

## Graphical User Interface with Python: Enabling Easier Adoption – A Beginner’s Guide

<sup>1\*</sup>Aadhitya Sriram, <sup>2</sup>Bose Sundan

<sup>1\*</sup> Student, Department of Computer Science & Engineering, College of Engineering, Guindy, Anna University, Chennai, India.

<sup>2</sup> Professor, Department of Computer Science & Engineering, College of Engineering, Guindy, Anna University, Chennai, India.

\*Corresponding Author

Date of Submission: 02-08-2024

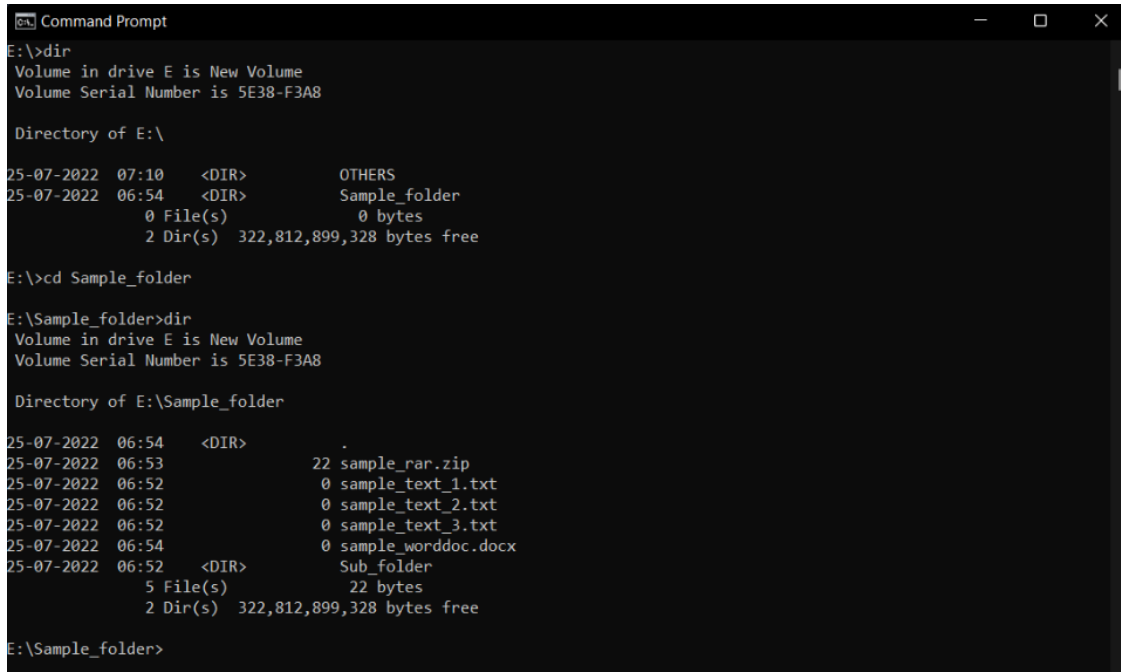
Date of acceptance: 13-08-2024

### I. INTRODUCTION

In recent times, we have grown accustomed to the things happening at the click of a button. Files, Folders, and Applications are presented fashionably as icons on the screen. Customizable desktops, wallpapers, icons and even your cursor are used by everyone which also is a very marketable area of business. These can also be seen in other devices like smartphones where the possibilities are endless. But this wasn't the case for a very long time. Ever since the Modern computers were invented in the 1960s, professionals were required to access all the stored files through something called a 'Terminal'. It was a black screen usually accompanied by either green or grey text boxes through which the user had to input functionality to do something on the computer. If you're a young reader, this might sound alien to you. And most important of all it might sound rigorous or time-consuming to you. And that is all true.

