Technical University of Munich

Automatic Integration of Procedurally Generated Environments as Context in Automotive Design Experiences

Bachelor’s Thesis in Informatics: Games Engineering

Scientific Work to Obtain the Degree

B.Sc. in Informatics: Games Engineering

Department of Informatics, Technical University of Munich.

Submitted by Ata Selim Berktürk

Automatic Integration of Procedurally Generated Environments as Context in Automotive Design Experiences

Automatische Integration von prozedural generierten Umgebungen als Erlebniskontext im Automobildesign
Bachelor’s Thesis in Informatics: Games Engineering

Scientific Work to Obtain the Degree
B.Sc. in Informatics: Games Engineering
Department of Informatics, Technical University of Munich.

Advisor Despoina Salpisti
Research Group for Augmented Reality

Supervisor Prof. Dr. Gudrun Klinker
Research Group for Augmented Reality

Submitted by Ata Selim Berktürk

Filed On 17.01.2022, Munich

Table of Contents

1 Background Research 1

1.1 Human-Computer Interaction 2

1.1.1. Understanding Human-Computer Interaction 2

1.1.2. Human Psychology 5

1.1.3. Design 6

1.1.3.1. Usage of Color 7

1.1.3.2. Information Grouping 8

1.1.3.3. Navigation 9

1.1.3.4. Design Rules 10

1.1.3.5. Principles to Support Usability 12

1.1.4. Place and Importance of Iteration in human-Computer Interaction ……………………14

1.2 Usability …………………………………………………………………………..15

1.2.1. Defining Usability ……………………………………………………………15

1.2.2. Testing and Evaluation …………………………………………………..17

1.2.3. User Characteristics ……………………………………………………...19

1.2.4. Components of Usability……………………………………………………20

1.2.5. Principles of Usable Design ………………………………………………..22

1.2.6. Measures of Usability ……………………………………………………….23

1.2.7. User Experience ……………………………………………………………25

1.3 Aesthetic Design in Automotive Industry ……………………………………..27

1.3.1. The Automotive Industry ……………………………………………………27

1.3.2. Aesthetics in the Automotive Industry ……………………………………28

1.3.3. Virtual Prototyping …………………………………………………………..28

1.3.4. State of the Art ……………………………………………………………29

2 Motivation 30

3 Materials and Methods 31

3.1 Menu Design 31

3.1.1. Color Palette 32

3.1.2. Text Fonts 33

3.1.3. Page Indicators 33

3.1.4. Futuristic Earth Image 35

3.1.5. Common Background Image 37

3.1.6. Areas of Improvement 37

3.2 Menu Sections 38

3.2.1. Time- and Weather Configuration 39

3.2.1.1. User Interface 39

3.2.1.2. Geaographically-Accurate Sun Placement 40

3.2.1.3. Weather Conditions 40

3.2.1.4. Areas of Improvement 41

3.2.2. Metahuman 42
3.2.2.1. User Interface 42
3.2.2.2. Working with Metahumans 44
3.2.2.3. Metahuman Creator 44
3.2.2.4. Challenges with Metahuman Creator 46
3.2.2.5. Importing Metahumans into an UE4 Project 46
3.2.2.6. Animating Metahumans 46
3.2.2.7. Inverse Kinematics (IK) 47
3.2.2.8. Metahuman Characters 48
3.2.3. Vehicle- and Camera Configuration 49
3.2.3.1. Vehicle System 50
3.2.3.1.1. User Interface 50
3.2.3.1.2. Driving System 51
3.2.3.1.3. Animating a Vehicle 51
3.2.3.1.4. Metahumans in Vehicle System 52
3.2.3.2. Camera System 52
3.2.3.2.1. User Interface 53
3.2.3.2.2. Pre-Defined Camera Modes 54
3.2.3.2.3. Custom Camera Mode 55
3.3. Back-End Implementation 56
3.3.1. Game Instance Blueprint 56
3.3.2. Data Structures 57
3.3.3. Displaying a View from an UE4 Level in the User Interface 57
3.3.4. Hotkeys for Real-Time Controls 58
3.3.5. Mapping Terrain Coordinates into Latitude/Longitude Format 59
3.3.6. Areas of Improvement 60
4 Results and Discussion 61

Abstract

In our work we developed a system for the evaluation process of automotive aesthetic design industry. Our system provides automotive designers an interactable 3D environment in which design prototypes can be evaluated, and a debatably usable user interface. Our work provides a detailed explanation of creating an evaluation tool by utilizing Unreal Engine 4 (UE4) and other state-of-the-art technology. For the purpose of achieving usability and thus providing a positive experience to users, we studied the subjects ‘Human-Computer Interaction’, ‘Usability’ and ‘Aesthetic Design in Automotive Industry’. By doing so, we developed a comprehension of where ‘human’ stands in interactions between users and devices, and which concepts must be taken into account to be able to attain usability. We aimed to understand the state-of the-art in the automotive aesthetic design industry for the purpose of understanding the industry and thus being able to review our system’s position more realistically.

1. Background Research

Our research is mainly based on the topics ‘Human-Computer Interaction’ and ‘Usability’. We intended to understand the theory behind these subjects, their evolution since the early ‘70s, and to apply their rules whilst developing our project.

The reason that we specifically selected these topics and took cognizance of implementing their principles is that our project consists of a user interface and an interactable 3D environment. This means that the quality of the interaction between designers and our system will influence the desirability and acceptance of the generated interface/tool from the automotive designers. For the purpose of reaching the highest level of interaction quality, the fundamentals of human-computer interaction must be fully comprehended and the essentials of the term ‘product usability’ as well as its guidelines on achieving usable design results must be studied and applied.

Our target user group consists of car designers who work to accomplish functional, ergonomic and aesthetic designs in a competitive industry, namely the automotive industry, while aiming to maintain and further-cultivate a unique brand identity. Therefore it was significant to support our knowledge with insight on the automotive industry: the competitive environment of the industry itself, the place of design and aesthetics in the industry, and the demand, benefits and current state of virtual prototyping in aesthetic design process. With the intention of reaching such insight and thus being able to contribute to the automotive aesthetic design sector by producing an innovative virtual prototyping tool, we investigated state-of-the-art technologies in terms of their benefits and shortcomings.

In the interest of being able to assist our own review on the potential place of our product in the automotive industry with analytical data, executing a qualitative evaluation of users’ experience with our system has also been among our intentions since the very start. Therefore we also studied the fundamentals of user experience and the core elements of measuring, evaluating and testing usability aspects of a software system. This helped us in both having a better comprehension about what must be taken into account while developing a usable product and realizing an insightfully executed evaluation of user experience.

1.1. Human-Computer Interaction

In this section we will explain the subject ‘Human-Computer Interaction’. We will elaborate by firstly defining the subject at hand. We will briefly describe the subject’s historical evolution from the ‘70s until today in terms of its focus and concerns and depict its place in industry.

In order to attain a better grasp of the ‘Human’ component of Human-Computer Interaction, we will continue with analyzing in what way human psychology has been a part of this subject. Following human psychology, we will give detailed explanations about the rules of design that Human-Computer Interaction has been originating since its foundation; the fundamentals of the ‘Interaction’ component of the subject, most specifically about menu-based user interfaces, core principles for task simplifications which provide assistance in increasing the ease-of-use of a user interface, and some essential rules of designing a user interface.

Subsequently, we will examine the subject ‘Iteration’ from the perspective of Human-Computer Interaction in summary for the sake of emphasizing the importance of testing and redesigning in the software development cycle and thus stressing the significance of user experience evaluation. Before finalizing our studies in Human-Computer Interaction and moving on to the subject ‘Usability’, we will concisely outline a number of principles to support usability aspects of a product.

1.1.1. Understanding Human-Computer Interaction

In the year 2022, it is possible to say that technology has surrounded our daily and business lives to an extremely high extent. Computers are now used in nearly all professional jobs [1] and on top of that we use numerous devices in order to accomplish daily tasks. These tasks my range from complex ones, such as computation or transportation, to exceedingly simple examples, such as setting up digital calendars, setting reminders, shopping and etc.

Customers clearly do not choose any device to include them in their daily or professional lives. Their selection process entirely depends on the product’s aspects and the customer themselves.

First of all, users must be motivated in order to perform properly. [2] We will analyze the factors which might or might not motivate a user towards using a product in the following sections, but in general it can be said that these factors might range from the price, function or the aesthetics of a product to the cultural background of the user.

Second of all, when the usage of products in users' business life is observed, the user's job satisfaction and, as a result, performance will suffer if a system makes it difficult or frustrating for a user to execute critical tasks. [2] In this context, the term 'system' can refer to many types of products which users employ in their professional lives, e.g., copy machines, coffee makers, or software products. As an arguably natural result, experts in human-computer interaction may be found in nearly any technological company of any size. [3] Although coffee makers or copy machines might be recognized as, probably due to their simplicity, products distant from computers, they are still digital items which users interact with, therefore the properties of the interaction that they offer may determine whether if they get chosen by users. In all likelihood, this emphasizes the place and importance of human-computer interaction experts in the technology world.

"Human-Computer Interaction is a term used to refer to the understanding and designing of different relationships between people and computers." [3] On the whole, it can be stated that our subject at hand, namely human-computer interaction, is a relatively old topic. It has been studying various topics, and its main concerns have been evolving ever since, along with the rapid expansion of technology. Nevertheless, human-computer interaction as a whole, takes unsatisfactory designs and demonstrates how to make them better. It also seeks to use its methods to create good systems from the beginning. [3]

When HCI occurred in the late 1970s, the primary focus was 'usability.' Since its inception in the 1980s, HCI has been largely focused with constructing more usable computer systems, whether it be the computer desktop, the VCR, the Web, or the mobile phone.[3] As we mentioned above, with the rapid growth of technology, human-computer interaction's focus has shifted to the most recent topics and concerns that came along with the rise of technology. In fact, we can even claim that, because the place of technology in our lives is now so centralized, that recognizing what it means to be human in a digital world is at the heart of the new agenda of human-computer interaction. [3] With a view to elaborate on the current extent of the concerns, with which human-computer interaction is presently occupied, we can take the ubiquitous state of technology. For example, as we age, our desire for vitality and independence may lead us to implant medical equipment near or even within our bodies. [3] Hearing-aid devices and heart batteries are textbook examples of this matter.

In summary, the place of the computer interface or point of contact, as well as the extent to which it is visible to us, is no longer as apparent as it was when we interacted at the desktop or terminal, and as a result, what an interface might be, where it is, what it allows a user to accomplish, and even whether one exists at all, are all questions for a forward-thinking HCI right now. [3] Consequently, human-computer interaction must recognize and analyze a broader set of concerns presently at play, most notably human values, including the moral and ethical elements of developing technologies for new areas. [3]

As to demonstrate the future of human-computer interaction, we will discuss how it should keep up with the modern concerns which escalate due to the most recent developments in technology. We will follow the suggestions from R. Harper’s (2008) book: ‘Being human: human-computer interaction in the year 2020, (2008)’. [3]

The first suggestion is to refine the fundamental ideas of HCI, such as 'human,' 'computer,' and 'interaction'. This is arguably a very open-ended statement, so for the sake of clarifying confusions: our interpretation of this suggestion is to firstly redefine the term ‘human’ so that it fits the most recent concerns of modern society better and puts the average user in a more adequate position in terms of capability and perspective for technology. Secondly it is suggested to reformulate the term ‘computer’ in order to incorporate the most recent trends and standards, e.g. augmented- and virtual reality, metaverse or smartphones. As the third, it is recommended to revisit the term ‘interaction’ with the intention of better incapsulating the ubiquitous usage of smartphones, home assistants and other various interfaces, e.g. car dashboards or gaming devices.

The second suggestion is to integrate another level of conceptual analysis into user-centered research and design that directly addresses the broader spectrum of societal and moral themes; such as privacy, health, ownership, fair play, and security. We can probably include other topics like sexism and racism, as these subjects are currently issues which modern society is debatably sensitive about.

The third and final suggestion that we will present is to establish new collaborations with fields that aren't traditionally associated with HCI but are capable of addressing societal, moral, and ethical issues. The implementation of this suggestion can possibly be an approach that also embraces the first two suggestions, as the involvement of new disciplines might be the key to redefine core concepts of human-computer interaction in a way that addresses society’s modern concerns more adequately.

1.1.2. Human Psychology

In all probability, psychology plays an arguably significant role in obtaining a strong grasp of the ‘human’ part of human-computer interaction as users’ perception and experience, therefore a relatively big portion of their interaction with products, depend on their psychological state. We will initially define the position of ‘human’ in human-computer interaction. We will continue by describing the significance of taking human psychology into account during a design process and conclude by presenting a suggestion on the ideal way to apply psychology in design industry according to S.K. Card.’s book: The Psychology of Human-Computer Interaction, (1983). [4]

Figure 1 is adapted from the ACM SIGCHI Curriculum Development Group [5] and represents the role of ‘human’ in its interaction with the computer. The human, in this approach, is viewed as an information processor, with inputs (mostly visual), mental processing, and outputs (keyboard strokes, mouse activities, and so on), which then 'input' information or data into the computer. [3] It can be argued that this representation simplifies the position of humans in their interaction with computers and therefore provides a basis for building and elaborating principles for designing usable interfaces.

