/java-awt-abstract-window-toolkit/.