### BRIEF HISTORY OF GUIs

The problem with old Interfaces (which are called 'non-UI' or colloquially as 'NGUI' which corresponds to 'Non-Graphical User Interface') is that the user had to type in functions to even go from one directory to another and none of the files or folders present in that directory was visible without another function. I think you realize how easy it is these days with the click of a button with your cursor. It was also very monotonous to look at with no customization available and it decreased productivity of the users in multiple ways.



```

Command Prompt
E:\>dir
Volume in drive E is New Volume
Volume Serial Number is 5E38-F3A8

Directory of E:\

25-07-2022 07:10 <DIR>          OTHERS
25-07-2022 06:54 <DIR>          Sample_folder
                0 File(s)        0 bytes
                2 Dir(s)  322,812,899,328 bytes free

E:\>cd Sample_folder

E:\Sample_folder>dir
Volume in drive E is New Volume
Volume Serial Number is 5E38-F3A8

Directory of E:\Sample_folder

25-07-2022 06:54 <DIR>          .
25-07-2022 06:53                22 sample_rar.zip
25-07-2022 06:52                0 sample_text_1.txt
25-07-2022 06:52                0 sample_text_2.txt
25-07-2022 06:52                0 sample_text_3.txt
25-07-2022 06:54                0 sample_worddoc.docx
25-07-2022 06:52 <DIR>          Sub_folder
                5 File(s)        22 bytes
                2 Dir(s)  322,812,899,328 bytes free

E:\Sample_folder>

```

**Fig 1: Representation of Olden Days' Interface using Windows Terminal**

This had to change. Experimentations on a GUI began as early as the late 1960s and many prototypes began to emerge. The NLS model implemented a cursor and multiple windows which even allowed local video conferencing. But the whole world was in awe when Apple Computers released their model of GUI which was the classical version of today's MacOS and is referred to as the 'Classic MacOS'. This was the beginning of the GUI wars because the next year Microsoft (along with an IBM contract) developed MS-DOS which like the classic MacOS was a game changer and also eventually led to the 'Windows' which a lot of us now have come to love and appreciate. By the 1990s both Microsoft and Apple updated their respective GUIs and became one of the most iconic rivals in the history of software development.

Other GUIs such as Linux which encompasses a lot of versions to choose from (such as Ubuntu, Kali, arch, etc.) are available as much as open-source software to this day and have a substantial fan following. On the mobile side, the main competitors are Android and IOS which we're all aware of. With a brief history of GUIs, we now embark on a journey on how you can create your GUI if you wish to do so. It should be noted that all the above-mentioned software is generally classified under Operating systems which contain GUI as a part.

## BASIC TERMINOLOGY

Before we start creating a whole new GUI, let's get you caught up on some terms that you may or may not be aware of.

GUI (Guided or Graphical User Interface):

It's a piece of software that allows you to interact with the data innovatively stored in the computer using hand-eye coordination. It significantly improves productivity because it's very easy and fast to use. It allows users to perform functions such as creating folders, navigating through directories finding a desired file, and deleting and reviewing items with a click of a button.

UI (User interface):

This is what is visible to the User and they interact with it to perform a function. It is also the link between the user and the backend functions that occur in a computer. It can either be Graphical or Non-Graphical. But these days most of us prefer graphical solutions as they are hassle-free and increase productivity by ten-fold or even more. Developers are striving to improve UIs to enhance performance and optimize it for the user's best benefit. Users also prefer this to be customizable and easy to use which brings us to the next term.

UX (User Experience):

The most important part of any GUI is how productive, intuitive and simple it is for the user. If it has complex add-ons, buttons and functionalities, it would disrupt the user's productivity and would probably be replaced by a better one. The User's experience also depends on other factors like the Color Themes available, customization options and ease of learning the structure of the Interface. These days applications and browsers tend to focus a lot on the UX because that is what draws in the user to use the application more because there are always better and more efficient solutions available in the market.

## GUI PROGRAMMING

At this point, I'm sure we're all aware that GUIs are completely programmed through some programming language and it takes some appreciable effort to create a good UI that allows a good UX. These days, many languages can be used to create one but some of them are best suited for easily updating, adapting and programming functionality. The most used ones are C, C++, JAVA and Python. These languages are at the top of the chain because of the extensive libraries that are present in them support for multiple devices and most important of all memory efficiency. In recent years a new programming language called DART, whose framework called Flutter (developed by Google) has emerged as one of the best ones available for cross-platform GUI development.

Moreover, languages such as KOTLIN, and SWIFT have also been developed specifically for Android and IOS mobile environment development respectively. Many of these languages mentioned above are not specifically developed for GUI development but have one or more Libraries/packages or frameworks that allow us to do so. Keeping this in mind, it's also worth mentioning that there is not one particular language that is considered the best overall for GUI development because it depends on the user's experience and choice.

Since Python is the most popular programming language in 2023, we shall be discussing some of the packages present in the language and learn about one of the modules in brief.

