Postgraduate Programme of Studies

Interaction Generative Design (I.G.D.)

Hellenic Open University

Title of Dissertation:

Human-Machine Interaction - Interactive Projection with Live Input from Moving Subject

Student's name, AM: Marina Litsiou, std523987

Supervisor's name: Anna Laskari

Thessaloniki, Greece, June 2024

“Human-Machine Interaction - Interactive Projection with Live Input from Moving Subject”

Marina Litsiou

Supervising Committee

Supervisor*:* Co-Supervisor:

Anna Laskari Nikolaos Kourniatis

Thessaloniki, Greece, June 2024

"To my mom and dad, who support me and all my crazy ideas, and to my friends, who suffered with me through my many mental breakdowns while finishing this dissertation"

Abstract

Human Machine Interaction (HMI), or the more correct term for our day and age, Human-Computer Interaction (HCI) is a rapidly evolving field, driven by the pervasive presence of technology in everyday life. Today, nearly every modern device, regardless of its complexity, is embedded with programmable microprocessors, emphasizing the importance of understanding, and optimizing the interaction between humans and computers. But where did this need for researching this interaction come from and how did it come to be, what we know today? This paper investigates the historical context and evolution of HCI, highlighting key principles that rule this field and some of the basic tools that we use, like the "Grid of Affordances" by dr. Murray. The discipline of HCI is dedicated to analyzing human interactions with machines to develop designs that are more intuitive and user-friendly. A significant area within HCI is emotional design, which seeks to understand the emotional responses elicited by different design elements. This research includes examining anthropomorphic designs, such as robots and prosthetics, where historical perspectives, including Dr. Mori's work, reveal insights into human perception and preference. In addition to emotional design, the paper explores computer vision, which is the capability of computers to interpret visual data. This technology holds considerable promise across various fields due to its ability to identify complex patterns and provide valuable insights in diverse applications. The final section of this paper introduces an application that embodies the principles discussed, showcasing the practical application of HCI research.

Keywords

Human-Computer Interaction, Human-Machine Interaction, Computer Vision, Emotional Design, Digital Medium, Processing

"Διάδραση Ανθρώπου και Μηχανής – Διαδραστική Προβολή με Ζωντανά Δεδομένα από Κινούμενο Θέμα"

Μαρίνα Λίτσιου

Περίληψη

Η διάδραση ανθρώπου και μηχανής, ή καλύτερα η διάδραση ανθρώπου και υπολογιστή είναι ένα σύγχρονο θέμα και ένα πεδίο έρευνας που μας απασχολεί αρκετά τελευταία με τους ρυθμούς που η τεχνολογία προχωράει και αναπτύσσεται καθημερινά. Ειδικά η διάδραση του ανθρώπου με τον υπολογιστή, εφόσον τα πάντα πλέον μπορούν να θεωρηθούν ως υπολογιστές, καθώς φέρουν μικροεπεξεργαστές που μπορούν να προγραμματιστούν. Αλλά από που ξεκίνησε η ανάγκη για έρευνα πάνω σε αυτό το πεδίο και πως εξελίχθηκε για να φτάσει στην μορφή που γνωρίζουμε σήμερα; Ποιες είναι κάποιες από τις αρχές και κάποια από τα βασικά εργαλεία που χρησιμοποιούμε, όπως για παράδειγμα το «Grid of Affordances» της Murray το οποίο και αναλύεται σε αυτή την διπλωματική; Υπάρχει ένα ολόκληρο πεδίο που μελετάει την σχέση μεταξύ ανθρώπου και υπολογιστή, καθώς και ανθρώπου και μηχανής γενικά, αναλύει τα δεδομένα και τα χρησιμοποιεί για να δημιουργήσει καλύτερα σχέδια, που θα είναι πιο εύκολα αποδεκτά και κατανοήσιμα από τους ανθρώπους. Αλλά γιατί ένα σχέδιο μας προκαλεί χαρά, ενώ ένα άλλο μας κάνει να νιώθουμε άβολα; Το emotional design είναι ένας κλάδος ο οποίος ήρθε να απαντήσει πολλά από αυτά τα ερωτήματα. Μαζί με αυτόν, γίνεται ανάλυση άλλων τριών προσεγγίσεων στον σχεδιασμό, που έρχονται να αναλύσουν την ανάμειξη των συναισθημάτων στον σχεδιασμό.

Ο σχεδιασμός, ειδικά ανθρωπόμορφων μορφών, όπως τα ρομπότ ή τα προσθετικά μέλη, ο οποίος έχει επίσης μελετηθεί αρκετά, φέρνει στο φως νέους προβληματισμούς για το πως οι άνθρωποι αντιλαμβάνονται τα πράγματα και τι τους κάνει να έχουν αρέσκεια προς κάτι. Αυτοί οι προβληματισμοί, έχουν μελετηθεί σε μεγάλο βαθμό από τον Mori και αναλύονται στην παρούσα διπλωματική. Μια ακόμα θεματική που αναλύεται, είναι το «Computer Vision», ο τρόπος δηλαδή με τον οποίο οι υπολογιστές «βλέπουν» και κατανοούν τον κόσμο, το οποίο είναι ένα πάρα πολύ ενδιαφέρον θέμα, που βρίσκει εφαρμογές σε πάρα πολλά επιστημονικά πεδία, καθώς οι υπολογιστές έχουν την υπολογιστική δύναμη να αναγνωρίζουν σύνθετα μοτίβα και μπορούν να βοηθήσουν τους ανθρώπους σε διάφορες δραστηριότητές τους. Αλλά σε ποιον βαθμό και με πόση επιτυχία; Στο τελευταίο μέρος της διπλωματικής αναπτύσσεται μια εφαρμογή, η οποία χρησιμοποιεί τα βασικά μέρη αυτής της έρευνας που πραγματοποιήθηκε στα πλαίσια της εργασίας.

Λέξεις – Κλειδιά

Διάδραση Ανθρώπου Μηχανής, Διάδραση Ανθρώπου Υπολογιστή, Υπολογιστική Όραση, Processing, Πλέγμα των Δυνατοτήτων

Table of Contents

List of figures

1. Introduction to HMI (Human Machine Interaction)

Let's start by trying to understand more about the term Human Machine Interaction. What is it, how did we come to speak about interaction between humans and machines and why do we need it? In the context of this dissertation the term Human Machine Interaction will be replaced by the abbreviation HMI, which has been used by previous researchers mostly for the term Human Machine Interfaces.

Figure 1. Sun Dial or the first watches that humans used to track the passage of time

To properly explain HMI, an example is needed and what better than the one in the introduction section of Guy A. Boy's book, The Handbook of Human-Machine Interaction: A Human-Centered Design Approach (2011), where he gives an example using the global knowledge of what a watch is and their long history, to justify his choice and present his arguments. Watches have existed since ancient times, in the form of sun dials that didn't even include mechanical parts, while the first ones which contained mechanical parts, which started appearing in the 1500s. The watch's main purpose, which hasn't changed since ancient times, is to tell us what time it currently is. A modern watch uses many parts to do that (taking it a step further, a smart watch uses even more), offering also the possibility for the user to adjust the time in case of a time saving, a time zone change or when it stops because the battery ran out; subsequently the battery can be replaced (or, in the case of smartwatches, be recharged). All these parts work, so the one wearing it can see the time. The user, when they look at it, is basically asking "What time is it?" and the watch is responding. This is an automation that

can be used in many aspects of our lives. Automations like these are what caused the term Human Machine Automation to appear.

However, despite automations and the need for the research — which is especially evident in our day and age, when computerized machines are heavily used in our everyday life — and study of these interactions, the field of HMI is quite a new one. Computers brought many changes in the way we see HMI and revolutionized the studies in this domain and how we perceive it. According to Boy, "four main principles, that is, safety, performance, comfort and aesthetics, drive this rationalization along four human factors lines of investigation: physical (that is, physiological and bio-mechanical), cognitive, social or emotional" (Boy, 2011, p. 1).

What pushed this field forward was the creation of the personal computer, or how we like to call it for short, the PC. The first truly innovative computer that tried to take into account the interaction between humans and machines and make it easier, was the Apple II. Before that, microprocessors did exist, but they needed to be assembled by the user or were very hard to use. The Apple II had a color screen and a keyboard, things that we now take as a given for our interactions, but back then they were truly innovative. Let's not forget that the computer mouse and the basic GUI (graphical user interface), did not exist before the 1950s, when Douglas Engelbart invented them and patented them. Before that, interaction with computers was harder and required very specific skills and knowledge.

In our day and age, we have come to a point where we usually no longer speak about Human Machine Interaction (HMI), but about Human Computer Interaction (HCI), two similar terms that an example will be needed to better explain the difference between the terms. We can use the example of driving. Driving a car is a Human Machine Interaction. But

on newer models there are computers on them. They may even be computers themselves, like self-driving cars. So that should be considered as a Human Computer Interaction. With this example in mind, we can see that today, the two terms share similarities, and they can sometimes overlap, but not always.

When we think of Human Machine Interaction or Human Computer Interaction, our approach must be human centered. This is why Human Centered Design (HCD) exists. That means that everything we design and create must have the user in its mind and try to simplify the experience of using that machine. As Murray says in her book, "From a humanities perspective, the design of digital objects is a cultural practice like writing a book or making a film ....To see any artifact (i.e., any human-made object) as part of culture is to understand how it becomes meaningful through the social activities, thoughts, and actions of the people who engage with it." (Murray, 2019, p.2)

1.1 What is "interaction"?

What is the meaning of interaction? Well, according to the Merriam-Webster Dictionary the term interaction means "a mutual or reciprocal action or influence" and its first recorded usage with this exact meaning was in 1832. This term, depending on the context we use it and the field where it's used, can take slightly different meanings. But in most cases, it refers to an exchange, an influence or maybe an action that happens between two, or more, entities.

For example, in the realm of sciences like chemistry, an interaction can be between forces or some energies, or even particles that influence each other and exchange for example

energy when a chemical reaction happens between them. This is also an interaction, and the term is used to refer to those.

In sociology, interactions happen between two humans, and this is the subject that sociology is studying. There, the term can be used to describe the way two (or more) humans exchange ideas, emotions and how this exchange can affect each other and influence their behavior or their way of thought and even their personality traits. The term can include any type of exchange, not only spoken word. That means it also includes, for example, gestures and facial expressions.

Circling back to the subject of this paper, in the realm of Human-Computer Interaction, or simply HCI as we'll see later on, and in the field of user experience, or UX design (a field that is very important to the studies of Human Machine Interaction as it is influenced directly by the studies done on HCI), the term is used to describe the way that us humans communicate, exchange information and both influence and get influenced, by the machines and specifically, computers. Here we can include everything from the actions that a user performs, to the way that a specific interface responds and generally, the way that us humans, or better yet, us users use technology.

So, in conclusion, as we see in every example, the term "interaction" contains the fundamental meaning of an exchange, a mutual influence or simply of communication between two or more entities, whether these are humans, particles or a human with a machine. This "interaction" always has an outcome, which depending on the context could be shaping a relationship, an explosion or maybe the creation of a new interface.

1.2 A brief mention in the history of HMI

The subject of human-machine interaction is vast and we can get lost in it easily as nowadays it encompasses many fields under its umbrella, For this reason, it's important to look into its beginnings. First things first. When did that field start developing and becoming a field of study? HMI's description, according to M. Di Nardo, D. Forino and T. Murino in their article published in 2020 with the name "The evolution of man–machine interaction: the role of human in Industry 4.0 paradigm", Human Machine Interaction is any communication and synergy that happens between a human user and a machine in a dynamic and changing environment and usually through some kind of interface. This interaction and cooperation have existed ever since humans started building machines.

This interaction has not always been smooth sailing and is something that is constantly evolving. Back in the beginning, before World War 2, humans were trained to adjust to the technology of the time. The machines were not built with the user in mind, instead they were built with only their function in mind, with the thought that humans could be trained later on how to operate them and adjust to them. The trainings received by the operator of the machine were specific to the type of the machine, the constructor and the model, which eventually resulted to a limitation in system operation of similar machines and created the view that operating computational systems is time-consuming.