One could argue that applying psychology in design process is a rather difficult task. In design, the system is still mostly hypothetical; it is a group of systems… Thus, for instance, applying psychology to system evaluation is unquestionably easier than applying it to system design. [4] However, despite the difficulties and uncertainties of applying psychology in the design process, it is in all likelihood the design process where the

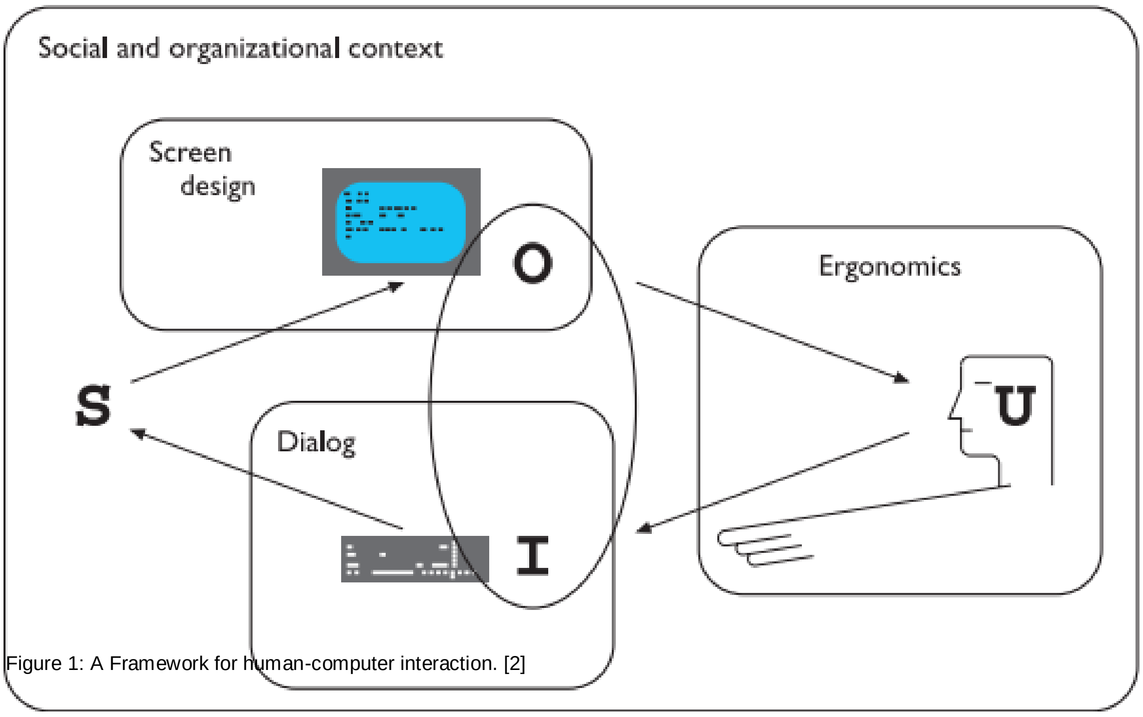


Figure 1: A Framework for human-computer interaction. [2]

application of psychology is at its most constructive and effective. There are enough degrees of freedom during design to make a difference and when applied psychology is used at a later time, it is doomed to have a limited impact. [4]

A potential approach to correctly applying psychology in the design process could be to unite the psychology units with designers in the industry. Separate psychology units are not preferred due to the additional separation that they indicate between psychology and interface development… The application of psychology would stray too far away from the major design processes and toward evaluation. [4] Therefore the major professionals—computer system designers—should be the primary agents for applying psychology, and the system designers should become specialists in the applied psychology of human-computer interfaces. [4]

1.1.3. Design

In this section we will define and explain several principles of interaction design. As people are becoming increasingly irritated by things that are difficult to use [1], we need to understand and design for interaction in a world where the concept of an interface is no longer simply defined, stable, or fixed. [3] Perhaps we can say that, only by applying the extensively formulated guidelines which were originated in human-computer interaction as technology has come to its most recent state, products can be designed to be functional, ergonomic and easy-to-use.

We decided to highlight design guideposts for creating usable menus because in menus the options are visible, which makes them rely on recognition rather than recall. [2] The inclusion of retrieval cues aids recall, therefore, interfaces should include recognizable cues whenever possible… So ultimately, menus make interfaces less exhausting for the user. [2] This is why we hold graphical menus as the core subject on which we will formalize our research and opinions about usable design.

We will elaborate on usable menu design by firstly giving a brief description of the usage of color, continue with introducing several guidelines on information grouping strategies which encapsulate contriving screen layout and visually presenting information, and afterwards describe how to offer users a correct course of navigation throughout the menu and briefly define the consequences if the delivery of a successful navigation component of a menu fails.

After having examined rather practical approaches for usable menu design, we by continue with explicating more theoretical concepts about design in human-computer interaction. We will start by defining different types of design rules. Subsequently, we will expose certain rules on interface design.

Ultimately, we will advert a number of specific principles to support usability and finalize by depicting the importance of evaluation and iteration in the design process.

The following definitions and explanations of design in human-computer interaction are based on the book: ‘Human-Computer Interaction, (1993)’, written by the author A. Dix [2]. We also give place to our comprehension and inferences about the topics with the intention of providing complete explanations.

1.1.3.1. Usage of Color

The use of color in displays is a problem in terms of ergonomics. It wouldn’t be wrong to claim that the color palette of a user interface is one of the important factors which determine whether it will be chosen by users to be regularly used. Although it might differ from one user to another, colors may be associated with certain feelings by users. Users’ cultural and professional background, as well their age and gender, could be several factors, among many others that play a role in shaping which feelings colors will be associated with. When it comes to color, the visual system has a number of standard constraints:

1. The colors used in the display should be as distinct as possible,
2. the number of identifiable colors should be considered,
3. the distinction should not be altered by changes in contrast and
4. the sharpness of the color blue should be at a relatively low concentration.

Besides the standardized constraints on the usage of color, there is also a simple guideline on the associated feelings that are invoked by the usage of several colors: red, green, and yellow are colors generally linked with stop, go and standby, respectively. As a result, red can be used to represent an emergency or alarm, green can be used to indicate regular activity, and yellow can be used to indicate a standby or auxiliary function.

We may state that following this insight on color usage could, in all likelihood, make a display more usable and might elicit more favorable emotions in the user.

1.1.3.2. Information Grouping

In this section, we will define three information grouping strategies and introduce certain guidelines on what should be taken into consideration while positioning information blocs on menu displays.

The first and debatably most primary idea is that things should typically be physically grouped together if they logically belong together. We can expand on the phrase ‘things logically belonging together’ by going through three types of possible information grouping strategies. These strategies questionably elucidate the uncertainty in logically correlating ‘things’ in menu displays. Those strategies are defined as the following:

- **Functional**: The controls and displays are arranged in such a way that those that have a functional relationship are grouped together.
- **Sequential**: Controls and displays are arranged in the order in which they're used in a typical interaction (this is especially useful in domains where a specific task sequence is required).
- **Frequency**: Controls and displays are arranged based on how often they are used, with the most frequently employed controls being the easiest to access.

We will firstly attend to the matter ‘White Space’ while presenting the guidelines on placing content on menu displays. ‘White Space’ refers to the ‘counter’, the space between the interface elements, on menus, so in other words, it is the empty areas which separate different groupings in a menu layout. The function of white space is that users can develop a feeling of the layout as a whole if they concentrate on the counter rather than the 'content' of the screen.

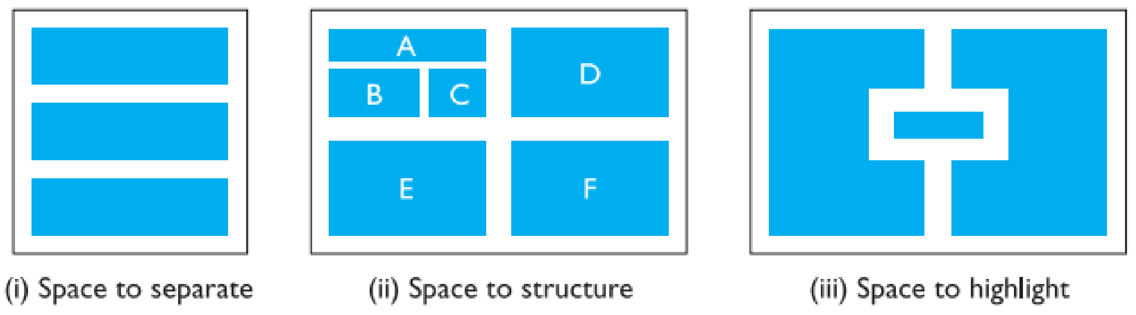


Figure 2: Using White Space in Layout. [2]

In Figure 2 three different usages of white space and their intended purpose is depicted. We can probably finalize by stating that menu displays can be designed to be more usable by utilizing the white space on the layout correctly because this makes it easier for designers to convey controls and information to the user in a deliberate manner.

1.1.3.3. Navigation

The term ‘navigation’ in menus refers to users knowing where they are, what they can do, where they’re going, and where they’ve been in terms of the state of the interface and their interaction.

Users will feel more in control and understand how to navigate the information environment if they can always answer each of these questions. The experience of being disoriented when users don't have enough information to know where they are and where they've been is referred to as ‘lost in hyperspace’.

The display on the screen, web page, or device should make it apparent where the user is in terms of the system's interaction or state. For example, users must be able to grasp what will happen when a button is pressed. One way of achieving this is to always remember that icons and shapes are rarely self-explanatory and should always be accompanied by labels or, at the very least, tooltips or some other comparable technique. It must also be clarified, if the menu display contains images, which images are purely decorative and which are interactive links, in order to prevent confusion.

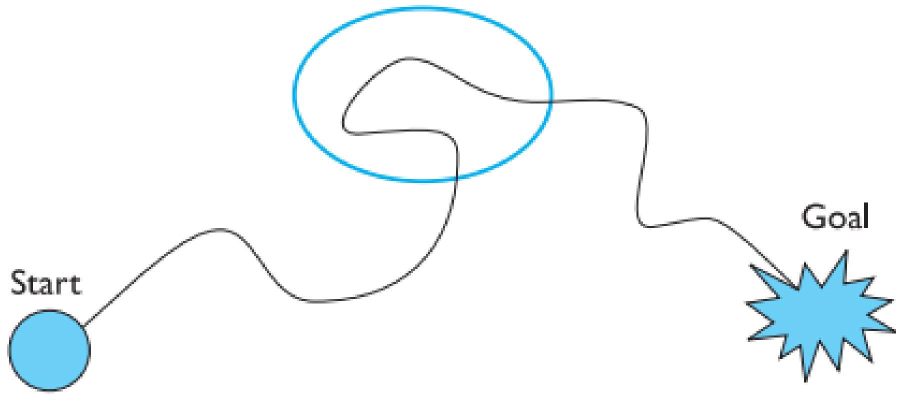


Figure 3: Visualization of a User's Navigation with Partial Knowledge. [2]

Users, on the other hand, meander through the system in an environment of partial knowledge. The crucial element is that they can judge if they are moving closer to their (usually partially formed) goal at each stage in the encounter, rather than taking the most efficient route. To do this goal seeking, each state of the system or each screen must provide the user with sufficient information about how to get closer to their goal.

The last subject about navigating a user interface which we will address is navigating in three dimensions. Especially due to the rise of augmented- and virtual reality, it could be said that the frequency of human-computer interaction in three dimensional environments is increasing, regardless of whether the user group is professionals or consumers. Humans require additional assistance when navigating in three dimensions. If we have a 3D interface or a virtual reality world, for example, we should present a ground plane and lock movement to be parallel to the ground by default.

1.1.3.4. Design Rules

In this section we will explain what design rules are by defining the term and presenting different types of design rules. Finally, we will introduce Schneiderman's eight golden rules of interface design, which present a concise and easy-to-understand overview of the main principles of interface design.

Design rules are methods that limit the range of design options available to a designer, preventing them from pursuing design options that are likely to result in an ineffective system. Design rules are used to put the theory into practice. A set of design rules will frequently be in conflict with one another, making strict commitment to all of them difficult. We find understanding these characteristics of design rules to be crucial for properly applying them in design because they arguably define the range and impact of design rules' application.

The different types of design rules provide a classification based on the rule's authority and generality. In this context, authority refers to whether the rule has to be followed in design or if it is only recommended, and generality indicates whether the rule may be applied to a wide range of design scenarios or whether it is more focused on a specific application scenario.

These types are defined as the following:

1. **Principles** are high-generality, low-authority abstract design rules. They are largely independent of technology and are formed from knowledge of the psychological, computational, and sociological elements of the subject domains.

2. **Standards** are specified design rules with a high level of authority and a narrow scope of application.

3. **Guidelines** have a lower level of authority and a broader scope of application. They are less abstract and, commonly, more technology oriented, but because they are also general, a designer must be aware of the theoretical reasoning that supports them.

In the following, we will introduce Schneiderman's eight rules on interface design, which could be assessed as high in generality, since these rules do not constrain their application to some domain or device, and high in authority because the rules debatably encapsulate a wide range of design aspects and the successful results of their usage can possibly seen in modern interfaces. Although if we one focuses on the fact that these rules are in fact general suggestions, these rules might as well be considered as low-authority rules.

These rules are presented and briefly expanded in the following:

Schneiderman's Eight Golden Rules of Interface Design [2]

1. *Strive for consistency* in action sequences, layout, language, and command usage, among other things.

2. *Enable frequent users to use shortcuts*, for instance, abbreviations, custom key sequences, and macros, to speed up the execution of routine, familiar actions.

3. *Offer informative feedback* for each user action, at a level commensurate with the action's magnitude.

4. *Design dialogs to yield closure* so that the user can tell when they've finished a task.

5. *Offer error prevention and simple error handling* so that users are ideally avoided from making mistakes and, if they do, are given clear and instructive directions on how to recover.

6. *Permit easy reversal of actions*. This way the user knows that they can always return to the prior state which helps in easing stress and promoting exploration.

7. *Support internal locus of control* so that the user has control over the system, which reacts to their actions.

8. *Reduce short-term memory load* by keeping displays basic, unifying multiple page displays, and giving time for action sequences to be learned.

We suggest doing further reading in A. Dix's 'Human-Computer Interaction, (1993)' [2] for the purpose of clarifying potential confusions due to the employed terminology while defining and describing Schneiderman's rules of interface design.

1.1.3.5. Principles to Support Usability

In this section, we will present a number of principles which assist in designing usable user interfaces. As elaborated in the previous section, namely 1.1.3.4, principles are design rules with a high level of generality and a low level of authority, which means that the rules which we will not be rules to which designers should strictly devote their design process. Instead, the ideas behind the following suggestions should be understood and brought to bear while designing interfaces.