## SOLUTIONS OFFERED BY PYTHON

The Python language offers many packages through which we can create good-looking GUIs with ease and efficiency. Most of these packages are not made by Python officials but by other users, since it's an open-source language and allows distribution. Some of the important ones are, Tkinter, PyQt5, Kivy, wxPython, and PyGUI. The difference between these packages is that the functions and methods available change grossly and require special expertise in each one of them. In short, the only thing common with these packages is the Python language, its rules and syntax.

Again, any one of these packages can be used along with Python as it purely depends on the developer's choice and experience with the package. Some of these packages such as PyQt5 are free to all for public access and development but require a license for distribution of a product using the package. It's a small price we pay as an appreciation of the ingenuity of the creators and gratitude for the ease of development. Further details regarding the packages are available in their documentation (Check them out in the References of this manuscript).

I prefer the Tkinter package because it's very easy to learn and can accomplish almost any GUI problem that can be addressed. It has many widgets that are easy to code and is very light on memory making it fast and highly efficient for small-scale programs. It also contains a special modern 'tk' library that has more features and capabilities than the Tkinter package. But if you want to create an application to release it to the world, heavy packages such as PyQt5 and Kivy are your best solutions because they provide extensive detailing of every UI element.

## STEPS TO FOLLOW

To create a good GUI that appeals to the user and also delivers a good overall experience, we need to understand what the user wants from an interface and deliver it to them. Every user loves to have an efficient, compact and customizable environment. Therefore, we must consider mapping out the Interface first including what goes on from the first to the last second that the user will see. This helps us to get an estimation of the overall coding procedure required to complete the application's GUI.

This process can be done in many ways through symbolic representation, mental maps or to draw and map out the individual pages of the interface using a graphical tool or on paper as well. This is a perfect way to visualize what you want the user to see and improvise on things such as placement of buttons, text fields etc. This method is called 'UI Diagram' and provides us with a clear structure to refer to later on in the process of coding the Interface without forgetting a great idea!

The next step of the Interface creation process is a fun and experimental one. It is that of selecting the correct colour theme. It seems like a very trivial process but is one of the most important psychological tactics that developers use. For example, imagine if you would use Google if the background was green and the text colour was brown. It's hard to read and work on and you cannot take the application seriously. So, we must choose wisely to attract more users to use our application.

As an extension of the previous idea, choosing the perfect font and its colour goes a long way in the creation of the Interface. It's because an Interface is not just about where what goes but also how well it blends and feels natural. It's important to remember that the User's experience is also an important part of the Interface as mentioned previously. Hence, choosing the colour theme, font and their attributes are important. So, I have included some good websites that have very attractive colour themes and fonts on the references page.

When you are creating an interface for a particular operating system or a cross-platform application, you will be dealing with different screen sizes, and hardware and the process gets difficult and we must optimize it for every device available in the market. Fortunately, tackling the hardware part is easy if a minimum requirement is not satisfied then the application cannot run on the device. However, the GUI will change for every screen size, from the layout to the size of the components (such as a button or a taskbar) of the interface. This can also be tackled but it's not easy.

The above-mentioned method requires some mathematics dealing with the percentage of space used on the screen rather than using pixels as a measure of the size of the component on the screen. This allows us to reduce complications and implement the same for a variety of devices with one display ratio. For displays that don't fall under any of the standard display ratio categories, the nearest lower ratio can be used to display the Interface to reduce problems and for a better view of the interface. Moreover, the contents of the Interface must be symmetrical on the screen to provide easy accessibility and to yet again improve the productivity of the user.

It should be remembered that we must create the Interface according to some conventional unwritten rules that almost all users are used to. These are mostly commonsensical and are preferred to be set that way. For example, the Menu bar of an application can either be horizontal at the top or bottom spanning throughout the display or it can be vertical and must be accessed by the click of the button. Since almost all interfaces use this pattern, it would be difficult for the user to get used to it if you deviate from it.

### WHY USE TKINTER?

Currently, in the Python language, there are more than 100 packages for GUI development. But one of the most basic, functionally efficient and easy-to-use packages is Tkinter. The beauty of the package is that it's completely built into Python and you can start using it by a simple import statement. This is an interesting point because some packages such as Kiwi require a specific version of Python to be present and the installation process is prone to errors and mishaps for a beginner.