That all changed with WW2 and the need for fast production of parts for the war. Companies needed to have trained workers for their machines, fast. The thought behind the concept and design of the machines needed to change. Now the designers started to take into consideration the user and making the machines easier to operate and the need to understand their function became a priority, as the need for an analysis of that interaction with the machines became apparent.

In the article by M. Di Nardo, D. Forino and T. Murino, they separate the studies on HMI into four different eras. The first one, from 1940 to 1955, where designers and developers were trying to find the limit of the human capabilities to understand how to interact and use a machine. New machines, equipment and accessories for operators were designed, in an effort to enhance the productivity of the user. Then came the second era, from 1955 to 1970, where we truly see things move forward and advance. Designers actually start to try and model things targeted a bit more towards humans. Around 1970, it was an interesting time, as electronics started rapidly advancing, introducing new things daily to the world, bringing advancements to every facet of life. The third era, from 1970 to 1985, machines were used to automate many tasks, tasks that usually required humans to be done normally. That highlighted the necessity of designing everything, with the human user in mind and the interaction that would occur between them and the machines. The human now was not only the machine's user, but also its supervisor. The last era is from 1985 to our day and age, where the advancements in the HMI happen every day. More and more, this interaction is studied and is pushed forward by technological progress and the need to have an easier way to communicate with the machines as their design become more complex and the tasks that they can accomplish are growing in number.

So after all this, we can see that the field of HMI has changed drastically over the years and is constantly evolving in a rapid way. A thing that has pushed it truly forward was the fourth industrial revolution and some of the pillars that this revolution brought with it, like the field of big data and analytics, robot-assisted production, self-driving vehicles, augmented reality and additive manufacturing. All of this has now made us talk not only about HMI, but also Human Computer Interaction, or simply HCI. It has spurred many fields of study on the subject and more people take an interest in it, with new research being conducted constantly and discoveries made on the improvement of the experience of the user, as we are now talking about human centered design, which will be covered later on.

1.3 A brief mention in Human Computer Interaction

Today, we can't talk about HMI without talking about Human-Computer Interaction, or HCI for short. We are in an age where we can't separate computers from machines. Let's take our watch example from before. We were talking about an interaction with a machine, back in the day where we could see the time and maybe the date in some of the more expensive ones. But nowadays, most people buy a "smart watch". This transcends the use of a stereotypical watch, as a watch now comes with a microprocessor on it and can do so much more than just show us the time. Smart watches can now act as heart monitors, as activity trackers and even accept calls and messages, reminding us of the watches that spies used to wear in old movies. So, what do we call these? Machines or computers? And in which field do we categorize them under?

And this question can now be applied to all the new machines, as they all come with microprocessors and interfaces on them. Even fridges and kitchens now have smart touch screens and even connectivity to the internet. After all this, we can see that we can't speak about HMI without HCI and not include it in this dissertation.

Human Computer Interaction is a field of study that contains many fields under its umbrella. Its main focus is the design, evaluation and implementation of interactive computing systems adapted for the human user and the study of anything that occurs around this interaction. The beginning of the history of HCI can be traced back to those early primitive days of computers, when you needed an entire room to store one and they were complex enough, that they could be operated only by experienced technicians in the field of computers. As computers rapidly evolved from being used only for military and scientific reasons and started becoming more accessible to the public, the importance of an intuitive

and user-friendly interface started becoming apparent with many scientists dedicating their studies on how to create them.

One of the most important studies and publications from those early computer days on HCI is "The Design of Everyday Things" that was published in 1988, by Donald Norman. In that book, which has nothing to do with digital design but studies industrial design, Norman takes a deep look and explores user-centered design. That research was so important for design that it laid the foundation of the whole modern field of HCI practices, despite not mentioning anything about computers. He suggested and created terms and concepts, that even though now we think are common knowledge, back then were revolutionary. He introduced concepts like affordances, constraints and mapping, concepts that are now the foundation of understanding how users are interacting now with technology.

Another scientist that really helped this field advance, was Ben Shneiderman with his book "Designing the User Interface: Strategies for Effective Human-Computer Interaction" that was originally published in 1987 but has since then seen many newer and updated editions with the most recent in 2022. Shneiderman has highlighted the importance for always keeping the user in mind when designing interfaces, making them enjoyable for the user, adaptable and easy to learn, while at the same time keeping them efficient.

As we can see by those two studies, the 80s were an era that really pushed the field of interaction forward. The introduction of graphical user interfaces, or simply GUI, made the computer easily operated by anyone, and made coding knowledge optional instead of neccery. This made the importance of user-friendly design so much more important. The creation and release to the general public of the Apple Macintosh by Apple and the Windows by Microsoft was a key point in the history of interaction and marked a very important

moment in the history of HCI. Those two replaced the command line interfaces, which were hard and required specialized knowledge, and brought in visual elements that were familiar from the real world (affordances will be further analyzed in a later chapter). Elements like icons, windows and menus, things we are now familiar with, made the computers easily understood and operated by anyone.

A good example of design of that era, is the introduction of the "messy desktop". The computer desktop is designed to mirror the real-world desktop of the user, where you sort all your papers and files into a folder. But you never get rid of everything. You always have loose papers and notes around, like you do with some files that you don't know where to store on your computer. See how the digital world mirrors the real one? That was a great example of a user centered design, as it was easy to explain to someone. But more analyses on this will follow in later chapters.

But to get back on track. The research on HCI continued and really blossomed in the 90s and 00s, as the field started expanding to include other fields under it, like usability testing, user experience, or as it is commonly known, UX, and all the new human centered methodologies that are connected with work done in the field of cognitive science and psychology principles and so many other fields of study.

Reaching our day and age, the field has expanded so much that it's connected with many other fields and sciences, some of which are, obviously, computer science, psychology, design, sociology and many more. A problem though, that has been presented is that technology moves too fast. That shouldn't technically be a problem, because experts try to find and offer solutions to the public, but when the solutions are coming too fast and the user is unable to familiarize themselves with the new technology, they start to get overwhelmed

and not utilizing the full extent of all the new things that are offered to them. As the design of digital artifacts is something quite new, especially compared to other fields, like for example the printing of books which dates back to over half a millennia and is something that has been done and studied extensively, the field of digital design, despite its advances, is still very much dependent on the technique of trial and error. That isn't necessarily bad, but it can be time consuming. With the incorporation of computers into every facet of our life, the study of all these aspects of HCI is more important than ever. It's a field that is constantly evolving to bring solutions to the human users' needs.

1.4 Goals and questions that will (hopefully) be answered

Here starts the fun part. After we saw briefly what HMI is and what HCI is and a small part of their history, it's time to analyze the main objectives of this dissertation. After finishing the classes of this master's degree and seeing many facets of interactive design, I've come to really appreciate the studies that are happening on the interaction of humans, machines and computers. There are many questions that can be set and analyzed, and this field is so vast, that I will try to not stray to too many studies, but rather focus on some questions that I am interested in answering and to emphasize on and do the more extensive studies and research on my own outside the confinement of this dissertation.

As this thesis has both a theoretical and a practical part, in the beginning, some of the questions that will be attempted to be answered by analyzing the texts of the scientists who have done extensive research, in combination with some of my findings on the practical part of this thesis will be presented. I will mostly focus on the interaction part, so a good starting point is to understand how does a user perceives and understand an interactive system and what are the cognitive mechanisms and the mechanisms that circle around the human perception of underlying user interactions with that system, focusing on the systems that take

live input from their users and use it to interact with them through projections. A very interesting question, that can be answered through the use of studies from the field of psychology and sociology is what is the role of emotions in interaction? Why does our brain like a specific digital artifact, but gets creeped out by another? What are the design principles and how can we make the design of the interaction easier both for the designer and the user? Is it easy to improve the artifact after we've created it? How can we reframe familiar things, so the user doesn't feel too alienated? On the subject of emotions, we'll see some of the approaches on them, analyzing some of the most famous theories that are proposed by scientists.

But let's not yet focus on that part, as it will be investigated further in later chapters. More interesting questions, at least to me, that I will try to answer are what are some of the design principles and guidelines that must be considered when creating an interactive system, while we aim for the optimization of user experience? And what role does cognitive theory play in shaping the interactions with the digital systems, especially in the dynamic and immersive environments that are analyzed here and as the one I am designing is?

For all these questions, an interactive live projection will be developed in order to help answer them and also for some of the more "practical" questions that need to be studied. The Processing environment will be used for this, as it is a great environment for the easy production of an interactive projection. Through testing that environment in many different settings, while having multiple users interacting and recording their experience, I'll try to discover which are the optimal settings and algorithms in the Processing environment for utilizing computer vision techniques, such as in our case, motion detection, to not only capture, but also process live input effectively, This is a very demanding process of the computer that is used for the processing of the live data that is captured. On that subject,

Figure 2 - The world of HMI and HCI is vast and very interesting.

we'll see some of the computational requirements and limitations that come from the computer, such as the CPU and the GPU limitations. Another hardware limitation will also probably be the camera. Can a variation in camera specifications, such as resolution, impact the performance and usability of the live input that is collected? I will be using the computer camera for cost reasons, but would using a Kinect camera for example, make the result better?

This subject is vast and for a curious person, like me, can expand in many different directions, as HMI is an integral part of many facets of our lives and has been interwoven in many different sciences. This thesis is an attempt to scratch the surface of this magical world, that has so many things to offer. So, let's start trying to put things in order.

2. New Technology, same principles

Starting our journey into the magical world of digital design we will first need to go over some basics of design and some terminologies. The best introduction in this field would be two books. One of them has not been written about digital design, but design in general and was the book we previously mentioned written by Donald Norman in 1988 and the other one is by Jannet H. Murray, INVENTING THE MEDIUM: Principles of Interaction Design as a Cultural Practice (2011). This is a great book to start, as Murray, takes a more humanistic stance, rather than a social science or industrial design stance to HCI. The key characteristic of this humanistic perspective is that it can be interpreted in a multitude of frameworks. This perspective is a very good one, as in design things are never black and white. There is no singular correct answer and problems that are encountered can be solved in a multitude of ways. Norman's book gives us some foundational design principles that are very easily transferred to digital design.

In the humanist perspective, the design of digital objects is as much a cultural act as writing a book or making a film. But what is culture? Why is someone cultured and someone uncultured? Well, culture can be interpreted as the infinite web of meaning that humans create. It can contain a very wide range of human behaviors, such as language (the most important one, as humans give meaning to words, in order to have a way of communication), religion, traditions, music, literature and many more facets of our everyday life. It is, down to its core, what shapes the way that humans see the world and express themselves. It is something that is "alive", like our language is, because it evolves over time, whenever different groups and humans interact.

Murray asserts three foundational design principles that will be used and followed throughout this dissertation. They won't be debated and will be taken for granted and as something that is confirmed. Those three foundational design principles are that: "All things made with electronic **bits** and computer **code** belong to a single new **medium**, the **digital medium**, with its own unique **affordances**. Designing any single artifact within this new medium is part of the broader collective effort of making meaning through the invention and refinement of digital media **conventions**".

Murray, also defines the four principles that describe this new digital medium. Its four core properties or as she mentions the, the four core **affordances**. These four are that the computer is encyclopedic, spatial, procedural and participatory. These four affordances will be later explored and explained further, but they are tools that can help us describe and develop any new digital medium. The things that we create may fall more on one of the four affordances, but the other three help us to develop our design further. They are important and need to be considered both at the beginning of a project, but also during its further development.

Having accepted these fundamentals of digital design, we can start by looking into and delving into some of the parts that form digital design, like further explaining the four affordances, complexities, and the cultural implications it entails, for starters. We will analyze more aspects of design further later on, but this is a good place to start and to set our basis for setting up for the design of the interactive interface that will be developed later.