We will start with introducing principles which should help in increasing the ease-of-use of an interface by presenting 'Norman's Seven Principles for Transforming Difficult Tasks into Simple Ones'. We decided to exclude explanations of two of Norman's principles because of their irrelevance to our studies. We will finalize this section by defining three core categories of numerous principles about usability and subsequently explain how these principles should be applied in design for the purpose of achieving usable design. The terms and topics which will be discussed in this section will be elaborated on in the 'Usability' section, namely 1.2. Our selection of Norman's principles of task simplification is listed and explained in the following:

Norman's Seven Principles for Transforming Difficult Tasks into Simple Ones [2]

1. *Use both knowledge in the world and knowledge in the head.* Systems should give the essential knowledge within the environment, and executed actions should be transparent, allowing the user to construct an appropriate mental picture of what is happening.

2. *Simplify the structure of tasks.* To minimize difficult problem solving and excessive memory load, tasks should be simple. Delivering mental aids to assist the user in keeping track of steps, offering better feedback, and automating the task or a portion of it could all be strategies for achieving successful simplification.

3. *Make things visible.* The user interface should make it apparent what the system can do and how it does it, as well as allow the user to perceive the impact of their actions on the system.

4. *Design for error.* Errors must be expected, and the system must be designed regarding error recovery.

5. *When all else fails, standardize*. If normal mappings are lacking, artificial mappings should be standardized so that users only need to learn them once.

6. *Get the mappings right.*

7. *Exploit the power of constraints.*

Moving forward, we will categorize various principles into three groups, namely ‘learnability’, ‘flexibility’ and ‘robustness’, and explicate these principles. We will only give place to the principles which are related to our studies. Following categorization is presented with reference to A. Dix’s book: ‘Human-Computer Interaction, (1993)’ [2].

1. **Learnability**: Learnability refers to the elements of an interactive system that enable new users to learn how to use it at first and eventually to achieve maximum performance. Summary of some principles that affect learnability is listed in the following:

   a. *Predictability*: Support for the user to predict the outcome of future actions based on previous interactions.

   b. *Familiarity*: The amount to which a user's knowledge and expertise from other real-world or computer-based domains may be used when dealing with a new system.

   c. *Generalizability*: Support for the user to apply what they've learned about a certain interaction within and across applications to other scenarios that are comparable.

2. **Flexibility**: Flexibility refers to the various ways in which the end-user and the system might interact.

Summary of a few principles that affect flexibility is listed in the following:

   a. *Customizability:* The user interface's capacity to be modified.

   b. *Multi-threading:* The system's ability to allow simultaneous user involvement with several tasks.

3. **Robustness***:* Robustness is the level of assistance supplied to the user in determining goal achievement and evaluation.

Summary of a few principles that affect robustness is listed in the following:

a *Observability:* The user's ability to assess the system's internal status based on its observed presentation.

b. *Recoverability:* The user's ability to take corrective action once a mistake has been identified.

1.1.4. Place and Importance of Iteration in Human-Computer Interaction

In this final section about human-computer interaction, we will shortly explain the importance of prototyping and iteration for human-computer interaction. Although the significance of testing, evaluating, and iterating will be elaborated in section 1.2, namely Usability, we intend to display the emphasis on iteration from the perspective of human-computer interaction. We will concisely describe the reason to iterate, what the iteration process consists of, and finally, define two different types of evaluation.

This section, similarly to section 1.1.3, namely Design, is mainly based on the explanations from the book: 'Human-Computer Interaction, (1993)', written by the author A. Dix [2]. We also included our inferences in order to consolidate the information from the book.

An interactive system's requirements can't be fully described at the start of its life cycle. The only way to be certain about key aspects of a proposed design is to build it and test it on real users. The evaluation of the system normally yields a list of deficiencies or problems, which is then followed by a redesign exercise, which is then prototyped and evaluated. Therefore it can be said that the cycle of prototyping, evaluating, redesigning according to the results of the evaluation, and then developing a new prototype based on the most recently conducted design can be repeated until perfection is reached. Since reaching such an ideal state of a product is in all likelihood impossible due to timely and costly constraints, an end goal in terms of quality and satisfiability must be set in the beginning, and designers should be open to revise this end goal according to the design and development phase.

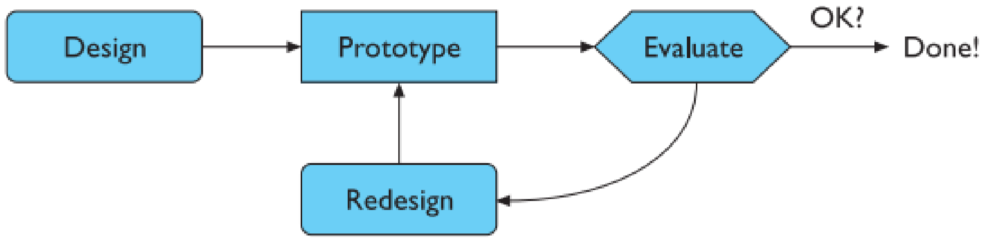


Figure 4: Role of prototyping. [2]

One way to evaluate prototypes is called a ‘Formative Evaluation’: and it is for improving designs. The second way to evaluate prototypes is called a ‘Summative Evaluation’: which is intended to verify whether a product is satisfactory in the end.

1.2. Usability

In this section, we will study the term ‘Usability’ in detail. We will start with defining the term by providing a few different perspectives on how to describe usability and displaying its importance by means of the consequences of its absence. We intend to introduce different perspectives on the term’s definition, which can probably be simply categorized as formal and informal, and mean to set out the results of the lack of usability because the meaning of the term, which subthemes it encapsulates, how it should be measured and evaluated, and many other aspects of usability can vary depending on how one comprehends the term. Therefore, we believe that presenting different perspectives on usability from the beginning will assist in defining and explaining the various aspects of usability in detail.

We will continue with examining testing and evolution in terms of usability. We will define the goals of usability testing and the goals of usability testing, and subsequently describe how usability should be evaluated by shortly explaining several techniques on how a diagnostic evaluation of user experience should be conducted.

Afterwards, we will proceed to explain how the characteristics of a user group can affect the usability of a product.

After that, we will define components of usability and principles of usable design for the purpose of elaborating on usability. The topics discussed in section 1.1.3.5, namely Principles to Support Usability, will be detailed.

Before finalizing this section by defining what user experience is and explicating the concerns and objectives which keeps user experience apart from usability, we will describe the measurements of usability through a number of perspectives.

1.2.1 Defining Usability

Indeed, usability is something that today's consumers want from a product. [1] If usability is defined in an ease-of-use oriented perspective, we can, in all probability, say that modern users’ decision on choosing a product depends on the usability of the product, as people probably tend to choose the products which are more convenient and practical with the intention of integrating the usage of these products in their daily- or professional lives.

The dramatic growth in the number of specialists employed by the industry who are responsible with ensuring that products are easy to use is possibly the most important indication of how seriously usability issues are now being regarded… Human factors experts and interaction designers are among them… Furthermore, product designers and software programmers are increasingly expected to be aware of usability issues and to prioritize the user during the design process. [1]

Usability is the user's perception of software quality for a software product. [6] But the term ‘perception’ is probably rather unclear, so in order to clarify this uncertainty, we will start depicting usability by providing an ease-of-use oriented description: usability should be defined as a product's ease of use and acceptability for a specified group of users performing certain tasks in a specified context. [6] The following definition can be employed with the intention of illuminating possible confusions about the term ‘ease-of-use’: Difficulties and errors can be recognized, classified, recorded, and measured, thus an adequate hypothesis is that ease of use is inversely related to the quantity and severity of software difficulties. [7]

Before moving on to the more formal definition of usability, we want to emphasize the significant role of interaction while describing usability.

The following explanation is based on J.R. Lewis’ technical report: ‘Usability Testing’, (2006). [7] Usability must be specific in order to be useful. It has to do with certain tasks, settings, and users.

It isn’t a property of a user or a product. There is no thermometer-like device that can determine a product's usability in absolute terms. Usability is a trait that emerges from interactions between users, products, tasks, and settings.

By inferring from the previous description, we can claim that, a product is not usable or useless in and of itself; it contains features that define its usability for a specific user, task, and environment. [6]

The definition of usability provided by ISO (International Organization for Standardization) is “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.” [7] In this definition the importance of specification while understanding usability can be seen again. The terms referred in this definition, namely effectiveness, efficiency and satisfaction, will be detailed later on in section 1.2.6, namely Measures of Usability.

One of the most essential aspects of the ISO definition of usability is that it emphasizes that usability is not merely a quality of a product in isolation, but also depends on who is using it, the purpose they are attempting to achieve, and the environment in which it is being used. [1]

The following explanations and examples of the potential consequences of the lack of usability are based on P.W. Jordan's book: 'An Introduction to Usability', (2002). [1]

Lack of usability can result in issues that may frustrate or annoy the user on one end of the spectrum and be life-threatening on the other. A daily example of the potential safety issues that could occur due to the lack of usability could be an unusable car stereo, which could endanger lives by diverting drivers' attention from the road.

Besides potential annoyance and safety problems, financial issues in terms of product sales and productivity may arise as well. First of all, the usability of products that will be used in workplace could have an impact on employee' job satisfaction. This can possibly have a negative effect on employee's productivity. Secondly, as an example to the financial issues in terms of product sales, it has been estimated that in computerized offices in the 1980s, up to 10% of working hours were lost related to usability issues. This could in all likelihood have a negative impact on product sales. On top of this, it wouldn't be wrong to claim that a decrease in productivity of employees would have a direct and detrimental impact on product sales.

1.2.2 Testing and Evaluation

In this section we will explicate testing and evaluation in the frame of usability. As we mentioned earlier in section 1.1.4, namely Place and Importance of Iteration in Human-Computer Interaction, a full list of the requirements and the end goal of a design process can't be precisely determined in the beginning of the process. Initial design ideas should be prototyped and evaluated in order to be certain about the practical results of the initially structured design and the rest of the design process should be conducted depending on the results of the most recent evaluation. The insight on testing and evaluation that we will provide in this section can probably address this matter in a solution-oriented manner, as well as emphasize on the significance of testing and evaluation.

The probable importance of testing and evaluation is that it enables designers to obtain a grasp of users' feelings and opinions about the product. Users are the endpoint where it will be determined whether the product will, in the financial manner, be successful or not. Thus, having an image of users' view on the product and regarding this image while designing can in all likelihood be crucial. After all, the user must believe that the benefits of using a product outweigh any other option for accomplishing a certain goal [6], after all, so that they choose to buy and/or utilize the product.

Usability tests might well be informal or formal… This could be explained as follows: usability tests may be in the form of conversations with the user during and after their interaction with the user where the conversations would be about their feelings and opinions about the product, or they might as well be in such form where an analytically structured questionnaire is filled by the user subsequent to their interaction with the system. [7] Nevertheless, a comprehensive test of a product will in all probability be regardless of the level of formality of the executed usability test. The measurements of users' experiences with a product are the ultimate test of its usability. [2]

The purpose of about any evaluation is not to find usability issues in a prototype… The issues that arise during the use of an application provide crucial information for product redesign… Usability testing's overall purpose is to detect and correct usability issues in products… where its fundamental purpose is to assist developers in creating more usable products… and its main objective is a rebuilt system that meets the system's usability objectives, allowing users to achieve their objectives and be satisfied with the product. [7] We can probably combine these probably rather distant statements and claim that; the detected issues during testing must be analyzed, the product must be redesigned according to the identified issues and the results of their analysis, and this process of testing and redesigning must be repeated until a certain level of usability is reached. We mentioned this loop-natured cycle in section 1.1.4, Place and Importance of Iteration, as well.

This certain degree of usability should probably be a trade-off between the costly and timely constraints and the extent to which the product can fully convey its intended functionalities to the user.

It is probably also essential to keep in mind that usability tests and evaluations are most commonly executed in a controlled setting… A controlled measurement always obscures some part of the real working environment, and interview or questionnaire procedures might provide insight into how actual usage may differ as a result of acceptance. [6] Therefore, perhaps we can conclude that designers should not dedicate their design process too strictly to the tests and evaluations where this certain difference between the controlled environment and the actual working setting is not taken into account.

We will finalize this section by introducing several evaluation techniques for executing a diagnostic usability evaluation. Diagnostic evaluations are performed with the purpose of identifying usability problems. These techniques can be described as the following and are assembled in reference to N. Bevans book: 'What is Usability', (1991) [6]:

1. *Professional Opinion:* By observing the issues, a usability expert may be able to give rapid ideas for fixes.

2. *Comprehensive Analysis of User Interaction:* The interaction record can be analyzed in conjunction with the user's description of difficulties to allow actions to be correlated with goals and subgoals, offering insight into why certain features of the interface cause trouble.

3. *Checklist Analysis of User Interaction:* Users can fill in extensive checklists. Hereby they evaluate the acceptability of various aspects of the interface, indicating certain types of issues.

4. *Checklist Analysis of Expert Interaction:* Usability experts can analyze the system's usability for pre-defined tasks by determining whether it satisfies detailed requirements outlined in checklists.

1.2.3 User Characteristics

In this section we will study the human factors that affect usability. It probably wouldn't be wrong to define these factors as the reason why products can't offer the same level of usability to every user. We will call these factors 'User Characteristics'. The user characteristics are defined and depicted in reference to P. W. Jordan's book: 'An Introduction to Usability', (2002). [1]

A product that is usable for one person may not be usable for another... Its crucial, then, to know who the product's users will be and their characteristics.

In the following the user characteristics are introduced:

1. **Experience:** Previous experience with the product will most likely influence how easy or difficult a task is to perform. Users should be able to generalize from previous experience to assist them execute new activities using well-designed products.

2. **Domain Knowledge:** Domain knowledge refers to a collection of information about a task which is independent of the product being used to execute the task. The designer can take this into account and try to construct a package that makes use of any domain expertise, while being careful not to assume that all users will have this information, as this could make the package difficult for those who don't.

3. **Cultural Background:** Users' cultural backgrounds can have an impact on how they interact with products. The 'population stereotypes' are responsible for this. This is especially the case when products have a safety-critical component, because people may return to 'instinctive' behaviors in an emergency.

4. **Disability:** Obviously, things that are useable by able-bodied individuals are not necessarily usable by disabled people.

5. **Age and Gender:** Young men, for example, are on the average physically stronger than women and the elderly, therefore gender can be a factor. Age can also be a factor because different generations have grown up with different sorts of technology.

