1.1. GUI Fundamentals

Graphical User Interface (GUI) is a Human-Machine Interface used for interaction between humans and computers.

- The GUI consists of two basic parts: input devices and an application window for output.
  - The basic computer input devices are keyboard and mouse, but there are quite a number of other input devices as well, such as touchscreens you can find on tablets and mobiles.
- The application window is an image (in other words, graphics) displayed on a computer screen.
  - Using the input devices, the user manipulates different parts of the application window.
  - This interaction is made possible by the operating system which works seamlessly in the background. In many cases, the operating system provides the basic buildingblocks that get used to build the application window.

In order to create a GUI, programmers use some kind of **application programming interface (API)** provided by the language library they are using. If they work directly with the underlying windowing system provided by the operating system, this kind of programming is called native window programming.

1.2. Basic Principles of Good Interaction Design

- **Clear Mental Model:** The user should understand the overall purpose of the software and how the individual components come together. How the software operates, on the level of user interaction, should make sense.
- **Reassuring Feedback:** The user should know if they've done something successfully; the software should communicate on some level. For instance, when the user clicks on a button, the button should change its graphics to indicate it's been clicked on.
- **Navigability:** The user should know where they are in the system. They should have a clear idea what they can do there, where they can go next, and how they can get back.
- **Consistency:** Every command with the same name should act the same way everywhere in the software. For instance, selecting "Copy" from the Edit menu should work exactly the same way as selecting "copy" from a right-click pop-up menu.

• **Intuitive Interaction:** This sounds a lot like the clear mental model listed above, in a sense.
A user should have a good sense of how the software behaves, without having to invest a
great deal of thought. This is like the “clear mental model”, but on a smaller, more immediate
level. The purpose of each button, each menu item, should be easy.
• **Behavior:** Or in other words, performance. There are potentially many solutions to a given
coding problem, but ones that don’t work well may leave the mouse moving sluggishly, or
button clicks less responsive.
• **Appropriateness:** The UI, and its controls, should be relevant to the task at hand. A
firstperson shooter game requires a different set of interactions and controls than a spreadsheet,
and imposing a clumsy pull-down menu in the middle of an online firefight is going to result in a
substandard experience for the users.
• And finally,
Importance: There are a variety of ways a UI element can be presented to the
user, and how it reacts to input. It might be just text, or have an icon, animations, or even
sound. Features with important consequences should use many of those elements, such as
the Recycle Bin. The Recycle Bin is graphical, animated, has a text label and plays sounds,
and deleting files permanently from one’s computer is much more important than whether a bit
of text is bold.