2.1 Exploring the affordances of the new digital medium

The main concept of this chapter will be the concept of affordances. This term encapsulates what we recognize as the object's (or medium's) abilities and attributes, that are taken from its design phase. We are mostly focusing on the digital medium which is malleable and has a huge impact on HCI. This malleability of the digital medium allows it to adapt and transform depending on the user. The user "feeds" it with information through the click of a mouse or through the swiping of their fingers on a screen or even through voice commands, and the system responds accordingly to each user. This gives the users impressive control over the digital environment. This also can give designers a great challenge, as they are called to create artifacts that are not only functional, but they also need to be easily used by the users and respond in a way that won't make the user to want to throw their device out of the window.

As the digital medium is quite new, designers usually function with the "trial and error" technique. The designers that are very experienced know how to frame the new design problems that are coming their way through existing traditions, practices and goals. An obvious example, for the average customer, would be the fridge, a machine can be found in every home, which everyone knows how it works and know how to use it, so a designer can then focus on just refining the elements that are familiar, for example a door that turns translucent so we can see inside, or an ice cube making mechanism or better freezing capabilities. As in fridges, some parts of the digital media design involve similar refinement of artifacts that the user is already familiar with. Like for example the save icon being a floppy disk, or the print icon being an actual printer. The user knows what these buttons do, without needing any explanation.

Figure 3 A familiar design. Saving a file utilizes an artifact that humans know what it does

Figure 4 A printer icon does exactly what you imagine when pressed. Prints whatever you have selected.

The refinement can also happen to already existing digital artifacts. Computers are always being redesigned within our now familiar frames. Newer technology always pushes the design of digital artifacts forward. A laptop is still a laptop, but newer ones, thanks to the advancements of technology and the shrinking of their components, have now become smaller and lighter, having even better performance than they had before, because the newer ones are designed with the user's comfort in mind. Websites are redesigned constantly to keep up with the times, the new trends in the field of graphical design, the new requirements for new functions and visuals. One needs to look no further than for example a social network like Facebook, that has undergone many redesigns and rebrands in order to always keep up with the times and to utilize all the tools that that specific era offers, like for example advertising, but then it made it into targeted advertisement.

As we've briefly mentioned in the introduction of this chapter, Murray recognizes that when we are looking at computers as singular new mediums, we can clearly define their four core affordances. These four affordances state that the computer is encyclopedic, spatial, procedural and participatory. That basically means that whatever is created out of computer code is capable of encyclopedic capacity, of modeling navigable spaces, of describing and performing conditional behaviors and of supporting participation by users. When a designer designs something for that medium, they always utilize these four affordances. Maybe their design is leaning more towards one or two of the four, but the rest will also be used. They are what govern the design of HCI and help with making designing new things much easier, as they help us by pushing us towards answering some simple questions and that way, solve more intricate problems by breaking them down into smaller ones.

When designing artifacts for computers, it's also important to look into some things that are borrowed by the science that is knitted closer than any other with HCI. That is computer science. Murray states in her book that "Even for those who will not be doing the coding, understanding how computer science describes objects and processes is crucial to making sound design decisions". As someone with a computer science background, the most important process we are using while coding is abstraction. What that means is that we take a problem, and we strip it down to smaller problems and we try to create some representations of the elements of that problem in the simplest terms, that also contain all of their essence. We try to group functions, things and elements by their most fitting attributes for our problem. Like categorizing cars. Do we categorize them by make and model, by their price range or just by their color? Or maybe we categorize them by their function, like do they run on gas, electricity or are they hybrid? Well, that depends on what question we want to answer. If we want to answer the question 'Which car is the one owned by the most people?', we can categorize them using their make and model. If we want to answer "Which car is the best for environmental reasons?", then we can categorize them by their emissions.

The same way of thinking is applied in designing a digital medium. We can break down our problems into simpler problems and categorize everything by what we need them to do and create our own abstractions. The four affordances come in very handy for this reason. Here, we can also call upon Norman's book The Design of Everyday Things (1988), which has nothing to do with HCI, but is one of the most used books in HCI. It's a book that comes to us from the field of industrial design, but its principles and Norman's ideas are applicable to designing for the digital world and for creating good human-computer interactions.

Norman, who started as a cognitive psychologist, called our attention to the mental models that a user creates, based simply on the appearance and behavior of the object. Mental models are especially important to consider, since the biggest confusion and fear about the new digital medium is all its hidden mechanisms and the unfamiliarity with it. We not only need to know the outcome of what happens when, for example, we click something, but we also need to know why that outcome was chosen. Like when you listen to music on Spotify. Why is the next song chosen? Was it at random or because of your listening habits?

When designing with the human user in mind, we need to make the design as clear as possible and remove any distraction that may cause the user discomfort and distract him from his goal. This is achieved by making the elements on the interface easily understood and not overcomplicating them. We always need to consider that the users may have different levels of knowledge in the digital world, so each one will perceive it differently. This is why the field of UX (user experience) was created and is so important in our day. It provides much-needed feedback to the designer, so they can change things accordingly.

2.2 The grid of affordances

Through the aforementioned affordances, Murray has created the "Grid of Affordances". That grid is a very useful tool when designing digital artifacts, as it can help us both plan the project and then further develop it. There are two responsibilities for the designer to follow, according to her. The designer needs to design and create artifacts that have the best interest of the humans who will interact with them in mind and to strive to push the digital medium forward. The grid comes in place here, as in each of its squares it has one of the four affordances. If we as designers use these four affordances and the grid they form, it is very easy to maximize the expressiveness of the artifacts we create and we can learn to design better.

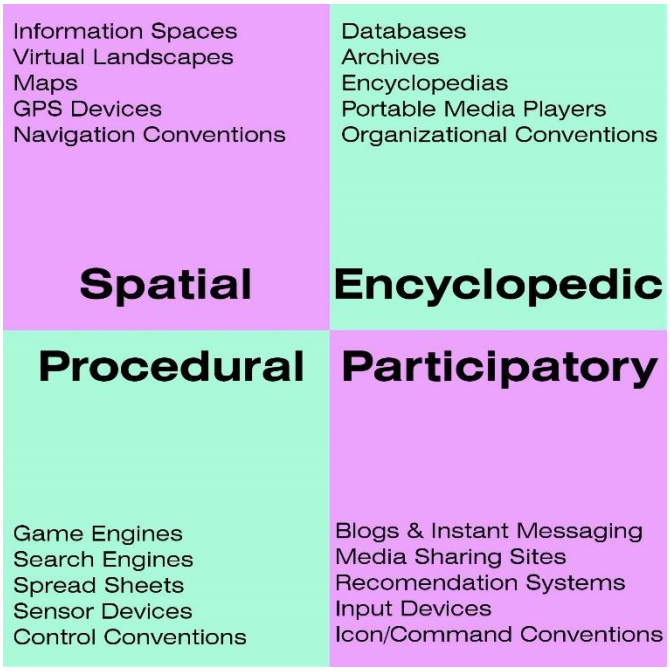

Figure 5 Murray's "Grid of Affordances".

But it's, also, necessary to mention that it is not mandatory for the artifact to fit every category of this grid. It doesn't need to contain all these technologies for it to be a good design. It can fit any amount and any percentage into those. These just help visualize the issue and simplify the design process or make development of the artifact easier. It's easier to see the artifact positioned on this grid and visualize its properties. Any good application, from even the first phases of development, is developed with a considerable amount of thought on what developments its growth will bring and how this development will maximize the four affordances.

Later on, the interactive application that will be developed in the context of this dissertation will be compared to this grid. But for now, we will take a look at an existing example, so we can better understand how this grid is used. A well-known example, as it is an application that is used by million users daily, is Spotify. The famous platform for streaming music. The company was founded in 2006 and the application was released in 2009. The platform was initially founded to combat the huge amount of piracy that existed in the early 2000's in the music industry. It gave the users a simple and very cheap way to listen to all their favorite songs. Later on, algorithms were introduced in the platform that only allowed the creation of custom playlists, but also created playlists for the user, based on their listening activities. It also introduced the ability to download your music and listen to it while offline. It even introduced at some point the ability to make friends on it and see what they are listening to, or even listen to music together, even if you are not in the same place. The latest feature that Spotify released, unfortunately not yet to Greece as of the time of writing this thesis, is the introduction of your own personal AI DJ. It takes the concept of a DJ, where you request songs, and tries to understand the "vibe" of the room in an effort to keep everyone happy, trying to mimic it with the use of AI. Spotify already knows the user's music taste, so introducing AI that can basically interact with them and make it understand their moods and what they want to hear in that specific emotional state they are in, was an easy

and a smart move, as it reframes a familiar concept and makes something new and innovative with it.

Spotify is a good example, as we see that this is a platform that started off as something but is constantly evolving, taking into account the current technology and the needs of its users and trying to incorporate them into solutions, always coming up with new ideas and trying stay ahead of its competitors. It utilizes most of the affordance grid, as it is Encyclopedic because of its huge catalogs of music, it is Participatory because you can make you can search and add your own music and make friends and listen to music together, it is Procedural due to its great search engines and to its use of AI and then we could argue that it is a little bit spatial as well, as it only shows you songs that it owns the copyrights for in your area, based on your device's GPS sensors. So, its current grid of affordances could look like the one below, where the size of its logo symbolizes the amount that it takes advantage of that affordance.

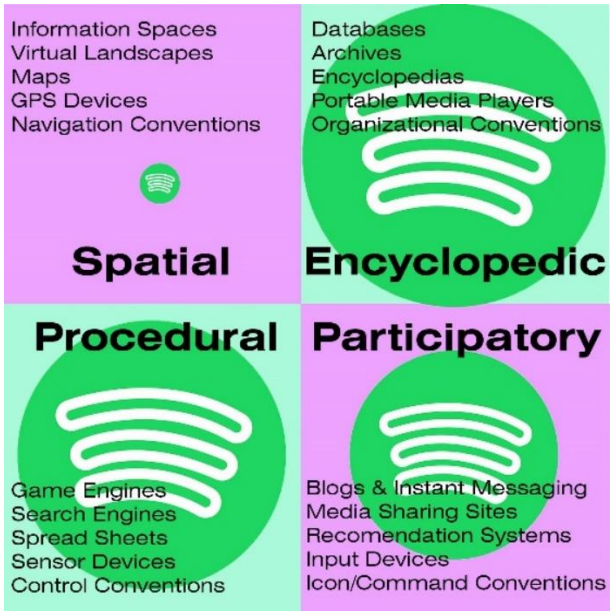

Figure 6 Spotify is a great example of an application that utilizes all elements of Murray's "Grid of Affordances"

Taking that grid into account, we can see that the really good well-established digital artifacts and applications are always growing, trying to fulfill the user's needs and utilizing all the new technology that comes out of the rapidly growing, vast, field of computer science. You only need to have imagination and the sky is the limit. If the technology does not exist today, it will exist very soon. Good applications and platforms never stop evolving, as that causes people to lose their interest and move on to other ones.

2.3 Ethical considerations in design

While we are exploring this subject, we have to also raise some questions for designers and creators of digital artifacts that have to do with human-computer interaction. As designers, we need to consider the outcome of our design decisions and try to figure out how they will shape the user's perception and how they will form their interactions with the

digital artifact we created. We have the obligation to design with empathy and with foresight, as design choices that are made without considering their impact can significantly impact individuals and communities alike, usually the most vulnerable ones.

A good example of that is judicial AI based systems that are used to judge whether people will get rearrested, or for handing out sentences. One for example is the COMPAS system, which is used in some places in the United States in order to help judges reach decisions about pretrial release and for sentencing, by grading a person from one to ten on how likely they are to get arrested again, if released. Someone would say that systems like that are a great idea, so they can take some of the load off from the heavily understaffed judicial system. But they come with a deep problem, which is racism. These systems are designed and trained by humans, ergo, it is normal that humans have passed onto them some of their traits. In the United States, you are more than two times as likely to be arrested if you are black than a white person and five times more likely to be stopped on the road without reason. These statistics are very easily misinterpreted by the automated systems, which make the system to hand out heavier sentences to black people.