As mentioned before, it's lightweight (meaning it's easy for the machine to handle), has many functions and is easily customizable down to the font, font colour, and background, using images for stuff like transparency, size of window, multiple windows, themed icons, usage of multiple widgets (to be explained later) and their built-in functionality. It also offers much more than that in the form of simple syntax, integration between components in the form of functions and multiple placement options for the components. The best part is that the Tkinter package is still under development by the community and officially supports any version of Python. The additional subsidiary library included with the Tkinter called 'tk' offers even more possibilities of modification and manipulation to develop complex models which is in addition to adding many new widgets and components to this package which makes the development journey a fun-filled and amazing one.

With that being said, the next section contains the very basics of the package to help us understand the different levels present within it.

### BASICS OF TKINTER

To start working on GUIs using Tkinter, we need to first understand some basic instances of the package itself. These are windows, widgets, Placement, arguments and mainloop.

#### WINDOW

An interface for an application exists solely on a placeholder that we see on the screen and can be defined using code to change its colour, size and shape. Tkinter is flexible enough to provide us with a window with a few lines of code and certain methods to change the attributes of the window. For example, the new tab button, bookmark bar, and website icons all exist on a separate window which opens up when the Chrome application is started. Some of the attributes that can be customized according to our needs are Transparency, size on screen, default placement and background colour. It has to be created with a special function from the package and an identifier has to be linked to it.

#### WIDGET

After creating an instance of the window which can be visualized as a basket, one must start putting things in it to use properly. These things are called widgets and give meaning to the Interface and can be modified according to our needs. Tkinter has many widgets and allows us to use any number of widgets on a single window. Some of the most used widgets are,

- Labels – used to display text on the window
- Buttons – used to incorporate click functionality to initiate performing some task
- Entry boxes – allow the user to input text into a field
- Frame – creates a specific demarcation on the window which can be customized separately
- And many more such as Radio buttons, Check buttons, and Menu buttons.

## PLACEMENT

After the creation of a widget, the natural thing one must do is to place it on the screen. If not, the widget isn't displayed and cannot be seen on the window. The placement methods also contain options for padding and spanning of widgets. This is done using 3 methods which are Pack, Grid and Place. Depending on your needs you can choose any one of the following. Pack adjusts the screen size to accommodate the widgets directly from top to bottom unless specified. Grid allows us to place the widgets using the row and column method which needs to be specified. Place allows the user to specify the height and width that needs to be left from the top left corner to set a widget onto a screen but also contains other features.

## ARGUMENT

The above definitions and explanations only tell us how to create the widgets but don't necessarily specify how to modify them according to our will. This is done through specifying arguments. If you are familiar with Python, then this would be familiar to you as well. Some of the arguments include a window to display in, content, font family, size & colour and commands (for buttons). How these are specified is very similar to how one would pass in arguments in a function call using Python. This will allow us to modify the widgets according to our wish. It is worth recalling that passing arguments in Python are of 3 types – positional, default and keyword.

## MAINLOOP

Every Interface that is created has to be supported on a loop that monitors the activity going on in the interface. This can be understood in the following manner. Let's say that an interface contains a button and upon clicking on it a message is displayed. If a loop is not present which monitors the window activity, then when the button is clicked the message will not be displayed as the program is unaware of the click. Hence, a loop (called the mainloop in tkinter) is very much necessary for a responsive and dynamic interface. Again, the implementation is done through an attribute of the window present in the Tkinter package. With all this knowledge equipped, we next look at how widgets can be integrated with Python to develop some complex functionality.



Fig 2: Basics of Tkinter

## TKINTER-PYTHON INTEGRATION

I'm sure that the majority of the readers are confused with the title of this section. What it means is that the widgets in tkinter when created are not of any of the core datatypes in Python but are objects of one of the classes available in the tkinter package and when tried to perform some function such as addition, subtraction or any other ones, results in an error. This is because the object instance in itself is not compatible with the basic operations. We need to make sure to get values from the widget using Tkinter's special methods. This is explained further.

Let's say we have a basic text entry box. If a user enters something in it, then it isn't directly available to the program to read unless requested through Tkinter. To get the value entered by the user we need to use a special Tkinter function (usually called a method as it is associated with a certain class) to get what was entered

at the time of request. This is preferred to be done using a button which allows clean transitions i.e., the user consciously clicks the button after entering into the Box.