We intend to elaborate on the user characteristic ‘experience’ before finalizing this section. There are two reasons of our intention.

First of all, the consistency of a product's user interface allows the user to generalize from one situation to the next. As we defined in section 1.1.3.4, ‘striving for consistency’ is the first of the eight rules of interface design provided by Schneiderman. It would in all probability be correct to claim that increasing the consistency of a user interface will assist designers in benefiting the experience characteristics of users. Consistency will be examined later on in section 1.2.5, namely Principles of Usable Design, as a principle of usable design.

The second reason is that the next section will be focused on more detailed aspects of this user characteristic. Aside from the issue of 'experience,' the user characteristics listed above are ones that, if they change at all, will most likely change over a lengthy period of time. Whereas the experience of a user with a product can probably change very fast.

The next section will address the issue of the change in task performance and usability due to the change in a user’s degree of experience with a product.

1.2.4 Components of Usability

As an activity is repeated, the usability of a product for a certain individual performing a specific task may vary rapidly. [1] This change in usability probably occurs due to the increase of user's experience as they keep repeating the same task while using the same product.

In the following, a five-component model of usability is introduced with reference to P. W. Jordan's book: 'An Introduction to Usability', (2002). [1] It should be noted that the model accounts for the improvement in task performance as a result of repetition.

1. **Guessability:** Guessability refers to the effectiveness, efficiency, and satisfaction with which specific users can execute specific tasks for the first time using a particular product. It is less important in scenarios where the user has received product training, or where there is little urgency to accomplish tasks correctly on the first try.

2. **Learnability:** Learnability is defined as the effectiveness, efficiency, and satisfaction with which certain users may acquire a competent level of performance on specific tasks using a product after having previously accomplished those tasks. It will be less significant in scenarios where training time and resources are relatively unrestricted.

3. **Experienced User Performance (EUP):** EUP refers to the effectiveness, efficiency, and satisfaction with which a group of experienced users can complete a set of tasks using a specific product. It will be the most significant component of usability in scenarios where there is less necessity to learn quickly but where it is critical that users perform well once they have learned how to use the product.

4. **System Potential:** System potential refers to the maximum level of effectiveness, efficiency, and satisfaction with which it would be possible to execute certain tasks using a product. Usually, a product's EUP falls short of its system potential.

5. **Re-Usability:** Re-usability refers to the effectiveness, efficiency, and happiness with which certain users can perform specific tasks with a specific product after a relatively long period of time away from these tasks. Re-usability is likely to be crucial When a product is used in intermittent 'bursts'.

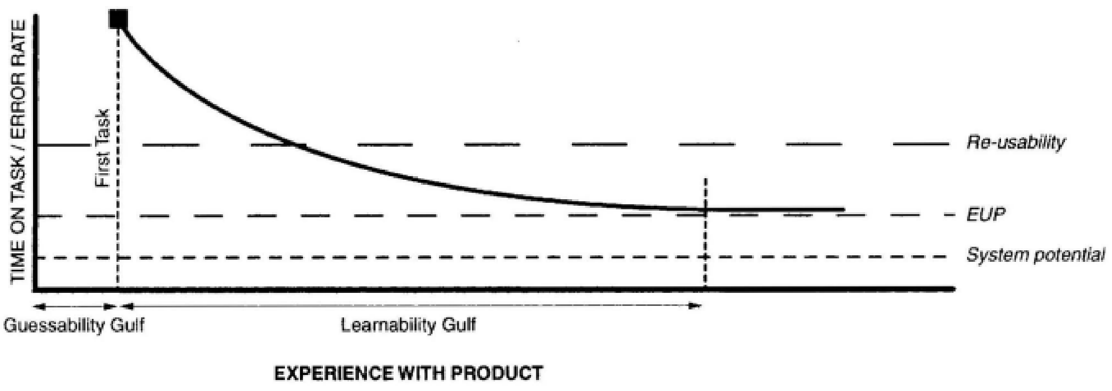


Figure 5: Idealized Learning Curve Illustrating the Five Components of Usability. [1]

Each component of usability is vaguely linked to a distinct segment of a notional learning curve. Figure 5 shows an example of such type of curve.

1.2.5 Principles of Usable Design

In section 1.1.3.4, namely Design, we have defined the meaning of principles and their range of domain. For the purpose of shortly bringing these to mind, principles are high-generality and low-authority design rules. This means that their range of domain is fairly high but designers should avoid heavily sticking to these rules.

In the following, ten principles of usable design are presented in reference to P. W. Jordan's book: 'An Introduction to Usability', (2002). [1]

1. **Consistency:** To achieve consistency, a product must be designed in such a way that similar tasks are performed in the same manner.

2. **Compatibility:** Achieving compatibility requires Designing a product so that its way of functioning is compatible with users' expectations based on their knowledge of other types of products and the 'outside world'. These 'other sources' of knowledge could be other sorts of products or anything else in the 'outside world' that influences how a user views utilizing a product.

3. **Consideration of User Resources:** Taking user resources into account requires designing a product so that its operation takes into account the demands imposed on

users' resources during interaction. When using a product, it's critical that none of the user's capacities are overloaded.

4. **Feedback:** To provide useful feedback, a product must be designed in such a way that the user's actions are acknowledged and a meaningful indication of the actions' outcomes is presented.

5. **Error Prevention and Recovery:** To prevent errors and provide easy error recovery, a product must be designed in such a way that the likelihood of user error is minimized and errors can be recovered fast and simply if they do occur. Many software products feature 'undo' functions, which are a good example of quick and easy error recovery.

6. **User Control:** To achieve high levels of usability, a product must be designed in such a way that the user's influence over the actions taken by the product as well as the state that the product is in is maximized. Customizability, presented in section 1.1.3.5 among the principles of the group ‘Flexibility’, can possibly be a principle which, when comprehended and applied correctly, will satisfy this issue.

7. **Visual Clarity:** To attain visual clarity, a product must be designed in such a way that the information displayed can be read quickly and easily without becoming confusing. Section 1.1.3.2, namely Information grouping, conveys explanations that probably help in satisfying to this issue.

8. **Explicitness:** Products should be designed in such a way that it is obvious how to use them. As a result, achieving explicitness entails designing a product with indicators as to its functionality and operation manner.

9. **Appropriate Transfer of Technology:** Appropriately transferring technology implies the suitable use of technology established in other settings to improve a product's usability.

10. **Prioritization of Functionality and Information:** Prioritizing functionality and information entails designing a product in such manner that the user can easily access the most critical features and information.

1.2.6 Measures of Usability

In this section we will describe how to measure the three aspects of usability which are used in ISO usability definition, which are provided in section 1.2.1. Afterwards we will introduce four different views on how usability should be measured. These views will differ from one another in terms of their focal point.

In the following, the three core aspects of usability; namely effectiveness, efficiency and satisfaction, are explained in reference to P. W. Jordan's book: 'An Introduction to Usability', (2002). [1] We will mainly approach these aspects by presenting several properties of user-product interaction which can be measured for the purpose of evaluating each aspect.

Effectiveness refers to the degree to which a goal or task can be accomplished.

Firstly, the quality of output can be measured in order to assess the effectiveness. In this context, the term 'output' probably does not have to be a concrete result, but fundamentally indicates the outcome of user's interaction with a product.

Secondly, the degree to which a task has been accomplished can be measured. With more complicated systems, it's likely that a user only succeeds partially in completing a task. Therefore the extent to a user's task accomplishment while interacting with a product can be measured in order to assess the effectiveness of this user's interaction with that product.

Efficiency refers to the amount of time and effort necessary to achieve a goal.

The attributes of user-product interaction which can be measured in order to investigate the efficiency of the interaction can be listed as follows [1]:

1. Deviations from the Critical Path: For most tasks, there is a critical path for task performance, which is the least effort approach of performing the task. It has a detrimental impact on efficiency when a user departs from the critical path. A diversion from the critical path could, for example, be defined as the need to consult a handbook or an online support system.

2. Time on Task: Obviously, the faster a user can finish a task with a product, the more efficient that product is for that task.

3. Mental Workload: The bigger the mental workload of users, the more likely it is that an error will occur. Mental workload is a productivity metric that has been commonly

used to evaluate the usability of products where task completion durations are fixed and error rates are low. This, along with error rate, is arguably the most commonly employed usability measure.

4. Error Rate: This is one of the most widely utilized efficiency measure. If a user can finish a task without making any mistakes, then the task may take less time than if mistakes are made and must be addressed.

Besides the efficiency measures which we have defined above, we will introduce a categorization of user errors based on the severity of the error.

These categories are; minor errors, major errors, fatal errors and catastrophic. It could probably be said that the category, which an error belongs to, depends on the recoverability of the error, whether the error prohibits the user from completing the task and whether it causes additional difficulties.

Satisfaction refers to the level of comfort a user feels when using a product, as well as how acceptable the product is to users as a means of attaining their goals. Satisfaction can be measured by performing two different types of user-attitude analysis, namely ‘Qualitative Attitude Analysis’ and ‘Quantitative Attitude Analysis’.

Qualitative attitude analysis can be done by using a questionnaire or interview. Any remarks made by the user when using a product can simply be recorded for the same purpose. The level of satisfaction users felt while using the product can then be determined by analyzing the comments.

Quantitative attitude analysis can be conducted by using 'standardized' questionnaire and interview-based tools. Users are asked to give numerical ratings, possibly regarding their level of satisfaction while interacting with the product.

In the following, four views on how usability should be measured are presented and explicated with reference to N. Bevan’s paper: ‘What is Usability?’, (1991). [6]

1. The product-oriented view: This point of view asserts that usability can be measured in terms of the product's ergonomic qualities. We intend to clarify possible confusions about the term ‘ergonomic qualities’. ISO defines ergonomics as the: effectiveness, efficiency, and satisfaction with which specific users in a given environment can achieve certain goals. This definition can in all likelihood recall the ISO definition of product usability.

2. The user-oriented view: According to this viewpoint, usability can be measured on the basis of the user's mental effort and attitude.

3. The user performance view: Usability, as for this viewpoint, can be measured by observing how the user interacts with the product. The observation focuses on either the product's ease-of-use or its acceptability. The ease-of-use of a product refers to how straightforward it is to use, whereas acceptability refers to whether or not the product will be used in the real world.

4. Contextually-oriented view: This viewpoint states that a product's usability is a function of the user or group of users being examined, the task they do, and the environment in which they operate. We could claim that this viewpoint, much like the product-oriented view, can be associated with the ISO definition of usability because it takes into account a specific user or group of users, a certain setting and a specified task.

1.2.7 User Experience

In this section we intend to study the term ‘user experience’. Our intention is to separate the terms ‘usability’ and ‘user experience’. In section Results and Discussion we will provide the outcomes of the user experience questionnaire we administered. Therefore we aim to emphasize on the concerns and objectives of each issue and thus expose the difference between them.

We will start by introducing a number of different ways to define user experience through different perspectives for the purpose of regarding as many details as possible.

User experience can be defined in a variety of ways: [8]

- A study of the usability component of satisfaction.
- Not to be confused with usability, which has a long history of emphasizing user performance.
- A broad term for all user perceptions and responses, whether subjectively or objectively measured.

“The definition of user experience in ISO FDIS 9241-210 is: A person's perceptions and responses that result from the use and/or anticipated use of a product, system or service… A person's “perceptions and responses” in the definition of user experience are similar to the concept of satisfaction in usability.” [8]

A rather informal way to define user experience might be the following;

Pleasure, aesthetics, fun, and flow have been used to describe the multidimensional nature of felt experiences on the one hand, and boredom, aggravation, and intrusiveness on the other. [3] The term ‘felt experience’ can be described as what it's like to interact with a device like an MP3 player or a pet robot.

There is a difference between usability methods, which aim to improve human performance, and user experience methods, which aim to improve user enjoyment while accomplishing both pragmatic and hedonic aims… A poll at Nokia demonstrated that user experience was perceived in a similar way to usability, however with the addition of anticipation and hedonic responses. [8]

The distinction between task performance and pleasure causes various concerns throughout development.: [8]

- **Usability Concerns**:

  - To evaluate effectiveness, efficiency, comfort and satisfaction, and to design with the intention of attaining them.

  - To design easy-to-use products and to evaluate for detecting and resolving usability issues.

  - When relevant, the temporal aspect raises concerns regarding learnability.

- **User Experience Concerns**:

  - To achieve the hedonic goals of stimulation, identification, and evocation, as well as the associated emotional responses, to the greatest extent possible.

  - Comprehend the user's experience with a product in based on what people do and why while interacting with that product.

1.3. Aesthetic Design in Automotive Industry

In this section we intend to provide an insight into the automotive design industry and the place of aesthetics in it. This awareness about our target user group can perhaps be of help when we will discuss the potential place of our system in the modern industry. Additionally we will give place to virtual prototyping, where we will define and explain this prototyping technique, compare virtual- and rapid prototyping, expose the benefits of virtual prototyping and provide a state-of-the-art example. Since our system also present an interactable virtual environment where design outcomes can be evaluated, we will aim to explain the benefits and shortcomings of virtual environments when they are used for assessing design prototypes.