This example shows us the importance of designing systems that are impartial and they don't carry our own misconceptions and racist opinions. We have to design digital artifacts with empathy and with the good of humanity in our minds. We need to use digital design as a medium to promote collaboration between humans and not to spread misinformation and hate.

The digital medium opens up new horizons for expressing and presenting our ideas, constantly challenging the limits of HMI. New and upcoming technologies like augmented reality and virtual reality encourage users and designers alike to manipulate and augment

reality in immersive ways, changing the way users interact with digital content and pushing forward the lines of HCI. This is a good point to also mention that, despite all the positive outcomes of this fast advancement of technology, there are some downsides to it. Like a knife that can be used both as a weapon and as a very useful tool, these new technologies and their not really familiar "applications" can pose some new challenges to both designers and users, on issues like authenticity, identity and representation within HMI. It's crucial to consider the ways in which the new digital technologies and artifacts push to change the perception of ourselves and the world around us and recognize how technology, culture and identity influence one another through our interaction with the digital world.

In this era, the role that the digital medium is playing is also one where culture and collaboration is studied within the field of HMI. The existence of websites like wikis and social media encourage the social engagement between users and the contributions by them in the wikis, changing the way we think of the interaction between consumers and creators. While this participation culture is promising a great future of users interacting and participating actively in those procedures, giving easier access to knowledge and information, offering a place to diverse voices and especially voices that were not heard previously, we need to be careful. This free flow of information and creation causes issues that have to do with ownership, authenticity and identity. Basically, whenever we are online, we need to take everything with a grain of salt and always double check the source.

To summarize a bit, both as designers and as users, we need to always engage with the digital medium in a thoughtful, critical and ethical way. By doing that, we can use the digital medium not as a weapon, but as a very useful, powerful tool, that can help us improve humanity as a whole. As we navigate this vast field, we need to embrace all of its possibilities, but try to use them in a positive way.

3. Emotions

What are emotions and why are we mentioning them here? Well, the word emotion comes from the old French word emouvoir, which basically means to stir up. The word's meaning according to the Merriam-Webster dictionary is: "a conscious mental reaction (such as anger or fear) subjectively experienced as strong feeling usually directed toward a specific object and typically accompanied by physiological and behavioral changes in the body" or, another definition one even more compatible in the context of this research, "the affective aspect of consciousness"; because when we talk about interaction, humans always experience some kind of emotion and that is what they act on, as creatures with a high EQ (emotional intelligence).

That is why, as designers, we need to consider emotions when creating artifacts. That means that we need to consider the Human Factor (HF for sort). Even more so, when creating digital artifacts, as because of their unfamiliarity and hidden mechanisms, they may make humans dislike them or feel uncomfortable and not want to use them. They are a powerful tool in our arsenal, and used correctly, they can be very valuable when creating successful digital artifacts. Multiple studies have been made on what causes humans to react negatively towards something; in the following pages some of the theories that came out of the studies conducted on emotions and HCI will be presented.

3.1 Four Approaches to Design Integration

In the field of studies related to human factors and HCI, the need and the importance of trying to use emotional and affective elements into our designs and creations is becoming clearer and more important by the day. Because of that importance and need to study emotions and their effects on Human Computer Interactions, four different approaches have

emerged in the field according to Myounghoon Jeon in his book Emotions and Affect in Human Factors and Human–Computer Interaction (2017), each of them taking a unique approach to the matter at hand. These four are: Emotional Design, Hedonomics, Kansei Engineering and Affective Computing. Below we will briefly analyze each of them, as all four give an interesting approach to the subject at hand.

3.1.1 Emotional Design

Norman in his most known book, The Design of Everyday Things (1988) didn't take emotions into concern initially. He suggested his design ideas simply from a design perspective, focusing mostly on the usability of the items we create and not taking into account the emotions this artifact causes. Later on, in his book "Emotional Design: Why we love (or hate) everyday things" (2003), Norman shifted his perspective and presented a big shift in his design thinking. He started changing things, by acknowledging the huge role that a user's emotions play in the experience of using an artifact and enhanced their experience during that interaction. He explored how much aesthetics and the emotions they cause change and shape the interaction of a user with a new product. For that reason, three design levels were introduced. Those three are: visceral, behavioral and reflective and these three offered us a way to understand how someone interacts with a product or artifact on an emotional, behavioral and cognitive level. What Norman basically said was that emotion and cognition has shown that attractive things really do work better. For example, how a cheap wine tastes better in an expensive glass.

This new way of thinking and approaching design in general demonstrated the importance of not only taking usability into mind, but also considering the emotions that our design would stir when the user used the artifact. It highlighted the importance of finding the balance between the two. Both being useful and also cause

a positive emotional reaction. Consequently, what Emotional Design basically shows, is that we need to try and create artifacts that are not only interactive and efficient, but also create an emotional satisfaction to the user, while simultaneously being be aesthetically pleasing (although someone could argue that what is and what is not aesthetically pleasing is subjective and differs from user to user).

3.1.2 Hedonomics

Hedonomics is a new and unfamiliar term that resembles the term "ergonomics". It's a new way of thinking in the field of design and its introduction was the catalyst for a shift in the design field, as it put the focus on the pleasure and the well-being of the person that uses an artifact while designing. The term "hedonomics", which was introduced first by Hancock(Hancock et al., 2005) and Helander(Helander and Tham, 2003), is used to focus on the broader study of user experience, which goes beyond the traditional and outdated way of thinking about ergonomics and tries to include into the term the emotional dimension of design. What this field of study tries to explore is how to pinpoint what causes the pleasurable experiences of users and amplify them, while trying to avoid any discomfort and displeasure it may cause to the user while they use our creations.

There is many important and valuable information and feedback we can gain as designers, by simply analyzing the interactions of humans with the artifacts we design and their emotional responses to them. This way of thinking about design tries to create a user-centric philosophy. It tries to bring forth designs that not only meet the user's need for function, but also meet their emotional needs and cause an emotional response. Hedonomics, with their involvement of emotions into the design

process, are trying to lift higher the total quality of user experience, wanting them to better enjoy the interaction and to increase their satisfaction of the use of the artifacts created.

3.1.3 Kansei Engineering

Kansei is a word that originates from Japanese. Its origins lay in the Japanese philosophy, which helps understand, in a tangible way, the user's emotional responses, this is what the kansei is, and then translate them into concepts that can be used in our designs. Kansei Engineering highlights and focuses on the emotional responses and the resonance that comes out of using a product/artifact and places a focus on them, in stark contrast to traditional techniques that prioritize function over emotions.

Using techniques that Myounghoon mentions, like semantic differential scales and factor analysis, we can actually take the emotional reactions that a user has to a product and actually measure them, analyze them and use them. These results and data are then used to fix specific points of the design, like aesthetic elements or maybe some of its functions, so the user can experience the emotional responses planned during the conception and designing process of the product.

There is, though, a distinct issue with the practices of Kansei Engineering, that is actually acknowledged by scientists. That is the subjective nature of emotions. While using Kansei engineering, the designer strives to create products that can excite users and connect with them in an emotional level, and try to make them happy that they use the product, and increase their engagement with it because of that emotional

satisfaction. Though this is a great way of thinking, trying to create a product, or a design or an artifact, that actually resonates with a wide variety of users, that each has a unique way of experiencing the world, is hard and near impossible. That is why despite the wide use of Kansei Engineering principles, this way of thinking faces challenges when used for designs meant for a wide-range clientele and works best when designing for a more specific group of people that share at least some characteristics.

3.1.4 Affective Computing

This term was brought to us by Rosalind Picard, who first used it in her book Affective Computing (2000). This term and way of thinking brought a very interesting and revolutionary shift in the way we approached HCI. It proposed that we focus on the way that we can actually integrate emotional intelligence into the computational systems that we design and create. What Affective Computing basically brings forth is the exploration of developing new technologies that are able to actually recognize, interpret and respond to complex human emotions. That has the result of making the results of HCI better and of higher quality.

Affective Computing uses techniques that are a bit familiar, like facial recognition, speech analysis and physiological sensing, through which we can actually detect a user's emotions and psychological state. Using that result and that data, our systems can then adapt their behavior according to each user. This ability of the systems to adapt to each user opens up a whole new world of possibilities. We can now create more personalized and responsive experiences than ever and adapt to any user. The possibilities and the applications are endless and can be used in a lot of

different technological fields and domains, like for example in entertainment, healthcare, education or even assistive technologies.

A very interesting application of Affective Computing and its studies is in applications that have to do with the help and empowerment of people with various neurodivergent conditions, such as, for example, autism, ADHD and dyslexia. This ability to use emotional responses and adapt to each individual gives people with those conditions tools to better understand and manage their emotions and learn easier.

But, despite all the helpful tools it brought us and the huge advancements it brought in the field of HCI, there have been ethical questions that are raised. As we previously mentioned, questions about privacy, consent and biases, that are trained into the systems by us, as we saw previously with the judicial systems. If we can address these ethical questions in a respectful and considerate way towards all users, Affective Computing can be a gateway to revolutionizing the way that both users and designers interact and design digital artifacts, creating an environment that is more empathetic and can be more intuitive and easier to use by everyone.

3.1.5 A summary of the four approaches

By studying the research mentioned above, done on the emotional impact on HCI from these four points of view, we can see that we can gather very useful data and methodologies, that are strong tools in our arsenal. Despite their different points of view, what these four have in common is that they all put the human emotions and reactions up and center and try to help us better integrate those into our designs. They

try to give us a way to put them into measurable results, in order for us to take advantage of these results and create systems that evoke stronger emotional reactions. What this means is that we design something, we fabricate it and then we test it. We then get back the results of those tests and we evaluate them. Based on the results of the evaluation, we can then redesign and adjust our "product" to evoke the maximum emotional response.

If we can better understand the complicated and vast nature that is the human emotions and experiences, using the tools that these four approaches give us, we can design and create more engaging and intuitive systems, that can cause stronger emotional reactions and connections, catering to diverse groups of people and meeting their diverse needs and preferences. That way, by using the power of the human emotions and the emotional reactions, we can not only design with function and ergonomic design in our mind, but we can also create strong and meaningful user experiences, that also cater to the emotional needs of the user.

3.2 Uncanny Valley theory

The Uncanny Valley, which I had studied when doing research on animation techniques and their history, is a very interesting one. It's a phenomenon that was initially studied in the field of robotics, but the theories and questions that it brings forth are applied and used in most designs, especially ones that have to do with designs of humanoid forms.

But where did the term come from and why is this the one used? The term "Uncanny Valley" is actually the direct translation of the term used in Japanese by Masahiro Mori, the creator of the term and the first to create this theory and study it. The Japanese term is

"bukimi no tani". The word "tani" is actually translated as "valley". The "no" in Japanese is used as a connective word. The word "bukimi" has quite a few translations. It can mean "eerie", "strange" and "uncanny". The choice of the term "uncanny" is attributed to the attuning with the psychological studies done by Sigmund Freud done in those years. In his paper from 1919, titled "Das Unheimlich" and was translated as "The Uncanny" (Freud, 1960), he described the meaning of "uncanny" as a category of scary, which leads back to the familiar and the likeable, something that was very in line with Mori's theory.

3.2.1 First studies on the phenomenon

In 1970, at the institute of technology in Tokyo, Doctor Masahiro Mori, a Japanese roboticist and professor at the university of Tokyo, expressed the following a hypothesis that became the central idea of the Uncanny Valley, that we can make a robot that is the closest to human appearance that we can imagine. When we got it so close to human likeness, would our relationship with the robot become better or would the correlation between how much we like the robot and its realism experience a drop?