The above was one example out of many others. To do this for multiple widgets we need to understand how Tkinter originally was written to a minimum extent. This means knowing all the functions (most of the common ones) and what the datatype returned by such methods to utilize them to a maximum extent. To do that one must refer to the original Tkinter Python documentation (See References) and go through the functions and possibilities of some of the widgets as per our choice.

One can easily look up what value or datatype is being returned by the particular method, assign it to a variable, do necessary functions on it and display it back to the interface if needed. The part where you can display back to the running interface is quite tricky. That is because it must be determined while coding and must be accommodated early on before running. This is done using the Configure method and is a very simple process. It can edit an ongoing widget and the changes are seen live on screen (this is also possible because of the mainloop running in the background).

This is what integration truly means i.e., what was done by a user in the Interface is reflected in the internal program and can be used according to our needs and manipulated in many ways to change another or used in a different part of the program or Interface. Once we get accustomed to the process it all becomes a matter of time and code. It is also worth mentioning that one can always look up things such as functions or solutions on the internet. In the next section, we will take a look at, how a young developer can improve their GUI development skills.

### CREATING BETTER INTERFACES

Now that we've understood one of the basic ways to create Interfaces, we look at how one can design and create new intuitive interfaces. The process of creating something new usually involves some level of creativity, comprehensibility and ingenuity. Hence one must understand and let the flow of creative thinking direct them toward the interface creation process. The meaning of this is that anything and everything that comes to mind must be tried out on paper first to see if it would suffice our goals for the interface.

The next important thing to do is check the documentation of the package to know every possibility that can be used to arrive at the solution that we need for the interface. This also allows us to check if there are new additions to the package that might be useful to us. Moreover, doing so generally increases the chances of making mistakes and increases productivity since we learn something every time, we look at the documentation. This goes for any package or module and is a learning process that many of us use.

Equipped with knowledge about the tkinter package now one must experiment with the functions and methods present in the module itself and create new and better versions of the same idea. You cannot plant crops without getting your hands dirty. By doing this, one realizes different ways to approach a problem and learns how to solve issues and fix bugs efficiently. One must also experiment with all the functions available at least once to understand the influence of that function on the interface. By doing this we are completely aware of what happens in and out of our program and can create intuitive designs that require some level of skill.

Another one of the methods is to create a UI diagram. Not only do they serve as a blueprint for what we are about to create but take a load off of our minds and allow creative energy to flow through us. It also serves as a constant reminder to finish our work and most importantly provides structure and clarity about what is to be built. This allows us to finish the work a lot faster while preventing us from overthinking the design and forgetting good ideas as things get hectic when working on giant projects that require months to complete. Hence, everyone who is building a GUI must understand the value of this method and implement it.

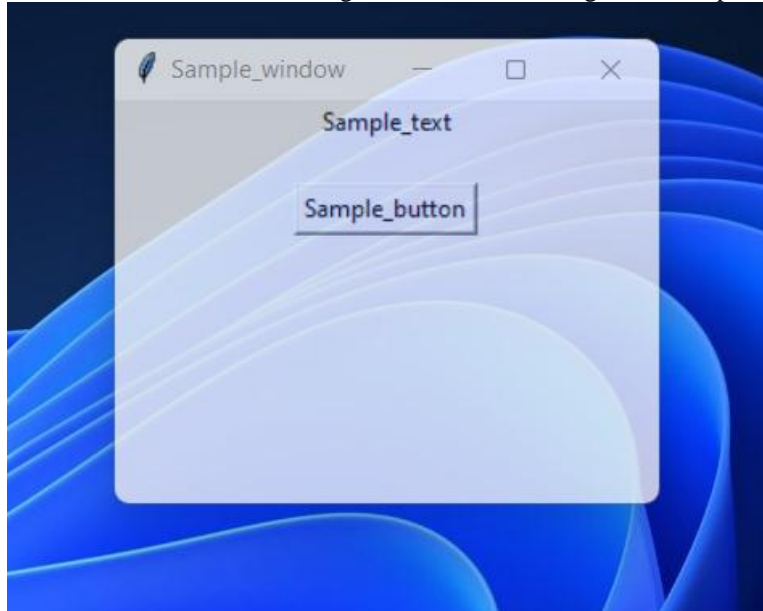
In the next section, we take a look at some intermediate and advanced Tkinter methods that come in handy one day.