1.3.1. The Automotive Industry

Automotive manufacturers must compete in an increasingly competitive market dominated by a few large multi-national corporations in today's global economy... To differentiate their products, individual brands are increasingly relying on brand image, quality, and style. [9]

When a user is presented with a large number of functionally equivalent products in a similar price range, they will frequently create an initial view about a product's overall quality based on visual assessment and comparison of its external 'look and feel.' [9] As we have emphasized in section 1.1.4, the cycle of prototyping a design, evaluating users' interaction or overall experience, and redesigning according to the results of this evaluation plays an arguably big role in design process. This cycle of iterating until a certain level of user satisfaction is achieved is probably one of the most important aspects in designing a usable and financially successful product. In the automotive industry, evaluation processes commonly utilize physical prototypes which are expensive, and take a long time to develop. [9] However, costs, technology, and productivity are all critical for the automotive industry's existence. [10]

1.3.2. Aesthetic in the Automotive Industry

There is no specific definition of aesthetic quality since it is a subjective product property that is experienced by a user through visual inspection and comparison… Such aesthetic quality aspects can have a significant impact on the final product's design and market distinction. [9] Those same aspects, along with brand impression and expression, are significantly influenced by product appearance and, as a result, are a top priority for today's manufacturers… Particularly interior design in automotive, is rapidly shifting from function and usability, to aesthetics. [11]

Relationships between these three aspects, namely function, ergonomics, and aesthetics, are a concern… There is an indistinct inclusion of aesthetic design into the existing functional and ergonomic design framework due to a lack of comprehension of the links between function, usability, and aesthetics. [12]

1.3.3. Virtual Prototyping

Our system will probably be used most commonly for the purpose of testing design prototypes. We offer designers an intractable and realistic environment where certain aspects of a users' experience with vehicles can be evaluated. Therefore we aim to understand the role of virtual prototyping in automotive design industry. As mentioned in section 1.3.1, employing physical prototypes is an expensive act. Therefore 'Virtual Prototypes' (VP) are becoming more popular as physical models are being pushed out. [9]

There have been numerous instances of considerable commercial benefits from the use of virtual prototypes in the design and production process. [9] If the current test prototype can't be reused, a new prototype must be built to re-evaluate the corrected model. This makes the 'Rapid Prototyping' (RP) process not only time-consuming, but also expensive, with a significant impact on a product's pricing… But by visualizing and determining difficulties at an early stage of the design process, the VP approaches minimize costs and shorten the life cycle… When compared to physical prototyping, this visualization is integrated into an interactive 3D user interface, allowing for faster design decisions and revisions…[10] Visualizing complicated quality issues enhanced communication and allowed for fresh insights and debates about acceptable quality targets between design and manufacturing teams, suppliers, and customers. [9] However, despite the significance of VP approaches, RP procedures have the upper hand due to the functional evaluation of the products. [10]

Furthermore, if engineers are to evaluate aesthetic quality using virtual prototypes rather than physical prototypes, they must be certain that what they see truly reflects how the product will look and function in the 'real world.' As a result, a photo realistic, interactive 3D image of the product that 'responds' to interaction and external forces is required for a virtual aesthetic quality evaluation tool...[9]

1.3.4. State of the Art

Engineers can use VITAL [9] to investigate "what if" scenarios by interactively visualizing the ultimate fit... Engineers were sure that a commercially viable system, comparable to the VITAL system, would substantially aid in the establishment and evaluation of quality targets…. Because these tasks are typically handled manually, the current problems connected with data exchange concerns and the time it takes to set up a scene (lighting, material, environment maps, etc.) combine to decrease efficiency… To cut down on time, the VITAL system would require a data preparation/migration tool. [9]

J. H. Oetjens (2014), on the other hand, provides an insight on the usage of virtual prototyping for safety measurements. [64] According to [64], virtual prototyping is being utilized for evaluating system safety and reliability, execute system-level testbences and testbench qualifications. Early safety analyses of automobile systems require the use of test benches and virtual experimentation environments. [64]

Early system verification and validation is a must if a system's reliability and robustness are to be ensured effectively. While employing Virtual Prototypes for early safety review is a very promising idea, there are still significant difficulties to overcome before it can become a reality. [64]

2. Motivation

Our research is mainly based on the topics ‘Human-Computer Interaction’ and ‘Usability’. Our goal in our studies and development is to offer automotive designers an interactive 3D environment where design prototypes can be evaluated. We aim to deliver this environment through a usable user interface. In order to achieve this, we studied the subjects ‘Human-Computer Interaction’, ‘Usability’, and ‘Automotive Aesthetic Design’. We believe our studies helped us achieving usability by understanding the fundamentals and practical implications of these subjects

We intend to contribute to the automotive aesthetic design industry by developing a system that presents designers a realistic environment while granting designers as much control as possible. Our system enables designers to control certain aspects of the interactable environment, and to execute the evaluation of a design process in a debatably rapid and learnable manner. While developing our system, we aimed to give place to state-of-the-art technologies. By doing so, we arguably draw attention to the possible future these technological features in the automotive aesthetic design industry.

Our ultimate purpose, besides contributing to the modern industry, is to highlight areas of improvement in our system in order to suggest a set of topics to regard for future research. We intend to compare our system with state-of-the-art technologies to get a better grasp of the place of our system in modern industry.

3. Materials and Methods

In this section, our implementation is explained in detail; the design ideas behind the menu, how the menu serves as an interface between the implemented features and designers, as well as the back-end implementation of the features.

We also reported the challenges that we have faced, how we have overcome them and also the improvement opportunities the final version of our simulation in terms of existing bugs and unimplemented features which could have made our simulation more wholesome and complete.

3.1. Menu Design

We intend to avoid presenting designers a fully conventional menu design. For the purpose of achieving a modern and interesting look in our menu we decided to enhance the appearance by implementing two different 'Material's [13] and using them as ornamentations.

One of the materials, the 'Common Background Image: Voronoi', hast its 'Material Domain' [14] set as 'User Interface' and is utilized directly by assigning an instance of the material on an Image [15]. We used this background effect in each of the menu pages in order to attain visual consistency.

The second material, the 'Wireframed Earth', is a masked wireframe shader that we use to render an arguably futuristic-looking 3D earth. We make use of this material by combining a sphere, which has the 'Wireframed Earth' material assigned to, with a second sphere, which looks like a simple and realistic 3D earth model. We use a display of the combination of these two spheres in our user interface. We gave place to this display in two menu pages, in which geographical configurations take place.

Besides the two materials that we use as visual strengtheners we also paid attention to the color palette and text fonts we use in the user interface. While assigning colors to our user interface elements we aimed to use conventional colors while trying to avoid creating a boring look. We tried to choose text fonts which were not confusing and futuristic enough so that they would not look out of place when observed together with the rest of the user interface elements.

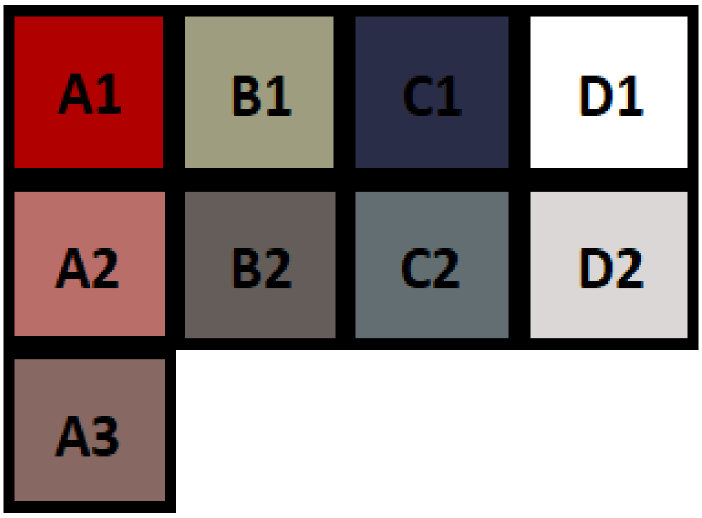


Figure 6: Color Palette

3.1.1. Color Palette

We tried to use apparent colors while designing the user interface.

For our 'Back' and 'Delete' buttons we used the color A1 in Figure 6 because, as we have stated in section 1.1.3.1, red is generally associated with 'stop' actions. In this context, user actions are categorized in 'stop'-, 'go'- and 'standby' actions.

We assigned the color B1 in Figure 6 to our 'Save & Proceed' buttons, which basically function as standard 'Next' buttons which are most generally seen in software- installation or setup interfaces (In the last menu page this button serves as a 'Finish' button which is also most generally seen in software- installation or setup interfaces). We decided to assign this color because we wanted the assigned color not to draw too much attention and simultaneously be, unlike the color A1, can perhaps be associated with a type of action similar to 'saving' or 'switching to the next menu page'.

We assigned two different tints of blue, C2 and C1 in Figure 6, for our drop-down menus and the background color of selected Metahuman entries, respectively. We decided to assign tints of blue to these specific interface elements because in each menu page designers are presented with one of these two elements in multiples. Therefore we aimed to select colors which would not create an overwhelming feeling when observed by designers in multiples.

We assigned less-vibrant tints of A1 and B1 in Figure 6; namely A2, A3 and B2 in Figure 6, to the rest of the buttons in order to achieve visual constancy.

Concurrently we succeeded making these buttons not stay in the background, which was our intention as well.

We assigned the colors D1 and D2 in Figure 6 as well as the color 'Full Black' (HEX #000000) to our 'Text Block' [16] elements. We aimed using the tints of white, namely D1 and D2 in Figure 6, as widely as possible and utilized the color full black where we thought using a tint of white would decrease the distinctness of the text block. We assigned the color D2 to the text blocks which we did not want to outshine the rest of the interface elements. We assigned the color D1 to the text blocks which we wanted to emphasize, such as the headings of each menu page.

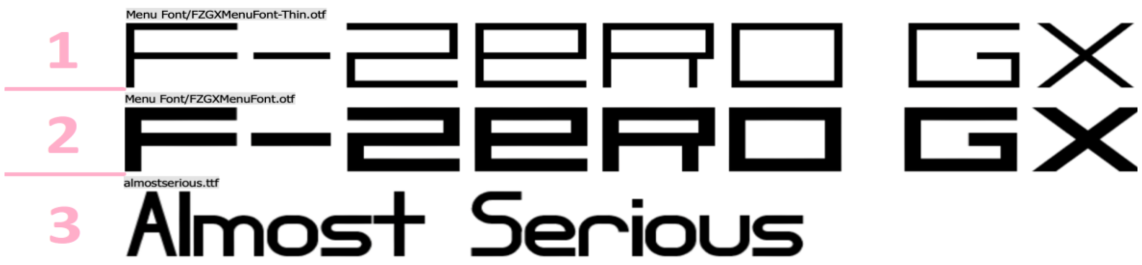


Figure 7: Text Fonts

3.1.2. Text Fonts

We used three text fonts which we obtained from internet. [17]

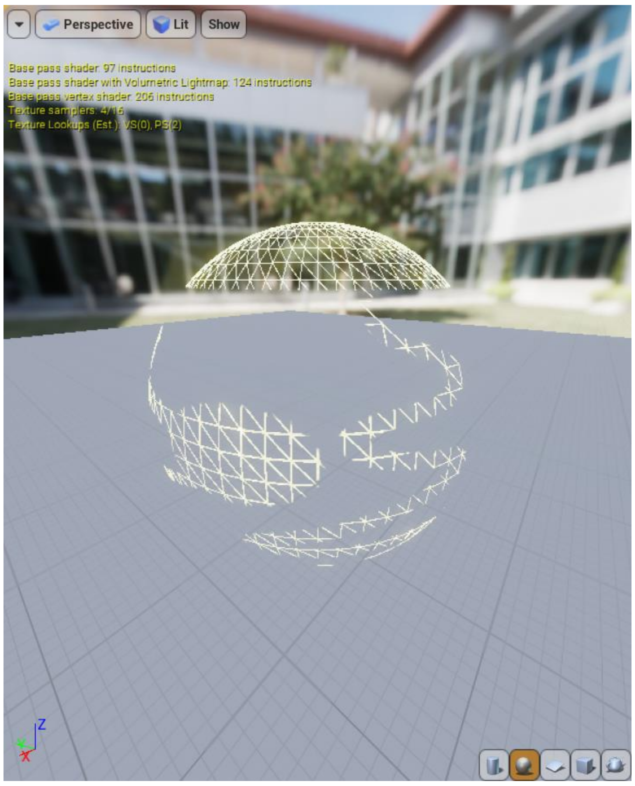
We used the 'Thin' and 'Bold' variants of the text font 'F-Zero GX Menu' (marked with number 1 and 2 in Figure 7, respectively) because we found it to be futuristic enough to fit the rest of our interface design and simple enough so that it would not confuse designers. We used the text font 'Almost Serious' (marked with number 3 in Figure 7) where we found the usage of the text font 'F-Zero GX Menu' not fitting in terms of clearness.

By using unconventional text fonts we aimed to achieve a style that is simple and not boring at the same time.

3.1.3. Page Indicators

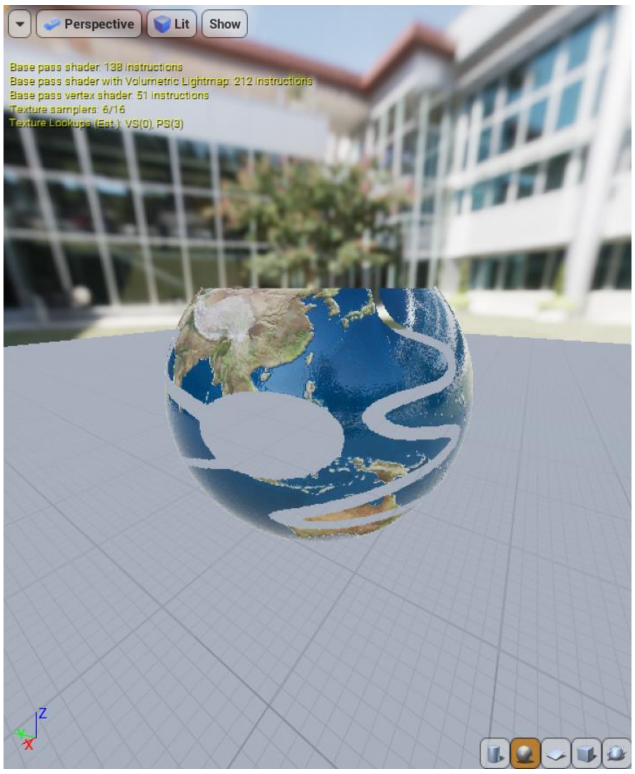
For the purpose of indicating which menu page designers are currently working on we decided to give place to rectangular-shaped tabs which we placed on the bottom of each menu page. The amount of these tabs correspond to the total amount of menu pages. We indicate the index of the currently-edited menu page by making the tab, which has the same index among the tabs when counted from left to right, blink.