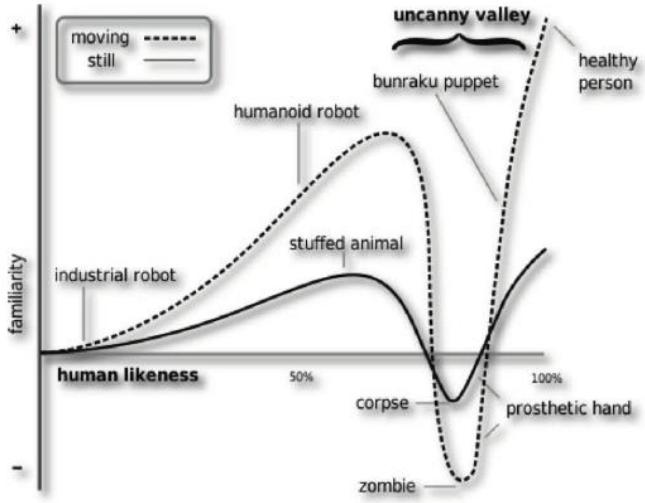

Figure 7 Dr. Mori's "Uncanny Valley" theory has been debated for years in the field of anthropomorphism. This is the graph that he used to explain his theory.

According to Mori's theory, movement and shape play a huge role in the acceptance of "realism". When we try to reach a human likeness in a design, if we imagine it as an axis system on one of which we have the human likeness and on the other we have the familiarity, we'll see that the more we reach it, there will be a point where there will be a drop in familiarity and it will cause discomfort to the user or interactor. So, in order to not cause displeasure, both shape and movement should be taken in mind. For example, in robots, as Mori said, we can see that despite their human appearance, when they can't move in a human way, it still causes us discomfort when we interact with them, so they still fall in the "Uncanny Valley" despite having a pretty realistic shape.

If we go a step further, Mori gave an example with a dead body. Though obviously not the best thing to look at, the body has a human appearance, so it doesn't

cause discomfort because of its shape. However, if the body was to jump up and start moving, it would cause discomfort, as it would cause negative emotions (fear, shock). That is because despite its human appearance and its human movement, this is an action that shouldn't be done by a dead body.

On the contrary, Mori did a test with some Japanese dolls, the "bunraku" dolls, which proved that movement can also help us avoid dropping into the valley. These dolls are moved by their puppeteers, which are on stage and dressed in black. The movement of the dolls is quite human-like. That causes our brain to accept them as human-like, but not human and not categorize them in the Uncanny Valley.

Some of the immediate uses of Mori's theory can be found in the designers of robots, prosthetic limbs, avatar creators, animators and everyone that creates human-like forms or projects. Mori supported that the first peak of acceptance in his graph, comes quite far from the point that the uncanny valley is shaped and it's even further than the complete realism. At that first peak in the graph, lies a good target for designers, that is easier achieved than the complete and utter human likeness.

Mori suggested some solutions, because the closer we try to reach reality, the easier it is to fall in the uncanny valley. For example, to people who design prosthetic limbs, he suggested that if they can't make a realistic hand, they should go the other way and create one that is more robotic. That will make people immediately recognize it as something not human and accept it as a prosthetic hand. And even if designers could design the perfect looking human hand, that illusion would break as soon as we went for a handshake as the temperature and the stiffness would be weird, and that would create discomfort and bad feelings to arise.

The best part about Mori's theory is the fact that he doesn't set any rules and any limits to this theory. He doesn't specify what is accepted and what is considered likable. This way, his theory can be applied in many fields of science and many different designs, without needing to be changed.

3.2.2 The existence of the phenomenon

Since this is a phenomenon that is recognized by scientists, we should be able to actually explain its existence and be able to successfully reproduce it. Unfortunately, an exact explanation doesn't really exist, not in a complete and absolute form at least. Instead, we'll take a look at some of the explanations that have been given over the years.

A common explanation that is given to the problem of hyper realism, is the direct connection to the increase of information that is picked up of our brain. That increase in information brings more details that our brains can take apart and discern whether something is real or not. Telling real things apart from fake ones becomes easier when more details are provided. For example, when looking into an AI generated image, we know that something is wrong and that it's not a real photograph. But the question that arises here is that shouldn't the increase in correct and realistic details make the lesser false details disappear? And since that happens, does that mean that we have some sensitivity to false information? Well, this is not the whole truth and not the answer.

Ramey, in a paper that he presented in 2005, mentioned that the Uncanny Valley phenomenon can be shaped in such a way that what we see always stays in a constant state of realism. This realism is always rooted in the space that exists between humans and machines and exists when we can actually tell in which category of the two something exists. Given that the filters that we use to categorize objects and situations into the uncanny valley phenomenon or not, are mandatory and are always "active", we can actually find the exact moment when what we have created stops fitting in either the man-made category or the human category. That point, when our brain loses the ability to categorize them in a specific group, is the moment that discomfort is created.

Another explanation that is rooted in the original theory by Mori, is the fact that a movement that is not considered human enough, can highlight mistakes and weird details that exist in the designed form. The opposite can also happen. This idea in general is basically that the human brain can understand human actions and these actions evoke specific feelings and reactions it has to those. If all the clues and information that the brain picks up are not consistent, then the combination of the many inconsistent indications that are picked can cause us stress or even make us not understand what we are seeing. The connection between the form and the movement is actually very interesting and has been studied by multiple scientists who mostly reach the same results as Mori. That result is that despite the fact that the movement and the appearance aren't conceived as something whole in the beginning, after observing the artifact closer, these two merge and need to both lead us to the same outcome, otherwise if they don't match, they lead us to discomfort.

There are even more studies done, like for example whether we are the only "humanoid" species that experience this or other species also experience the same problem as us (it is in fact, not a human only experience and it is experienced by other species like chimpanzees and monkeys). But for the sake of keeping this dissertation short and to the point, we won't analyze it further. A whole thesis could be done on the Uncanny Valley phenomenon and we would still not have covered everything.

It is important to mention that the studies on the Uncanny Valley are covering many fields, like neuroscience and psychology, robotics and animation. The subject is a vast one. However, we should consider that as with Emotional Design, this phenomenon can't be put in strict boxes. Since it has to do with human perception and emotions, each human perceives the phenomenon differently. My graph on what I consider human, can be completely different from someone else's. That is why some concerns on this theory have been raised. However, it is a helpful tool when designing digital artifacts, specifically interactions with AI, as these are still too new for people to understand, and this unfamiliarity can make the users uncomfortable.

4. Computer Vision

An important chapter and field of study for this thesis is Computer Vision. If we had to compare computer vision with something, it would be human vision. Computer vision is the way that computers "see and understand" the world, by analyzing and interpreting patterns that exist in images and video in a similar way to humans, but not necessarily as they can run much more complex algorithms than the human brain. Computer Vision is the field that is at the crossroads of computer science and image processing. It tries to understand the way that humans interpret visual information and decode it and aims to replicate it on the machines, for example, by recognizing a face. How do we see a face and know who it is? What are the characteristics that we pick up on and think that this is a human and not a cat and that their name is Peter and John?

Computer Vision is the field that studies and answers such questions, turning them into systems that can interpret and analyze visual data. These types of systems can find application in many fields, for example on autonomous vehicles where they help the car move in the busy streets and recognize signs and humans and react accordingly, or even medical systems that can interpret the exams of a patient and give out a diagnosis. The applications are many and spread across many fields of study, bringing new solutions to previous issues in many industries and opening up new doors for this technology to be used.

Basically, what Computer Vision focuses on is the extraction and processing of any data that is deemed useful, under the conditions we have set, from images and video. In order for us to understand those processes and functions of computer vision, it is important to understand things like image acquisition, processing, feature extraction and the interpretation of all these afterwards. This field is truly vast and if one takes a deep into it, they can get lost

in it. In order for this thesis to remain at a normal length, we will barely scratch the surface of this field and only some key points will be mentioned here.

4.1 Pattern recognition

Pattern recognition in Computer Vision is the most important thing to understand. It is the holy grail of it, as it is what makes the computer "understand" what it is that it's seeing at that moment, for example, discerning between a human and a dog in order to find those key points that we will have to teach the computer to look for in an image. This steps into the world of Machine Learning, a field that is so important for computer vision.

Everything in pattern recognition is based on our ability, as humans, to perceive the world around us by understanding patterns. We see leaves upon wood, being tall and we know it's a tree. We see two eyes, a nose and a mouth, in specific places on the face and we know we are speaking to a human. This is a human trait that we learn at a very young age, without even understanding that we are doing it. Even from our infancy, we have the ability to recognize familiar faces, shapes and objects. We hone this skill every day. This skill of ours is what lays at the heart of pattern recognition in computer vision, where algorithms are trained and programmed by scientists and designers alike, to function like a human brain, teaching them our capacity for visual perception. It's a task that is easier described than actually done.

Very important and tied in everything that has to do with pattern recognition is Machine Learning and Neural Networks. Machine Learning is what makes algorithms able to "learn". By feeding the algorithm with the patterns that we want it to detect, letting it run in many iterations in order for it to be trained, we teach it what we want it to recognize and what

we want it to give us as a result. Feeding it, for example, images of bikes, we can then have it recognize and track a bike in a video. While we run that algorithm and we continue to optimize it, the results get better and better with each iteration, and we can have it make accurate predictions or even detect things that the human eye couldn't pick up.

4.2 Image Segmentation

Image segmentation is an important field of computer vision. What it does is basically break down the image into more coherent and meaningful regions, so that the computer then can understand more easily, without spending too many resources. It aims to set a clear boundary on what the computer is looking at at the moment, for example set apart objects from their background. It tries to make the machine understand the structure of the scene. There is a wide variety of techniques and tools to do that, each of those fit to deal with certain challenges and goals.

A good example, as color-based criteria will be used in the application that will be developed for this thesis, are some of the most well-known image segmentation methods, such as thresholding, which operates with very simple criteria. It looks at an image and tries to divide it into homogenous zones using pixel similarity. That similarity can be based on color, for example. That way, the computer has less information to go through. This technique is quite basic and newer and more advanced ones exist and can do the job faster and easier.

As technology advances, new tools appear. These have helped the science of image segmentation make huge leaps very fast. These leaps were pushed forward by the developments made in the fields of Machine Learning and Deep Learning, who have come forward with new methodologies and techniques.

Just a brief mention of the graph-based segmentation algorithms, which have helped in the development of spatial relationships between pixels. What this means is that the computer can distinguish regions inside complex visual scenes. Another very brief mention should be to the convolutional neural networks field (or CNNs for short), a smaller field under deep learning. This field has given very powerful tools for semantic segmentation. What this means is that the pixels of an image are all assigned a label, based on some categories that were set by the designer. This helps the computer put every pixel into a semantic category and discern easier between them.

There are still some limitations and challenges in the image segmentation field, despite all of its advancements. To address those, there are ongoing researches and new techniques applied, trying to solve the issues and improve the algorithms that do this job. Generally, this is a field with still many possibilities to be discovered and with a lot of room for growth and that can help reshape the way we think of visual perception and can help us improve and advance our understanding of the visual world.

5. Practical Application

5.1 Initial Idea and Description of the Art Concept

One of the biggest challenges was to choose and focus on one main subject for this dissertation, given that the world of the HCI offers vast and magical possibilities of research. Delving into the subject, it was important to have a concrete design idea which would utilize my background as a computer scientist and would build the code around it.

After going through many different ideas, motion tracking became the most prominent one because I already had some basic understanding of its foundation due to my studies on 3D animation.

Physics simulations in the digital world are very interesting. The attempt to recreate the natural world with bits inside the computer's environment, as close as possible to reality or to even create completely new worlds that bend reality, has always been fascinating and a field where humans struggle constantly to reach perfection. In the fields of 3D animation and VFX, that is done by using simulation programs, that are very demanding of the hardware in order to create a realistic enough result, or one that is believed to be real by the viewer. But can it be done with code, and can it be done on a low budget computer, if we don't use 3D graphics? Maybe not so realistic, but we can simulate physics with code. We can also throw all real world physics out of the window and create our own. It is up to us. Here, since I wanted to simulate stars falling, I operated with one force that exists in reality: gravity. But more about this will follow later in the breakdown of the code and how it was done.

Taking inspiration from a popular fantasy book named: "A Court of Mist and Fury" by Sarah J. Maas (2016) and its quote:

"*Another star crossed the sky, twirling and twisting over itself, as if it were reveling in its own sparkling beauty. It was chased by another, and another, until a brigade of them were unleashed from the edge of the horizon, like a thousand archers had loosed them from mighty bows."* (Maas, 2016, p. 434)

From that, the concept of creating an original and personal starry night was adopted.