### MORE FROM TKINTER

Now that the basics of GUI development and Tkinter are done, we move ahead to some other things you can do with Tkinter. Keep in mind that these are just some of the useful ones present in tkinter and there are a ton of other cool things that can be done using tkinter. It is also noteworthy to mention that some of these functionalities are from the popular subsidiary of Tkinter known as the 'tk' library.

Several widgets can be grouped together and individually dealt with using another widget called 'FRAME'. This allows us to put a few widgets of our liking into it and paste it onto the window. For most basic Interfaces, a frame is unnecessary. But for advanced ones, frames play a major role in their development as it makes the process and incorporating the functionality a matter of a few lines of code. For example, a taskbar needs to contain a few icons which in turn have their functions. To hide the taskbar for full screen normally we would need to hide every icon widget present on the taskbar. But if frames were used, only the frame needs to be hidden and all the associated widgets within it follow it.

Nowadays all interfaces have scrollbars. They are very useful for navigation through the page being shown on the interface such as a website on a browser interface. This provides flexibility and allows us to expand our pages without having to change the size of the Interface manually every time. The process of including a scrollbar and mouse scroll functionality is difficult in the sense that it involves some complex functions and some mathematics but is worth having on the Interface as it gives a neat presentable look.



**Fig 3: Transparent windows in Tkinter.**

If you want to give your Interface a futuristic look then you should play around with the transparency methods offered by Tkinter. It increases or decreases the opaqueness of the window allowing us to see behind the screen and the stuff on the screen as well. It's worth incorporating it into the Interface you're building. Moreover, there are options to include colors to it as well and the result is just amazing to look at.

The window created can be customized in many ways as previously mentioned. Some of the things that can be done are changing the size of the window which is very simple and very useful too. This is usually used to make pop-up alert boxes with just an 'OK' button. The window background colour can also be changed quite easily by mentioning the colour name or the hex value of the colour. The window is every bit customizable in tkinter down to the title of the window and Transparency as well. If you use Tkinter themes then you can change the color of the window's bar as well and customize the icons too.

But it doesn't stop here. There are still many if not a lot undiscovered widgets and functionality that you can discover from the documentation of the Tkinter package. Another library called 'tix' is referred to as 'Tkinter.tix' contains many more widgets, animations etc. The possibilities are endless when it comes to any designable or customizable thing. So, anything that anyone does is purely original and of their collective thought. Now that you are well equipped with information about GUI development, use the package to start building an amazing Interface.

## II. CONCLUSIONS

Today's creators are very ingenuine, especially the upcoming generations. So, to stand out from the rest we must be equipped with knowledge in various fields of development and starting with Interface development is as good a choice as any other. Interfaces aren't going anywhere for a few more decades until everything can be accessed through our thought process. Therefore, it is a really good idea to learn and master GUI development using any one of the languages and any one of the supported packages as it might come in handy sometime in the future.

I hope that all the concepts that were explained in this manuscript were useful and acted as a source of inspiration. So, learn well and build awesome things.

## REFERENCES

Package documentation:

- Tkinter - <https://docs.python.org/3/library/tk.html>
- Kivy - <https://kivy.org/doc/stable/>
- Wxpython - <https://wxpython.org/pages/documentation/index.html>
- Pygui - [https://www.csse.canterbury.ac.nz/greg.ewing/python\\_gui/version/Doc/index.html](https://www.csse.canterbury.ac.nz/greg.ewing/python_gui/version/Doc/index.html)

Tkinter:

- [https://www.tutorialspoint.com/python/python\\_gui\\_programming.htm](https://www.tutorialspoint.com/python/python_gui_programming.htm)
- <https://www.geeksforgeeks.org/>
- <https://docs.python.org/3/library/tkinter.ttk.html#module-tkinter.ttk>
- <https://docs.python.org/3/library/tkinter.tix.html#module-tkinter.tix>

History of GUIs:

- <https://www.britannica.com/technology/graphical-user-interface>

Wikipedia

- [https://en.wikipedia.org/wiki/History\\_of\\_the\\_graphical\\_user\\_interface#Augmentation\\_of\\_Human\\_Intellect\\_\(NLS\)](https://en.wikipedia.org/wiki/History_of_the_graphical_user_interface#Augmentation_of_Human_Intellect_(NLS))

Fonts and colour theme:

- <https://colorhunt.co/>
- <https://www.fontspace.com/>