The previous tabs (tabs that are on the left of the blinking tab) are tinted with the color white and the remaining tabs (tabs that are on the right of the blinking tab) are tinted with an almost-fully transparent color white. This helped us in stylizing the user interface in a more intuitive way rather than a more numerical, therefore traditional, approach.



```
Base pass shader: 97 instructions
Base pass shader with Volumetric Lightmap: 124 instructions
Base pass vertex shader: 206 instructions
Texture samplers: 4/16
Texture Lookups (Est.): VS(0), PS(2)
```

Figure 9: Material, Wireframed Earth



```
Base pass shader: 138 instructions
Base pass shader with Volumetric Lightmap: 212 instructions
Base pass vertex shader: 51 instructions
Texture samplers: 6/16
Texture Lookups (Est.): VS(0), PS(3)
```

Figure 8: Material, Realistic Earth

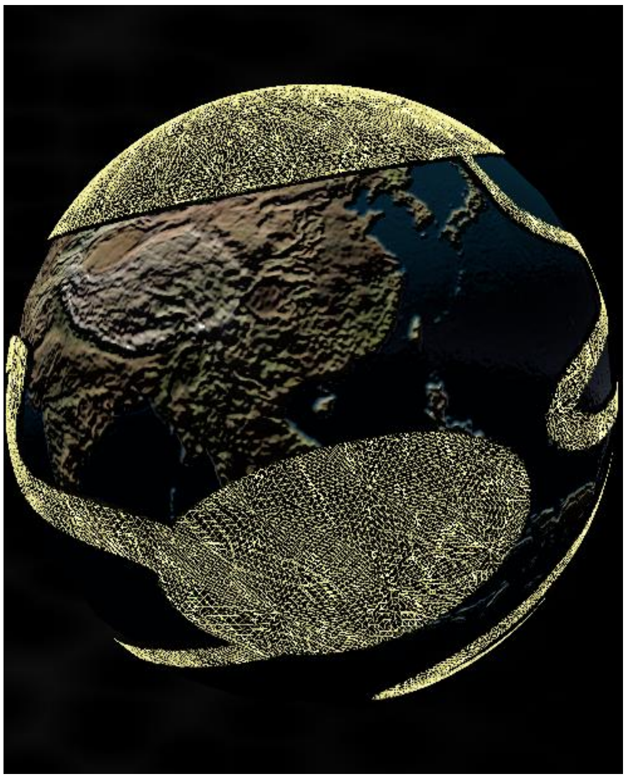


Figure 10: Interface Element, Futuristic Earth Image

3.1.4. Futuristic Earth Image

We implemented two different materials, created a Material Instance [18] of both materials, assigned each of the material instances to a different sphere (two spheres in total) and ultimately utilized the combination of these two spheres in order to achieve the look on Figure 10.

The view of the combination of these two spheres is rendered on the user interface by applying the implementation explained in the 'Displaying a View from the Level in the User Interface' section.

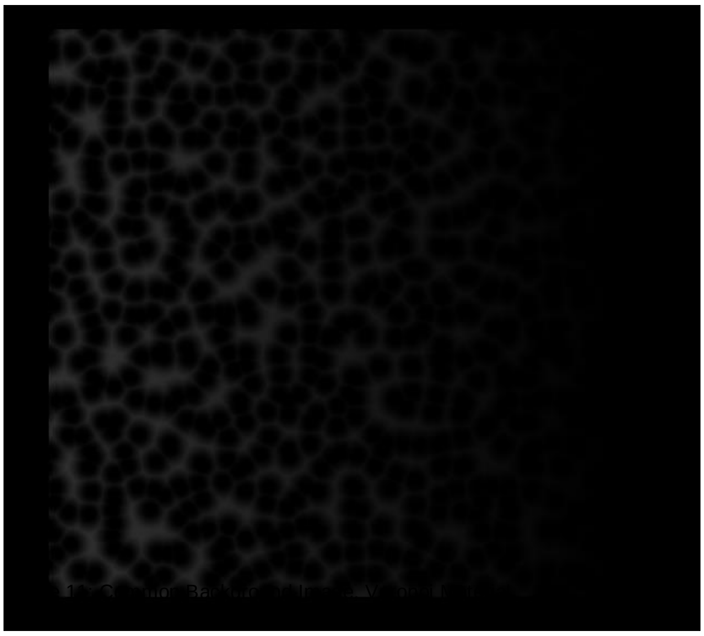
Both materials have their 'Blend Mode' [19] set to 'Masked'. The purpose of this approach is to render the sphere, which has the 'Wireframed Earth' material assigned, exactly on the areas where the other sphere, which has the 'Realistic Earth' material assigned, is transparent.

As can be seen on Figure 8, the 'Wireframed Earth' material is an emissive wireframe material. We use a noise texture to add an extra shimmer on top of the existing emission. The 'Panner Material Expression Node' [20] is utilized in order to animate the UV coordinates of the noise texture for the purpose of moving the extra-shimmering areas on the wireframed surface. We also used 'Tessellation' [21], with the 'PN Triangles' option chosen, in order to increase the impact of the 'Wireframe Earth' material by increasing the amount of rendered triangles without actually having to use a high-poly sphere model. Lastly we made use of the ‘Material Node: World Position Offset’ [22] in order to increase the liveliness of the material. We combined multiple sin- and cos functions of different frequencies and offsets in order to randomize the movement that is induced by the ‘World Position Offset’.

As can be seen on Figure 9, the ‘Realistic Earth’ material is a simple material which uses diffuse and normal texture maps. We succeeded to give a shiny look to only the areas where there is water by making use of the blue color in the diffuse texture map by executing a few simple mathematical operations. The mask texture which is used in ‘Wireframe Earth’ material (shown in the Appendix section) is also used in the ‘Realistic Earth’ material, we only inverted its usage so that the transparent areas on the ‘Realistic Earth’ material exactly correspond to the opaque areas on the ‘Wireframe Earth’ material.

The mask texture used for the Futuristic Earth image (shown in the Appendix section) is hand drawn. (The blue outline of the mask texture in the appendix is placed for clarifying the boundaries of the texture, the original mask texture does not have the blue outline.) We drew a red outline around the black areas on this mask texture in order to implement an emissive border between the wireframed and the realistic areas on the futuristic earth’s surface for the purpose of fusing the two materials and thus creating a complete look.

We eventually decided to not give an emissive color to the border area because we thought it would attract more attention than it should. Therefore we just painted this border area with the color black, which arguably succeeds to serve the same purpose.


e 11: Common Background Image_ Voronoi Material

3.1.5. Common Background Image

The menu has the same background image in each of the menu pages. The common background image (Figure 11) approximately covers 80% of the background, from left to right. The remaining ~20% of the background, the right portion of the background, is left empty in order to fill the area with a different visual queue which is related to the menu page's functionality; this portion is filled with the 'Futuristic Earth Image' (Figure 10) in the 'Time- and Weather Configuration' page, in the 'Metahumans' page the remaining portion is filled with a single image of a robotic human face and in the 'Vehicle- and Camera Configuration' page we filled this portion of the background with a photograph of a highway.

We implemented a 'Voronoi' [23] material for the purpose of creating a common menu background effect. We implemented the material by practicing the more efficient approach rather than the simpler one: we divided the surface into grid cells and instead of looping through every moving point on the surface for each pixel we only loop through the surrounding grid cells of a pixel, including its own cell, which basically means that for each pixel we loop through 9 grid cells instead of looping through each moving point (in our scenario the minimum number of moving points would approximately be 20 in order to be able to fill the screen with a 'voronoi tessellation' that is dense enough).

Finally, we decrease the alpha value (opacity) of this material depending on the horizontal UV coordinates so that the common background image fades out when it gets closer to the right ~20% portion of the background. This way the common background image arguably blends with the remaining ~20% portion of the background.

3.1.6. Areas of Improvement

- The changes that happen in the user interface are indicated in a fully static way. Thus, we could say that the feeling of the user interface is conventional rather than modern, which we intended to avoid.

A simple example to this would be the page indicators: when designers switch the menu page the transparency of the page indicator tab which indicates the previous menu page is set directly to either fully opaque or almost-fully transparent. Similarly, the transparency of the page indicator tab which indicates the current menu page is set directly to blinking mode.

If the page indicator tabs, as an example, were connected with a line which would be filled or emptied while transitioning to the new blinking page indicator tab, then the reaction of the page indicators to the change of the menu page would be dynamic and lively. Such an approach would probably give designers a less-boring feeling.

Speaking in general, the solution to the static design of the user interface would be to implement the user interface in a more organic and lively way by animating the changes that happen during designers' configuration.

It would be a fitting solution, as a second example, to linearly interpolate the camera location (the camera which renders the selected vehicle model) when designers change the camera mode selection in the Camera- and Vehicle Configuration menu page instead of snapping the camera from one location to another.

Implementing the design of the user interface by following such an approach would probably induce a more positive and progressive feeling as designers configure the simulation settings.

- We initially intended to implement a material which would consist of moving, interconnected lines. We decided to implement a material which consists of voronoi tiles in order to make the voronoi edges equidistant and thus accomplish our intention. In order to implement this we attempted to follow the instructions from Inigo Quilez's guide on 'Equidistant Voronoi Edges' [24] but we could not succeed. Therefore we decided to abandon the idea of presenting designers a background filled with lines of equal thickness and proceeded with the traditional voronoi look.

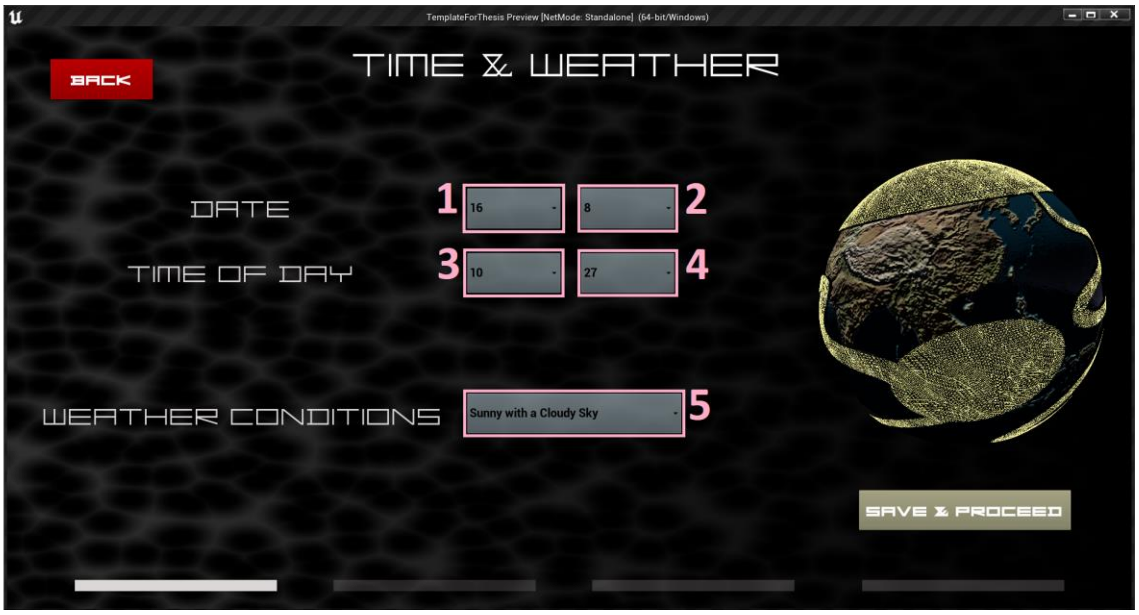
3.2. Menu Sections

In this section we explain in detail in what segments our user interface can be decomposed into. We describe how each of the menu pages serve as an interface between the designers and the back-end functionalities, as well as the back-end implementation itself. We also give place to detailed descriptions of each interface element in every menu page in terms of the purpose they serve and how they were implemented in a technical perspective. Finally we reported the challenges that we have faced and the improvement opportunities which we could not carry out and would make our simulation more complete.

3.2.1. Time- and Weather Configuration

We intend to offer designers a realistic simulation environment. For this purpose we implemented a variety of weather effects and a geographically accurate sun placement.

We present designers a simple user interface to configure these two environmental features. To configure the placement of the sun designers only need to choose the date and the daytime. The weather condition is selected through a variety of pre-defined options.


TemplateForThesis Preview [NetMode: Standalone] (64-bit/Windows)
BACK
TIME & WEATHER
1
2
3
4
5
SAVE & PROCEED

DATE 16 8
TIME OF DAY 10 27
WEATHER CONDITIONS Sunny with a Cloudy Sky

Figure 12: Menu Page, Time- and Weather Configuration

3.2.1.1. User Interface

Definition of the markers on Figure 12:

1. Drop-down menu to choose the day

2. Drop-down menu to choose the month

3. Drop-down menu to choose hours of daytime

4. Drop-down menu to choose minutes of daytime

5. Drop-down menu to choose the weather conditions

3.2.1.2. Geographically-Accurate Sun Placement

We utilized the ‘Sun Position Calculator’ Plugin in UE4 [25] in order to provide a geographically accurate sun placement. The user interface expects the date in (Day, Month) format because our simulation automatically sets the year according to a public variable in the GameInstance Blueprint in order to increase designers’ efficiency by reducing the number of data which our simulation asks designers to register. Similarly, the user interface expects the daytime in (Hours, Minutes) format because the difference that seconds make on the position of the sun is neglectable. Our simulation automatically sets the seconds to ’00’.

3.2.1.3. Weather Conditions

We made it possible for designers to edit the environmental conditions in the time- and weather configuration menu page. Our implementation provides a discrete variety of options as different weather types, which we created by utilizing UE4’s Particle System [26] and UE4’s Exponential Height Fog [27], which are supported by a realistic cloud system. Designers choose the weather type via a drop-down menu (marked with number 5 in Figure 12).

We used UE4’s Volumetric Clouds [38] in order to simulate realistic clouds, on which we had total control while designing our level. We adjust the profile of the clouds depending on the choice of the weather type.

Weather Types:

1. *(Clear and Cloudy Sky) Sunny*: To implement these two features we only adjusted the cloud profile and sun light intensity while benefiting the accurate sun placement.

2. *(Light and Heavy) Foggy*: We only used UE4’s ‘Exponential Height Fog’ to simulate a realistic fog. The light and heavy variants of this effect differ at the attributes of the ‘Exponential Height Fog’ instances that we use.