In the book, later on, the stars are revealed to be the souls of the fallen and the deceased coming to visit their loved ones during the Starfall night. That description and that idea behin.d it were inspiring enough to consider creating a starry night with some of its stars falling

The stars, as in the book where they fall and collide with the faces of the fay people there, will collide with the user's face and bounce off. Initially, I thought of having the program track an umbrella of a specific color that the user would be holding. But that didn't sit well with my vision for the scene from the book, so I decided to go with the face tracking. Bellow, I will go further into analyzing how I came up with the final solution for the problems I came across.

I tried to create my design based of Murray's "Grid of Affordances". The idea was to fit my application into all four grids. It is participatory, as it creates more stars when the spacebar on the keyboard is pressed. It is procedural, as it uses sensor devices (the computer camera) and it gamifies a concept. But, due to time restrictions, I couldn't fit it into the other two, but as we'll see in the Future Development section, the idea was to make it encyclopedic, by including real constellations and spatial, by having it reflect the sky of the country you are using it in.

I then thought about the principles of emotional design when I came up with the idea. The idea that a wine tastes better in an expensive glass, even if it is not the best wine, was something that I was thinking while coming up with an idea. I didn't have the best hardware or the best sensors, but if the design was good, it could work. I knew that I wanted to include the starry night and have the user interact with it. The problem was the cost. Since I couldn't afford some better equipment, I tried to make a good design, that could cause some emotional response despite not having the best equipment. While designing my final product, I was trying to keep in mind the wonder that the night sky causes to us when we look up to it and the playfulness that the scene in the book that inspired me invoked. It may not have been done completely and really successfully, as I had to overcome the physical limitations that I came across, but I reached a pretty good point. If this idea was upscaled and set in a full room, so it could be more immersive, and music was also used, I am sure that the results would be more breathtaking. For now, we'll have to be satisfied with a small taste of the smaller scale product.

5.2 Methodology

5.2.1 A small introduction into Processing

The most interesting part of this dissertation, since the research came down to finding and utilizing to the fullest the tools that I had in my possession and broadening my knowledge in the coding world. I came across many difficulties, both in terms of coding and hardware, as I had many limitations on both areas. Starting from the basic handmade sketch and the idea described in my favorite book, trying to recreate it digitally by programming it, I came across many obstacles, that forced me to find creative ways to overcome them.

Figure 8 Processing is a project created to help people learn to code, while being creative and giving them the ability to create stunning visuals.

Processing is a very adaptable software sketchbook. It is also a programming language, suitable for learning how to code. It caters especially to people who did not think that coding could be used to create visual art. It is Java based (initially, as in later versions JavaScript, Python, and Ruby were introduced to the environment) so it's easy to learn and understand, especially by someone, who has a coding background and even better if they have a Java background. Working on an open-source platform, that is developed by volunteers and its users, means that there is a community behind it that has created an endless number of projects, libraries and tutorials, that can help a beginner to learn and understand both to code, but to also create stunning visual art.

For all of these reasons, Processing and, specifically its newest version Processing 4, was chosen Java was also chosen as the coding language of the sketch, as it's the coding language I am most comfortable with. It is much simpler to create visual output in Processing than to write code into basic Java in a coding platform, like for example Eclipse, which is great and has an amazing debugging console and is great at helping you understand where your error is, but is quite rigid when it comes to graphics, as it demands specific knowledge and effort to create. Meanwhile, you can do the same in Processing with a lot less lines of code.

5.2.2 Breaking down the idea into parts

Starting the design process meant that I needed to break the idea into smaller parts that could be more easily handled. When coding, this is the most important part, as it can make the code much simpler. A program, or better to use the term sketch as it is in Processing, always has the main class. In it, the main two functions of Processing exist. These are the setup and draw functions. In setup we initialize our variables and set up anything that will be needed later on, like for example our physics world. In the draw function, the sketch is created and the functions and classes we created are called.

My idea was that I would have some static stars and some moving stars that would fall down. Both of them have something in common. They are basically stars. Their properties just change. That meant that I could create two separate classes, one that handles static stars (named Stars in my sketch) and one that would handle the falling stars (named FallingStars in my sketch). These two classes are carrying a constructor function each and a display method. But we already see that they both carry the exact same classes. Most of the attributes of the stars are the same, apart from their speed, the falling stars have a speed that is larger than zero while the static stars have a speed of zero. So why not keep the two separate classes, but write the functions only once? Here comes a great solution. Inheritance. I will explain this further, but what inheritance does is basically create a parent children relationship between the two classes and all the functions and variables from the parent class are automatically transferred to the children classes as well.

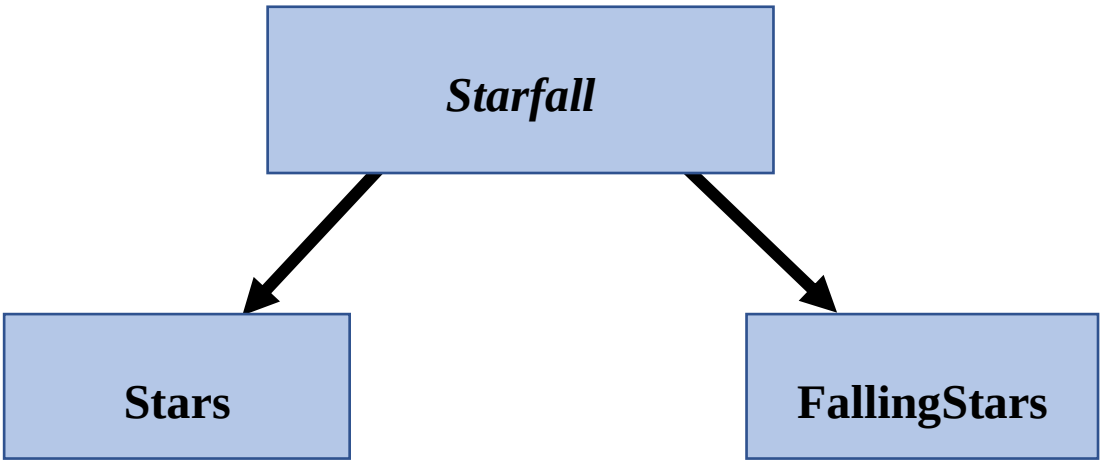

Figure 9 Graph that shows how the main class is connected with the other two classes.

In Processing, and Java in general, I've found that we can create an Array List that generally contains stars. No matter if they are FallingStars or Stars, they are all contained in the same array. Processing is able to distinguish between the two types of stars as it travels through the array and call the classes and functions appropriate for each of the two types of stars. This is what we call polymorphism in Java. It's a very helpful tool, especially in combination with inheritance, as it can help us save time and simplify our code.

Having broken down the idea into these parts, made it easier to focus on the next steps. The next step was to think of the rest of the "tools" that would be used to materialize the concept. That meant that I had to do some research in the physics libraries, plus the face detection libraries that are offered.

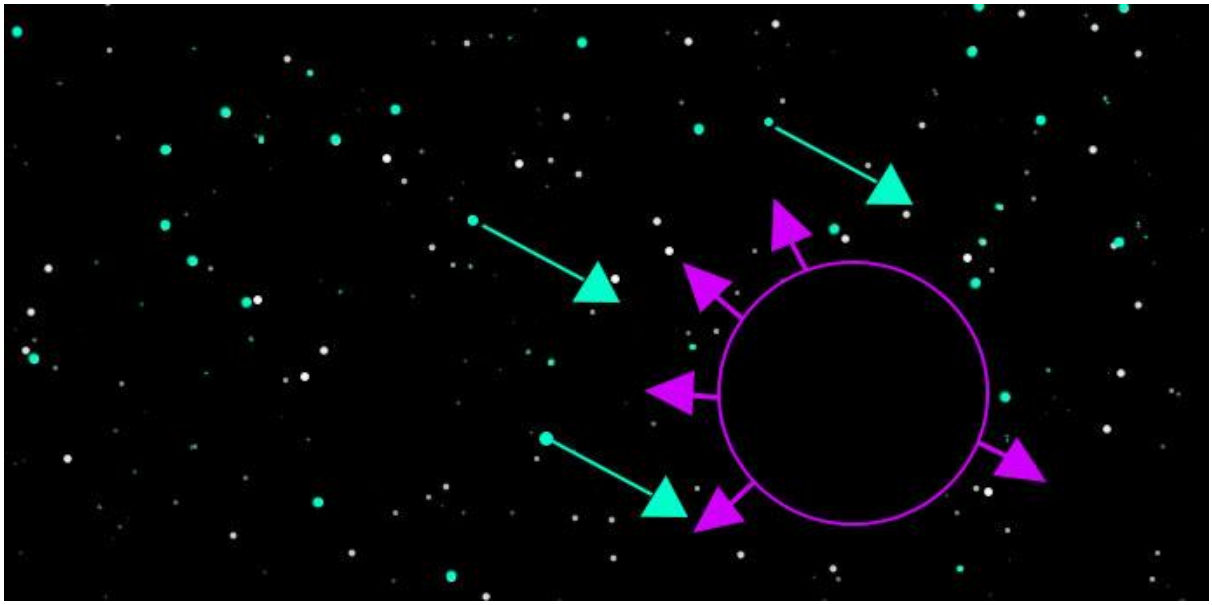

Figure 10 First sketch of the stars. The teal stars are the ones that will be moving, along with the gravity that pulls them down. The face that is used as an attractor, is the pink circle and the pink arrows are the force that it uses to "repel" the stars and make them bounce away from it. The white stars are the ones the static ones. Their size depends on how far they are from the screen.

5.2.3 Libraries

After having chosen the platform, the language and having broken down the problem, I started researching into the libraries that I could use. Since Processing is not something new, there are libraries for whatever one may think of. I started researching and I came across many simulation libraries, each one with their own

limitations and uses. Finding the right ones came with many trials and errors, as well as to how easy I found it to use, how well documented its functions were online and if there were any video tutorials on them (because that made understanding them so much easier).

The first library I looked at was the teilchen library created by Dennis P. Paul. This library offers a very extended explanation about how a particle system works and is great for introducing someone into how particles work. It offers a great explanation of how forces work on particles and how mathematical operations happen between two vectors. It is also one of the libraries that comes with the Processing environment, so it is very easy to install and access its examples. After experimenting with this library, I found it quite good and that it has many useful functions. My issue was that I couldn't find many explanatory videos on its classes and how they work, so after a few days of experimenting with it, I decided against it and realized that it would be better to use a library that will be easier for me to use. However, the basics that I learned from this one were very useful for me later on understanding the other libraries.

The next library that I did research in was the library Fysica by Ricard Marxer, which is actually a wrapper around JBox2D. This was ultimately the reason that I abandoned it. Fysica tries to simplify a lot of the functions of JBox2D. But JBox2D is more extensively used and has more tutorial videos and explanations, so that was the only reason that I dropped Fysica, even though it was a nice fit with my concept.

After having decided that the first two libraries were not a good fit for me, I came across classes online by Daniel Shiffman. In his YouTube channel "The Coding

Train", Shiffman has extensive videos on physics in both JBox2D and toxiclibs, both powerful and very useful libraries, each offering tools for many different simulations. JBox2D library is a project led by Daniel Murphy and it is a close Java port of Erin Catto's excellent C++ Box2D physics engine and Google's LiquidFun physics Engine. Toxiclibs was originally written by Karsten Schmidt for Java and Processing and is being ported to javascript by Kyle Phillips. Because of Shiffman's videos, I started experimenting with the two. There were some key differences. The first thing I considered was that JBox2D, as the name suggests, offers tools only for 2D designs. Toxiclibs however, works both in 3D and in 2D. That wasn't really an issue, as my design is a 2D design. The second one was that JBox2D is offering tools for collisions. However, toxiclibs does not do collisions. It is targeted more towards soft body and liquid simulation. (However, as we'll see below, there are some ways to trick it and make it work with collisions). The final difference is that JBox2D works with real world coordinates, cartesian coordinates, in contrast with toxiclibs, or most other libraries in Processing and processing itself, that work with pixel coordinates. What that means is typically when, for example, set our window to the location (20, 20) the window in pixel coordinates will pop up in the pixel 20 in the x axis and 20 in the y axis. But if we wanted to do that in JBox2D, we would have to translate the cartesian coordinates to pixel coordinates. That was ultimately the reason that made me choose toxiclibs over JBox2D, despite being targeted more towards what I wanted to do. It's easier if you don't have to translate coordinates, every time you need the library to calculate something.