3. *(Light and Heavy) Snow*: To catch a realistic feeling, we created a particle system with 4 emitters, which spawn particles with two different sprites. The light and heavy variants of this effect differ at the particle count and movement. Additionally, we implemented an environment material to render ‘snow bulk’s on objects and utilized Quixel Megascan’s material library in order to obtain a realistic interaction of the environment with our weather system.

4. *(Light and Heavy) Rain*: We created a particle system for our rain effect. The light and heavy variants of this effect differ at the particle count and movement. We used a translucent and refractive material for our water particles. Moreover we implemented a shader for the car windows to simulate the water droplets on glass surfaces on a rainy day. Besides, the rain particles emit new water particles on the surface for simulating water splashes as it collides with an object while falling.

For both of snow and rain effects, our goal was to optimize the implementation by lowering the particle count and quality while keeping the feeling realistic. We also use the Exponential Height Fog instances for both effects in order to boost the realistic atmosphere of the weather conditions. Additionally, for each of the ‘heavy variants’ we lowered the sunlight intensity to achieve a colder ambience.

3.2.1.4. Areas of Improvement

1. In the final version of our simulation, snow- and rain particles can fall through the vehicle roof. The solution to this inconvenience would be to kill particles upon collision.

2. We could implement an environment material for rainy weathers to simulate the splash of rain droplets on water surface.

3. We could implement an interactive snow feature, where the snow bulks on horizontal surfaces can get thicker by time and can get affected by collisions.

4. The direction of both snow and rain particles could be adjusted according to the movement of the vehicle. In the final version of our simulation the direction of the particles is globally uniform.

5. The direction of the water droplets on the vehicle windows could be adjusted according to the vehicle movement.

6. Location-based Weather Profiles: We initially intended to generate an accurate weather profile depending on the input coordinates. The first difficulty that we faced here was the fact that weather, just like time-zone, cannot be derived from the real-world coordinates because of factors like humidity, height and geographical past. As we searched for alternative approaches, we came across NASA’s annual ‘Global Temperature Map’ [29], which represents the average temperature of earth’ surface with a color code. It was our intention to extract the average temperature data for a given date by using these colored earth maps. To achieve this, we tried to read the color data of the pixel in the temperature map which geographically corresponds to the input location. We could not

achieve this goal, so we decided to give the total control over the weather conditions to designers.

3.2.2. Metahuman

In order to develop a simulation that presents designers as many realistic options as possible, we decided to use humanoid avatars. This way, we can enable designers to construct a car scenario which can be populated with a variety of photorealistic humans. We benefited Epic Games' arguably newly release 'Metahuman' because it provides a high quality system which is well optimized and can easily be integrated into UE4. We animated the Metahumans by using free-to-use animations for enabling the Metahuman characters to execute tasks and interact with their environment.

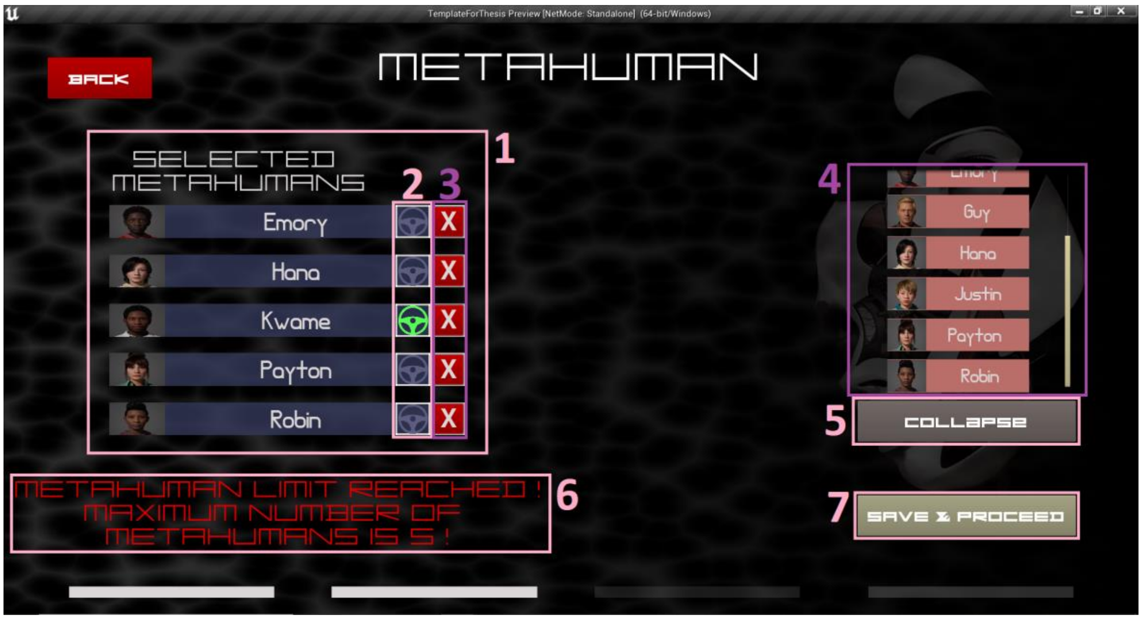

TemplateForThesis Preview [NetMode: Standalone] (64-bit/Windows)
BACK
METAHUMAN
SELECTED METAHUMANS
1
2
3
Emory
Hana
Kwame
Payton
Robin
X
4
Guy
Hana
Justin
Payton
Robin
5
COLLAPSE
METAHUMAN LIMIT REACHED ! MAXIMUM NUMBER OF METAHUMANS IS 5 !
6
7
SAVE & PROCEED

Figure 13: Menu Page, Metahumans. Save & Proceed Button Active.

3.2.2.1. User Interface

Definition of the markers on Figure 13:

1. List of selected Metahumans

2. Buttons to determine the driver among the selected metahumans

3. Buttons to remove a selected metahuman from the selected metahumans list

4. ScrollBox widget [30] where designers can select a Metahuman to add to the selected metahumans list

5. Button to toggle the visibility of the ScrollBox (marked with number 4)

6. Warning text which is shown when designers try to add a sixth Metahuman to the selected metahumans list

7. Save & Proceed Button to save the configured settings and continue to the next menu page

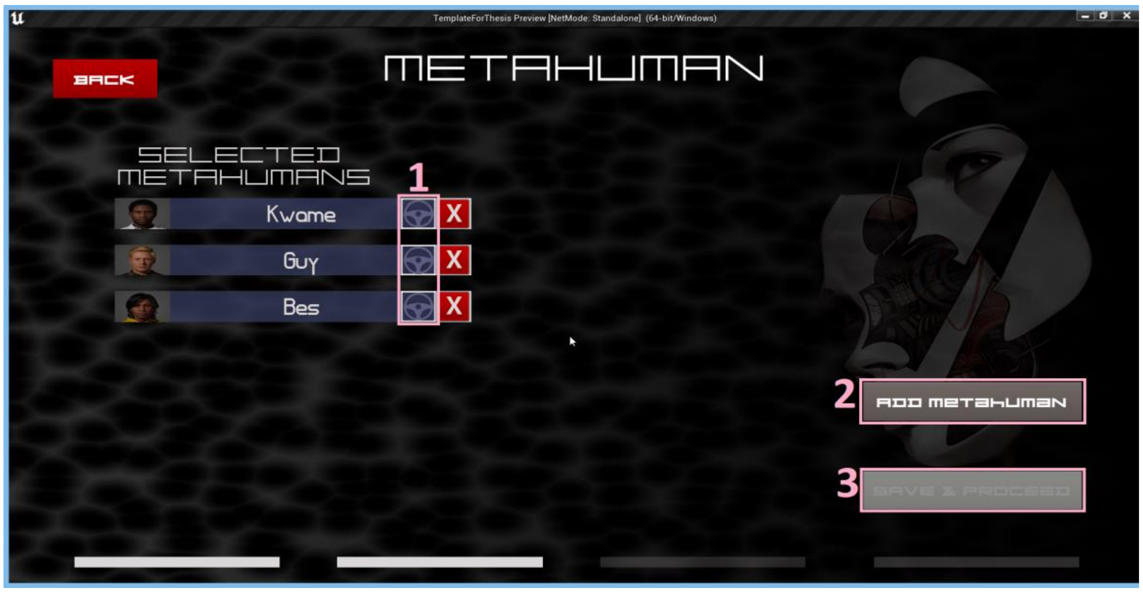


Figure 14: Menu Page, Metahumans. Save & Proceed Button Inactive

Definition of the markers on Figure 14:

1. Buttons to determine the driver among the selected metahumans

2. Button to toggle the visibility of the ScrollBox (marked with number 4 in Figure 13)

3. Save & Proceed Button to save the configured settings and continue to the next menu page (inactive)

Our user interface allows designers to choose the humanoid avatars which they want to see in the simulation and the driver in the simulation. The same Metahuman can be added multiple times. The maximum number of selected Metahumans is five. Therefore if designers attempt to choose a sixth Metahuman, then our simulation displays a warning text (marked with number 6 in Figure 13) for a certain amount of time and subsequently hides it.

The Save & Proceed button has an inactive state (marked with number 3 in Figure 14) and an active state (marked with number 7 in Figure 13). The Save & Proceed button is inactive when there are no drivers selected. It automatically gets activated when a driver gets selected. Our simulation allows designers to select only one driver at the same time.

If a selected metahuman who has been selected as the driver gets deleted, then the metahuman which comes as next in the selected metahumans list is automatically set to be the driver. If the metahuman who has been selected as the driver and is the last metahuman in the selected metahumans list is deleted, then our system does not set one of the other metahumans as the driver and deactivates the Save & Proceed button.

Example:

If, in the scenario which is displayed in Figure 13, Kwame gets deleted, then Payton is automatically set to be the driver and the Save & Proceed button is not deactivated.

If, in Figure 13, Robin had been selected as the driver and she gets to be deleted, then no other Metahuman on the selected Metahumans list is set to be the driver and the Save & Proceed button gets deactivated. The Save & Proceed button stays deactivated until a new driver is chosen.

3.2.2.2. Working with Metahumans

Metahuman [31] is a new system of Epic Games. It enables developers and creators to create and utilize high-detail humanoid meshes by following a simple work-flow. The meshes are automatically textured and rigged. They are also automatically equipped with an LOD system that provides 9 levels of detail. Our workflow on utilizing Metahuman for our simulation consists of 4 consecutive sections; creating a Metahuman, importing it into the project, animating it and then finally building a ‘No Player Character (NPC)’ [32] out of it.

3.2.2.3. Metahuman Creator

“Metahuman Creator is a free cloud-based app that empowers anyone to create photo-realistic digital humans, complete with hair and clothing, in minutes.” [33] Developers and creators must firstly use this web-application in order to create their own Metahumans. To create a Metahuman, the developers must choose a Preset Metahuman and then optionally edit that Preset. Metahuman Creator currently provides over 50 different Presets which are ready to be used and edited.

Besides using a Preset directly or slightly editing it, we also benefited the ‘Blending’ feature [34] of Metahuman Creator. The utility of the ‘Blending’ feature is to “…blend facial attributes from a selection of other MetaHuman Presets onto your own MetaHuman.” [34]

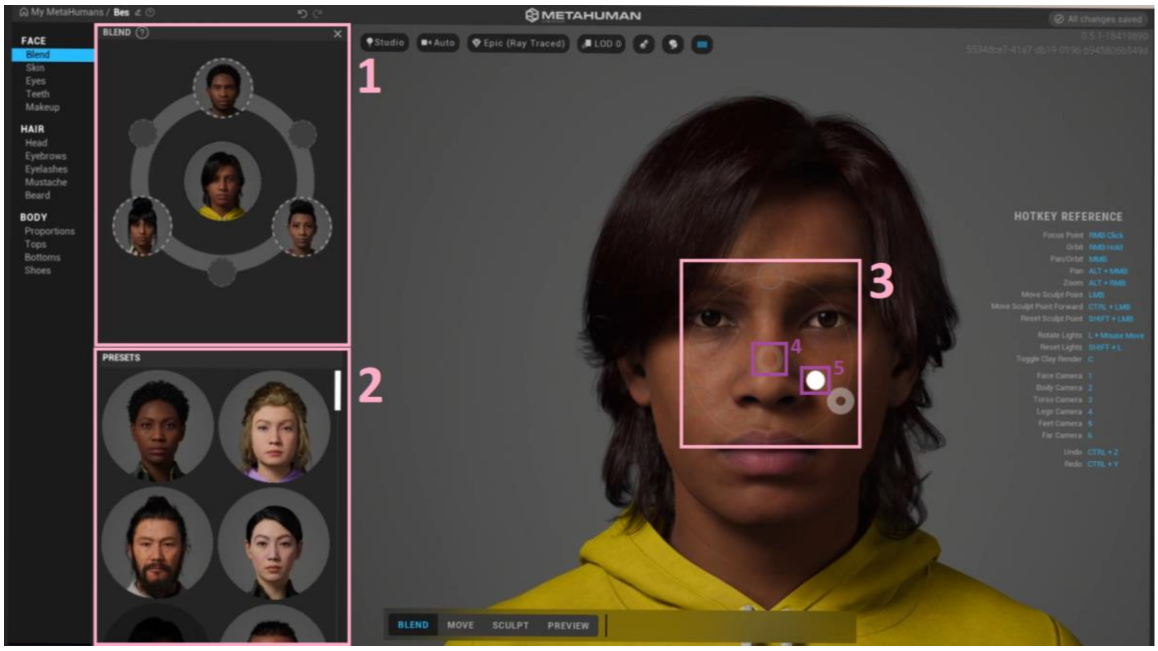


Figure 15: Metahuman Creator, Blending Feature

Definition of the markers on Figure 15:

1. Blend Panel
2. Presets
3. Blend Circle
4. Control Marker
5. Control Point

The control marker on Figure 15 is one of the many control markers from which the user can choose to edit but the rest of the control markers on Figure 15 are blended out because in this image a control marker (Number 4) is already chosen.