The physics library that was chosen for the practical part of this dissertation is Toxiclibs, which is an excellent library that has many tools for physics, especially for particles and springs. I found out pretty soon that this library was easily integrated with my project. This library is not really recommended for particle collisions, as it

does not have tools for these. However, it offers good attractor tools, that can easily be turned into repeler objects and can simulate collisions with the attractor object.

Finally, since I wanted to track the face, I looked into how I could do that with the computer's camera, in order to use equipment already in my possession, but also accessible in case that someone else wants to recreate or test the code and experiment. Researching into how that could be done, I came across the OpenCV library. According to the creators, OpenCV is an open-source computer vision and machine learning software library. It was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. It has C++, Python, Java and MATLAB interfaces. For our purposes, the Java based Processing one was used. This library offers tools that use a camera to do image processing. It can easily track a face, by just using a few lines of code. One issue I came across is the fact that OpenCV works only with the default Processing renderer. When I tried to use the P2D or the P3D renderers, for example, the library stopped working properly. This is something that needs to be kept in mind when using it, as someone may need to change their design method.

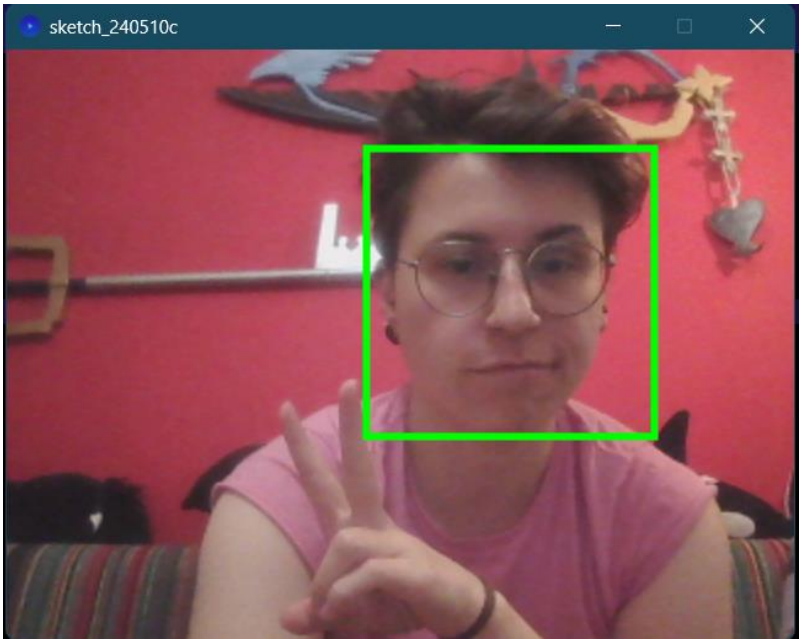

Figure 11 First time I successfully tracked my face

5.2.4 Toxiclibs

A special mention has to be made on some of the functions and some of the tools that were used from toxiclibs and how this library works. First of all, not only with toxiclibs but generally with physics engines, we need to set our world and our physics system. That is done in the setup function in the main class. Then we need to create our bodies and shapes (depending on the library used, these may be slightly different, or they may be one and the same). In toxiclibs, shape and body are one and the same These are the things that experience the physics that we set up. Afterwards,

we have to add all the objects and all the forces that we set up in the physics system, to essentially bind everything together.

```
physics = new VerletPhysics2D();
```

Code 1 Creating the physics object

One of the first things I set up was using the face as an attractor (or repeler in our case). An attractor object has three variables: what is the object that acts as the attractor, the radius of attraction and the strength of the repelling. If in the attractor object, instead of a positive number, we give a negative one, it is immediately turned into a repeler object. Finally, if we set the attractor's radius into a number close to zero, because the particles have to reach the attractor for it to show its effects, it is basically perceived as a collision. This is one of the ways that we can simulate collisions in toxiclibs. We could do them manually by calculating the collisions of the particles using forces and then adding them to the acceleration of the respective particle, but that would take more time while coding, so since we have the tools from the library, we can save time by utilizing them.

```
faceAttractor = new AttractionBehavior2D(facePos, 10, -20f);

physics.addBehavior(faceAttractor);
```

Code 2 Creating the attractor

In toxiclibs there are already objects that have to do with gravity, which is the force that pulls down the stars. It can be set up to drag them diagonally and its strength can also be adjusted, so it pulls them faster or slower. This creates the illusion of stars "falling". The Gravity Behavior we are adding is the 2D one, as the sketch is a 2D one. This behavior is expecting a Vec2D vector from us, so it knows the direction towards where it will pull the objects and also to know the strength with which it will pull them.

```
physics.addBehavior(new GravityBehavior2D(new Vec2D(0.06, 0.1)));
```

Code 3 Adding the gravity to the system

Toxiclibs also offers its own type of vector variables. These are the Vec2D and Vec3D vectors. As we are working in 2D space, we'll use the Vec2D type. This is very useful, combined with the VerletParticle2D, which is toxiclibs' particle class. This class actually inherits all the traits of the Vec2D class (this falls under the inheritance part and will be explained further later). We could have created our own particles, so they have only the characteristics that we wanted them to have. But again, due to time constrictions and since the physics library I use have a particle class that works, the Verlet Particles were chosen. This inheritance relationship between the two classes allows us to use the VerletParticle2D class and control the particles we create through using vectors and their coordinates. These classes can also be used to extend our own particle classes, so our particles can have physics. Very cool stuff!

In toxiclibs, when we create anything that has to do with the world's physics, we need to add it to the physics object we created in the beginning. So we always need to add everything we create there. That means that all the following commands add something into our physics system every time they are called:

```
physics.addBehavior(new GravityBehavior2D(new Vec2D(0.06, 0.1)));

physics.addBehavior(faceAttractor);

physics.addParticle(star);
```

Code 4 Adding things to our physics world

Here, in the order they are written, they add the gravity in the system, they add the attractor and then they add the stars in the system, so they can be affected by the other forces.

Last thing that we need to remember about toxiclibs, is that we need to update() the physics. Especially at the end of every time we loop through, with the draw() function.

```
physics.update();
```

Code 5 Updating the physics system.

5.2.5 Inheritance – Tying everything together

Inheritance. Such an interesting term in the world of object-oriented programming. It is actually one of the basics that one should understand when starting to code with object-oriented languages. It is very important as it can make our code more easily reused and also more easily understood, as it helps structure it better. It allows to set a very general class and then create more specific ones, that add some more specific details to the original function. This saves us time, especially in more complex applications, as the more specific ones, "inherit" all the general traits from the "parent" class and then we only have to code the new abilities that these classes have!

Using this general notion of inheritance, we can consider the following graph:

Figure 12 Stars extend VerletParticle2D and then FallingStars extend the Stars class. That means that all stars inherit everything from the VerletParticle2D class.

This means that the Stars function inherits all the traits of VerletParticle2D and then the FallingStars class inherits all the traits from the Stars class, which means that it also inherits all the traits of VerletParticle2D. It's a parent child situation, where the class that is extended by another class, gets all the "family characteristics". A simple solution that can actually save us so much time when coding, as we can simply inherit some functions or some variables from another class. In our example, we inherit the constructor from the VerletParticle2D type and then the FallingStars class inherits all the classes we've written in stars. What we can do to differentiate them a bit, as we can utilize the star's movement, is overwrite some classes. That means that we rewrite them, so they have some unique traits for each class. For example, the isDead() function is always false for the static stars, since they are either way drawn inside the screen from the get-go.

```
boolean isDead(){

    return false;

}
```

Code 6 The isDead() function for the Stars class is always false, since the static stars are designed inside the screen area from the get go.

However, in the case of the falling stars, we'll need to check whether they are outside the screen, so we can delete them. So, we overwrite the function for them, so when it is called on a FallingStar star, it checks whether it's true or false.

```
boolean isDead(){

 if (posx > width || posy > height){

    return true;

 } else {

     return false; }

 }
```

Code 7 Checking if the FallingStars are dead, which means they go out of the screen's bounds.

5.4 Issues Encountered

While coding and trying to bring my imagination to life, I tried many things and I ran into some issues that I had to find a way to overcome, in order for the code to run successfully and to also optimize some things.

Starting off, one thing that I first tried, was to create some stars on photoshop, and then tried to use them as sprites on the particles and add them in the sketch, so they could be more realistic and have them leave a beautiful trail. That didn't go really well, as that was very demanding for my computer, and despite having a decent enough laptop, it still couldn't support having an image for each star. So I had to scratch the idea and design them myself within processing, so the sketch could be easier to be run by any machine. Designing the stars in their classes and only calling their display functions instead of doing it in the draw function, made the sketch run even faster and be more stable.

At first, I wanted to give a better understanding of depth in my sketch by using a 3D renderer. Processing comes with numerous renderers that can be used depending on what we want to achieve. So here I ran into one big issue. OpenCV does not work with other Processing renderers, apart from Processing's standard one. That meant that I had to find a way to "cheat" and create depth in another way. That way was to play with the size and transparency of the stars, so I could trick the eye and create some depth. Another issue that I encountered while using OpenCV was the fact that it couldn't track the body very accurately. If I set it to track a body, or even the upper body, if I didn't wear black clothes in front of a white wall, it would miss the body a lot of the times. So I had to stick with tracking the face, as it does track the face very accurately.

The last issue I ran into was to find a way to remove stars and create new ones, whenever those left the screen, while the sketch was running. For that, I created a getter function in the Stars class which gets the current position of each star. I then remembered a very useful tool that Java offers. Those are the Iterator objects. These objects can travel in the array, find the variable we want to remove based on a condition we've set, remove it and then tidy up the array, so it's ready to be run through again. These work even when the size of the

array is dynamic and changes. I created a function in each of the Stars class, to check whether the stars were outside the screen and using that and the iterator object, I managed to remove the stars that left the screen. Every time a star is removed, a counter goes up and at the end of the draw() function, that amount of stars are drawn again, so the number stays the same.

The other issues that I ran into, were explained in previous paragraphs, as they had to do with the physics libraries, inheritance and polymorphism, the last two being tools that everyone that wants to code in Java needs to know and have understood, as they can be encountered in every stage of code writing, be it front-end, back-end or anything in between.

5.5 Future Development

This project started as a project of love for coding and the magical world created in Sarah J. Maas's books. One day, I hope I will be able to refine this application and create something more visually stunning, worthy of the vast and rich world that she has created, as I would like to use it as a visual for people to see what I was imagining when reading that book and invite them into this world. I am an avid reader, so if this application tempts even one single person to pick up a book, especially from this specific book series, I would be happy, as books can transfer us into worlds very far away from us and help us broaden our way of thinking and our imagination.

On that note, in future versions, I would like to include some constellations in it and remove some of the randomness from the background. That would better reflect the real sky, or even include the constellations from the book, and it could maybe even be used to help people learn the names of the constellations and their position in the night sky. I could have made a background on photoshop and had that for this version, since there is no interaction

with the background stars either way. But I wanted to keep the option of including them in the physics system open, so later on a user could interact with them as well, by touching them and moving them with their hand. I would like to look into ways of designing them with code and not making them into a simple drawn background image. What would be even better, is if I could take data from the device's location, as well as the date, so the background could actually reflect the night sky from where the user used the application and the moment that they used it (as constellations change, depending on the season of the year we are in). Time restrictions again, didn't allow me to look into it in depth, but it is something that I want to investigate in future iterations of this project.

Another improvement I would like to add, is more realistic stars, as now they are just dots. I wish that I will be able to add some luminance to them, so they could shine and blink as real stars do. Maybe if I have done a lengthier investigation in drawing tools that are offered either by Processing or even its libraries, I would have found something. I would also like for them to explode into small particles once they come into contact with a face, like they do in the book, and stick on the user's face for a bit.

To have better control of the user's position, I would also like to use a Kinect camera. This is the camera that was used by Microsoft's Xbox to control games using the user's body movements. That means that it can give us an abstract skeleton, so we can track a specific body part and calculate better the distance of the user from the camera. It also offers more details about the contours of the body. With all of these controls, I would be able to control collisions with the whole body and when the hand moves, I could maybe clean the face from the pieces of the stars that will have stuck on the face, or the user could even be able to shield their face and avoid the collisions all together or simply interact with the stars and move them around and play with them.

I wish one day that I could make this into an installation that takes up a full room, so the user can immerse themselves into the world and even have appropriate music, to elevate the illusion of being into a magical world. The music would help to trigger some more stars into falling down and would also make this design more magical and immersive.

All in all, this project made me delve into some basics of object-oriented programming and challenged me to change the way I thought of things in the world of coding. It challenged me both as a computer scientist and as a new designer. There are many ways to make this application better and I wish I could have explored some technologies more. We'll have to wait and see where I'll be able to take this and how I can improve it. For now, enjoy looking up to the stars and I'll close this with my favorite words from the book that inspired this project:

"*To the stars who listen-and the dreams that are answered*" (Maas, 2016, p. 337)

References

Boy, G. A. (Ed.). (2012). *The Handbook of Human-Machine Interaction: A Human-Centered Design Approach*. Ashgate Publishing, Ltd.

Encyclopedia Britannica. (2024, April 4). *Uncanny Valley*. In K. Emily (Ed.). https://www.britannica.com/topic/uncanny-valley

Giese, M., & Poggio, T. (2003). *Neural mechanisms for the recognition of biological movement*s. *Nature Neuroscience Review, 4*, 179-192.

Hancock, P., Pepe, A., & Murphy, L. (2005). *Hedonomics: The Power of Positive and Pleasurable Ergonomics. Ergonomics in Design: The Quarterly of Human Factors Applications, 13, 8-14.* DOI: 10.1177/106480460501300104.

Heaven, W. D. (2020, July 17). *Predictive policing algorithms are racist. They need to be dismantled. MIT Technology Review*. https://www.technologyreview.com/2020/07/17/1005396/predictive-policing-algorithms-racist-dismantled-machine-learning-bias-criminal-justice/

Helander, M., & Tham, M. P. (2003). Hedonomics--affective human factors design. *Ergonomics, 46*(13-14), 1269-1272. DOI: 10.1080/00140130310001610810. PMID: 14612318.

Jeon, M. (2017). *Emotions and Affect in Human Factors and Human-Computer Interaction*. Academic Press.

Maas, S. J. (2016). *A Court of Mist and Fury*. Bloomsbury Publishing.

Merriam-Webster. (n.d.). *Emotion*. In *Merriam-Webster.com Dictionary*. Retrieved May 2, 2024, from https://www.merriam-webster.com/dictionary/emotion

Merriam-Webster. (n.d.). *Interaction*. In *Merriam-Webster.com Dictionary*. Retrieved April 29, 2024, from https://www.merriam-webster.com/dictionary/interaction

Murray, J. H. (2012). *Inventing the Medium: Principles of Interaction Design as a Cultural Practice*. The MIT Press.

Norman, D. A. (1988). *The Design of Everyday Things*. Basic Books.

Norman, D. A. (2004). *Emotional design: Why we love (or hate) everyday things*. Basic Books.

Picard, R. W. (2000). *Affective computing*. The MIT Press.

Ramey, C. (2005). The uncanny valley of similarities concerning abortion, baldness, heaps of sand, and humanlike robots. Paper presented at the Proceedings of the "Views of the Uncanny Valley: workshop, IEEE-RAS International Conference on Humanoid Robots, Tsukuba, Japan.

Savitch, W., & Mock, K. (2016). JAVA (7th ed.). Pearson Education.

Shneiderman, B. (2010). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Pearson.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2016). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (6th ed.). Pearson.

Umbaugh, S. E. (2023). *Computer Vision and Image Analysis: Digital Image Processing and Analysis*. CRC Press.

Appendix A – Code

Main Class

```
import toxi.geom.*;

import toxi.util.events.*;

import toxi.physics2d.*;

import toxi.physics2d.behaviors.*;

import toxi.physics2d.constraints.*;



import gab.opencv.*;

import processing.video.*;

import java.awt.*;

import java.util.*;



Capture cam; //creating the camera object
```

```
OpenCV opencv; //creating the OpenCV object

int h = 900; //height of our sketch

int w = 1600; //width of our sketch

int NUM_OF_STARS = 2000; //number of stars that will be initially be drawn

int MAX_SIZE = 4; //the maximum size of the static stars

int MAX_FALLING_SIZE = 7; //the maximum size of the falling stars

float r, xvar, yvar; //the r variable is for controlling the percentage of stars and fallen stars created. The xvar and yvar are for creating random coordinates for them



int deadStars;



ArrayList<Stars> stars = new ArrayList<Stars>(); //the array that will be used to store all the stars



VerletPhysics2D physics; //our physics world. This is where every object that has to do with the simulation, will be added
```

```
AttractionBehavior2D faceAttractor; //the attractor object. or in our case, the repelor object

Vec2D facePos; //vector object of toxiclibs, that will be used to manipulate the attractor



void setup(){

  size(1600,900);

  surface.setLocation(20, 20);

  //fullScreen();



  cam = new Capture(this, w/2, h/2); //we halve the quality of the camera, so the sketch runs faster

  opencv = new OpenCV(this, w/2, h/2); //same here

  cam.start(); //starting the camera object

  opencv.loadCascade(OpenCV.CASCADE_FRONTALFACE); //setting up OpenCv to detect a face when the user looks at the camera straight on
```

```
/*creating the physics of the world, setting the bounds of this world and adding the gravity

all of these are added to the physics system we cerated in the beginning*/

physics = new VerletPhysics2D();

physics.addBehavior(new GravityBehavior2D(new Vec2D(0.06, 0.1))); //adding the gravity in our system



//creating the stars and filling our array initially with 70% static stars and 30% falling stars

for (int i = 0; i < NUM_OF_STARS; i++){

    r = random(1);

    xvar = random(width);

    yvar = random(height);

    if(r <= 0.7){

      Stars star = new Stars(xvar, yvar);

      stars.add(star);
```

```
    } else {

      Stars star = new FallingStars(random(width), random(height));

       stars.add(star); }

   }

}



void draw(){

  background(0);

  scale(2); //because we halved the quality in the beginning, we now upscale it by 2. This makes the draw function work faster

  opencv.loadImage(cam); //"feeding" the camera live image in OpenCv



  //starting the camera and reading from it

  if (cam.available() == true) {
```

```
  cam.read();

 }



  //using iterators is a good way to control our array lists dynamically, as we can loop through them and remove contents from them and the iterator takes care of everything,

  //like not leaving blanks and reading the next object and knowing when the list is ending, despite us adding and removing things

  deadStars = 0;

  Iterator<Stars> it = stars.iterator();

  while(it.hasNext()){

    Stars d = it.next();

    d.getPos();

    if (d.isDead())

    {

      it.remove();
```

```
    deadStars++;
  }
}

//creating as many stars as we previously removed
if (deadStars > 1)
{
  for (int i=0; i<deadStars; i++)
  {
    Stars starBorn = new FallingStars(random(width), random(height));
    stars.add(starBorn);
  }
}
```

```
//displaying the stars

for (Stars s : stars){

   s.display();

}



fill(0);

stroke(0, 0, 0);

Rectangle[] faces = opencv.detect(); //Rectangle[] is an OpenCV object and is used to create a rectangle around the center of the face



for (int i = 0; i < faces.length; i++) {

   ellipse(faces[i].x + faces[i].width/2, faces[i].y + faces[i].height/2, faces[i].x/2, faces[i].y); //I take the rectangle and create an ellipse. The split by 2 in some variables, is what I ended up on after testing

   facePos = new Vec2D(faces[i].x, faces[i].y); //creating a vector that points at the position of the center of the face
```

```
    /*Creating the face attractor that collides with the particles and repells them back. The attractor is an object from toxiclibs.

    For it we need to set which object will be used as the attractor (the face, so we use the face position), the radius of its effect (since we simulate a collision,

    the radius needs to be small) and finally the attractors strength(the random is random and picked after trying many others)*/

    faceAttractor = new AttractionBehavior2D(facePos, 10, -20f);

    physics.addBehavior(faceAttractor); //Finally we need to also add in our physics world the attractor.

    physics.update();

  }

  physics.update(); //at the end of every run of draw(), we need to update the objects that are in our physics world

}



void keyPressed(){
```

```
  //I was trying to figure out how I could generate easily more falling stars whenever I wanted, so I figured that the keyboard was a good solution

  //Whenever someone hits the spacebar, a pseudorandom amount of stars are generated

  if (key == ' '){

    for (int i = 0; i < random(500); i++){

        Stars star = new FallingStars(random(width), random(height));

        stars.add(star);

    }

  }

}
```

Stars Class

class Stars extends VerletParticle2D { //by ectending the Stars class with VerletParticle2D, Stars inherit all the functions from VerletParticle2D

VerletParticle2D star; //the stars will in essence be VerletParticles

float starSize; //size of the stars

float transparency; //transparency of the stars. Depends on their size. The closer to the screen the more "bright" they are

Vec2D starPos;

float posx, posy; //will be used to get the current position of the stars

Stars(float x, float y){

super(x, y); //calling the super constractor that is inherited from VerletParticle2D

```
  star = new VerletParticle2D(x, y);

  this.starSize = random(MAX_SIZE); //creating random star sizes within the limits that we set

  this.transparency = (250 / MAX_SIZE) * this.starSize; //the 250 is a random number that was just giving me nice results

  starPos = new Vec2D(x, y);

 }



 //the display functions creates the stars and makes them bigger and more bright if they are close and smaller and more transparent the further away they are

 void display(){

  stroke(this.transparency);

  strokeWeight(this.starSize);

  point(this.star.x, this.star.y);

 }

 //a getter function, so we know the current position of the stars
```

```
  void getPos()

  {

    posx = this.star.x;

    posy = this.star.y;

  }

  //checking if the stars are out of bounds. The static stars are always drawn inside the screen, so this function is a bit useless for them

  //however, we need it, so we can overwrite it when we check the position of the falling stars

  boolean isDead(){

    return false;

  }

}
```

FallingStars Class

class FallingStars extends Stars{ //extending the FallingStars class with the Stars functions, it means that it inherits everything from the Stars class

```
  FallingStars(float x, float y){

    super(x, y); //the inherited constructor

    this.starSize = random(MAX_FALLING_SIZE); //picking a random size. The maximum size is set in the main class, since this is a global variable

    physics.addParticle(star); //by adding the falling stars to the physics world, we make them affected by the forces in it. The gravity. That means they are pulled down by it and they can also interact with the attractor

  }



  void display(){

    stroke(this.transparency);

    strokeWeight(this.starSize);
```

```
    point(this.star.x, this.star.y);
  }


  //ovewritting the isDead() function to check the position of the stars
  boolean isDead(){
    if (posx > width && posy > height || posx > width || posy > height){
      return true;
    } else {
      return false;
    }
  }
}
```

Author's Statement:

I hereby expressly declare that, according to the article 8 of Law 1559/1986, this dissertation is solely the product of my personal work, does not infringe any intellectual property, personality and personal data rights of third parties, does not contain works/contributions from third parties for which the permission of the authors/beneficiaries is required, is not the product of partial or total plagiarism, and that the sources used are limited to the literature references alone and meet the rules of scientific citations.