“Selecting control markers on the face displays a blend circle that matches the one in the Blend panel with the same number of slots filled. Drag the control point towards a Preset you want to blend with the MetaHuman in the viewport. Adding or removing Presets from the circle will not change the control points influence in the circle, that stays constant. However, changing placement, or selection, of Presets on the circle will change the blend influence on the MetaHuman in the viewport.” [34]

‘Blending’ feature helped us in creating the children of our grown-up Metahumans. We chose the grown up Metahumans as Preset and adjusted the face of the children so that they look similar to their parents. We can say that the ‘Blending’ feature made this process highly intuitive and realistic.

3.2.2.4. Challenges with Metahuman Creator

It is also possible to say that we have engaged some inconveniencies while utilizing Metahuman Creator for our project. First of all, the system is currently in Early Access so its functionalities are not finalized. One consequence of this is that not each of the Metahuman Presets is fully accessible: some of them are only usable with LOD 0 and LOD 1, which might turn into an optimization problem. The same problem also eventuates with some of the hair templates, which decreases the level of freedom that Metahuman Creator provides. We have overcome this inconvenience by only using the Metahuman Presets and hair templates, of which the LOD system is not restricted to LOD 0 and LOD 1.

The second difficulty that we have had to overcome was modelling children with Metahuman Creator. Currently there are only a few Presets which look like an adolescent and the editor does not provide any options to modify the skin of the metahuman, therefore developers are restricted to choose among the few fitting Presets and reach their goal by editing the chosen Preset. We have overcome this difficulty by benefiting the 'Blending' feature, as explained above.

3.2.2.5. Importing Metahumans into an UE4 Project

It could be said that the workflow of importing a Metahuman, which is created by using Metahuman Creator, into an Unreal Engine 4 project is very quick and intuitive. "Through Bridge, you can access your bespoke MetaHumans created in MetaHuman Creator, as well as over 50 premade MetaHumans. With both UAssets and source assets available, you can use Bridge to download and export your digital humans for direct use in Unreal Engine or further editing in Maya." [23] "Quixel Bridge is a desktop application that serves as a supporting tool for browsing, searching, downloading, importing and exporting Megascans assets." [24] Quixel Bridge is also debatably generous at providing options for downloading and exporting; it allows developers to choose the resolution of the asset ranging from 1K to 8K textures, developers can choose the variety of the textures that will be downloaded and directly export their Metahuman into UE4. The exported Metahuman assets are automatically assembled into an Actor Blueprint and are immediately usable. We followed the naming convention 'BP_MetahumanName' for each of our Metahumans.

3.2.2.6. Animating Metahumans

Animating Metahumans is a debatably easy but sensitive task. The Metahumans are imported automatically with their Physics Asset, Mesh and Skeleton. The name of the skeleton asset of Metahumans is 'metahuman_base_skel'. If the target skeleton of an animation asset is different than metahuman_base_skel, then developers must execute certain adjustments in order to use that animation asset on Metahumans. The required adjustments can be listed as the following:

1. The IK (Inverse Kinematics) targets under the Retarget Manager of the metahuman_base_skel must be set to 'None'.

2. The rig of the animation's target skeleton under the Retarget Manager must be set so that it matches to the metahuman_base_skel perfectly. (This is only the case if the animation's target skeleton is not the UE4_Mannequin, the standard humanoid character of UE4, because Metahuman_base_skel's rig matches perfectly to UE4's Mannequin.)

3. The pose of metahuman_base_skel must be adjusted so that it exactly matches the pose of the animation's target skeleton.

4. In the skeleton tree of metahuman_base_skel, the "Recursively Set Translation Retargeting Skeleton" must be chosen.

5. The animation asset must be retargeted to metahuman_base_skel in order to use it on a Metahuman.

We benefited free-to-use animations from Epic Games' Marketplace [25] and Adobe's Mixamo [38]. The animation pack 'MCO Mocap Basics' uses UE4's Mannequin as target skeleton, while Mixamo animations are (by default) targeted to a different skeleton. Each Metahuman in an UE4 project is targeted to the same skeletal mesh instance.

As mentioned in the 3rd list element above, where the required adjustments for animating a Metahuman are listed, the metahuman_base_skel's pose must be modified if it doesn't match the pose of the animation's target skeleton. Therefore, we faced a conflict where we needed the metahuman_base_skel in two different poses. We bypassed this problem by using a second UE4 project where we executed the adjustments for the metahuman_base_skel. We then migrated the retargeted animation assets to the original project. This approach helped us keeping a tidy project hierarchy while maintaining our previous implementations.

3.2.2.7. Inverse Kinematics (IK)

On top of standard animation retargeting, we also utilized inverse kinematics in order to achieve a realistic driving animation. We integrated the benefits of inverse kinematics by implementing the entire animation behavior of the metahumans in a single animation blueprint (Let the driver’s instance of this animation blueprint be called animInstance for the sake of convenience). Once initialized and the metahumans are spawned, our simulation feeds world-location data into the animInstance in each frame. This way, the driver’s hands are always attached to the steering wheel and therefore we achieve the visual accuracy of having the wheels, the steering wheel and the driver’s hands working in sync.

For inverse kinematics we used the ‘Two Bone IK’ function [39] from UE4’s animation library. “The Two Bone IK control applies an Inverse Kinematic (IK) solver to a 3-joint chain, such as the limbs of a character.” [39] depending on two input parameters: ‘Effector Location’ and ‘Joint Target Location’. “The effector location is where the IK is trying to place the bone, while the Joint target location indicate how the bones will rotate.” [40].

We call Two Bone IK in each frame for both left and right hand bones of metahuman skeletal mesh. In the animation blueprint we set the ‘Effector Location Space’ to World Location and the ‘Joint Target Location Space’ to Component Space. We provide the world location data for the Effector Location by using reference points (Spheres as Actor Components) on the steering wheel and the Joint Target Location is hard-coded in the animation blueprint. The absolute world location data of these reference points on the steering wheel are later on fed into the animInstance by using the game instance blueprint.

Flow of the World Location data in each frame:

1. Absolute world location of the reference points on the steering wheel, indicating the effector location for Two Bone IK, are saved in the game instance blueprint.

2. The game instance blueprint saves the location data in BP_DriverMetahuman.

3. The animInstance reads the location data from its owning actor.

3.2.2.8. Metahuman Characters

Besides the sitting and driving animations, we also implemented a walking feature for the metahumans. Our goal was to spawn Metahumans on the sidewalk and to make them walk to the car doors before they get placed on the car seats.

In order to achieve this goal, we changed the Parent Class of the BP_Metahuman-Names, in which the imported Metahumans are assembled, to ‘Character’ Blueprints, as this provides the ‘Character Movement Component’ [41] and therefore the freedom to utilize NavMesh [42] for way-finding. This enabled us to implement the ‘Walking’ feature for our Metahumans fast and precisely.

This feature is ultimately excluded from the final version due to timely constraints and changing priorities. Implementation of this feature in our simulation would bring further complex tasks with it; such as the configuration of Metahumans’ behavior, creation of a variety of scenarios in which the event flow of metahumans are unique, the integration of this configuration into the user interface and etc.

3.2.3. Vehicle- and Camera Configuration

Our simulation contains a camera system that runs similar to the traditional approaches, which are standardized in game development, and a vehicle driving system where designers can thoroughly observe and analyze a vehicle in a customizable environment. We decided to let designers configure the settings of both systems in the same menu page. In this menu page designers can choose the camera mode and the vehicle model which will be used in the simulation, as well as choose to create a custom camera mode. By presenting the option to configure both settings in the same menu page we were able to integrate a display of the chosen vehicle which is rendered through the perspective of the chosen camera mode.

This approach helped us in increasing the efficiency and accuracy of designers’ workflow because this way designers can preview the result of their configuration and therefore decrease the risk that their configuration, later on in the simulation, turns out to be an unideal choice. Our simulation allows designers to change the camera mode during the simulation by using pre-defined hotkeys (see Hotkeys for In-Game Controls) section but by seeing the preview beforehand, designers do not have to search for the best-fitting camera mode during the simulation. Furthermore, our simulation does not provide a vehicle selection feature during the simulation. Therefore choosing an unideal vehicle for the simulation is penalized by having to restart the simulation.

Definition of the markers on Figure 66:

1. Drop-down menu to choose a vehicle
2. Drop-down menu to choose a camera mode
3. Preview of the selected camera mode and vehicle
4. Button to create a custom camera mode

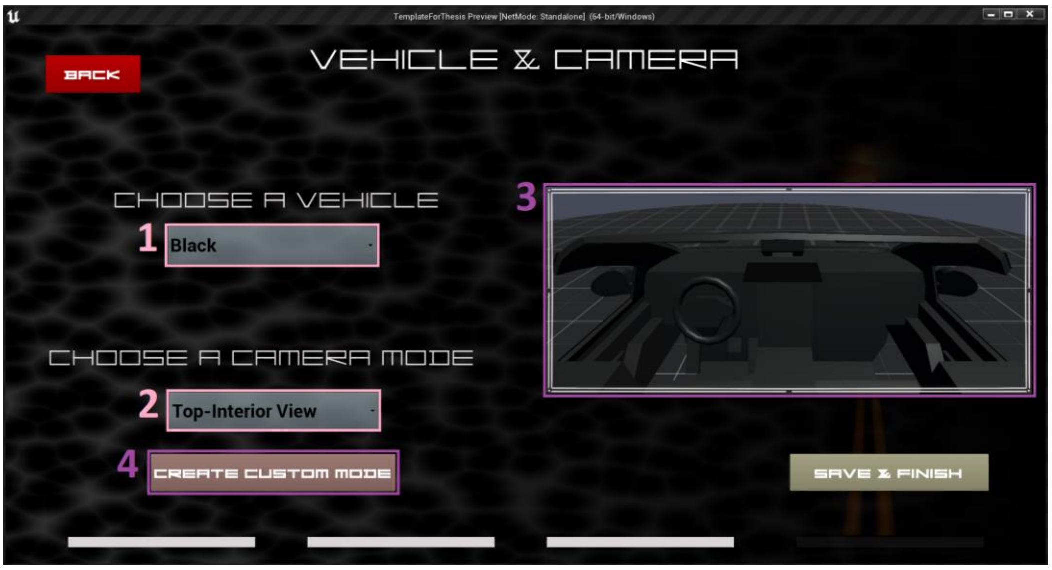


Figure 16: Vehicle- and Camera Configuration, Menu Page

3.2.3.1. Vehicle System

Our simulation mainly consists of a driving experience. As a result, we could say that it was only intuitive that we implemented a vehicle-driving system. By implementing a driving system, we achieved a higher level of immersion by putting the designers in the driver's shoes: while driving, depending on the camera mode, the awareness of designers can be more attracted to the interior/exterior of the vehicle, the environment can be perceived through the desired perspective and designers can play an active role instead of merely witnessing the driving experience.

3.2.3.1.1. User Interface

In order to present an option to choose a vehicle model, we decided to read the skeletal meshes which follow a certain naming convention and are located in a certain project folder, in the beginning of runtime. These skeletal meshes are then presented to designers in the vehicle- and camera configuration menu page. In the end designers can choose a vehicle among the presented options via a drop-down menu (marked with number 1 in Figure 16). This way, in order to add new vehicle assets to the simulation, designers do not have to register vehicle assets in a data structure and they must only follow the naming convention and place the assets in the correct project folder.

As we were searching for a technical approach to implement this system we, fortunately, came across IsaraTech's explanation about UE4's Asset Registry system [43]. Since Epic Games' documentation on Asset Registry system is arguably lacking a considerable amount of both theoretical and practical information about its usage in Blueprinting, such an alternative source of information as IsaraTech was crucial for our simulation.

We utilized the Level, which we use for our initial menu, for the displaying the preview of selected camera mode and vehicle mesh (marked with number 3 in Figure 66). The menu hides the content of this level completely, so we had to feed the view that we wanted to display, from the Level to our user interface.

The preview of the selected camera mode and vehicle mesh is rendered on the user interface by applying the implementation explained in the 'Displaying a View from the Level in the User Interface' section.

3.2.3.1.2. Driving System

We implemented the driving system by following the 'Vehicle User Guide' of UE4 [46]. This process of implementing a fully functional vehicle contains creating a certain group of assets: a Skeletal Mesh and a Physics Asset of the vehicle, an animation blueprint with its parent class as 'VehicleAnimInstance' in order to animate the vehicle according to the movement input, a vehicle blueprint with its parent class as 'WheeledVehicle', one or more blueprints for the wheels with their parent class as 'VehicleWheel' and a Tire-Config data asset. With the help of these individual assets, we can configure the vehicle on a very high level of detail by i.e. being able to control the behavior of each wheel, the vehicle's mass and its components such as the chassis, gear, etc. We decided not to include these high-detail vehicle settings in the user interface in order to simplify the configuration process.

We assigned the W,A,S,D and Space keyboard keys for the movement input.

3.2.3.1.3. Animating a Vehicle

We used the modeling software Blender [47] in order to rig the vehicle which we ultimately decided to use in our simulation. Rigging the vehicle is mandatory because a skeletal mesh is needed in order to utilize the 'VehicleAnimInstance' parent class for animation blueprints. [46]

The skeletal mesh must have one bone for each wheel and the bones for the wheels must all be parented to a fifth root bone, which must be positioned on the vehicle's center of mass.

In order to execute the rigging process in Blender, in first place the 'Unit Scale' Scene Property [48] must be set to 0.01 and the 'Clip End' Camera Property [49] must be set to 100m, so that the result is utilizable in UE4. The bones are implemented with the 'Armature' object type in Blender. "An 'armature' is a type of object used for rigging. A rig is the controls and strings that move a marionette (puppet). Armature object borrows many ideas from real-world skeletons." [50] The local axes of the armatures must match with the global axes in the blender environment. Finally, the complete vehicle must be parented to the Armature object and the result must be exported as an FBX file.

The export options must be set as the following:

1. Main Settings

   a. The forward- and up axes must be set correctly to the implementation in blender

   b. Only the Armature and Mesh options must be chosen

2. Geometry Settings

   a. The smoothing groups must be set to 'Face'

   b. The 'Apply Modifiers' checkbox must be checked

3. Armature Settings

   a. The 'Add Leaf Bones' checkbox must be unchecked

The exported FBX file must ultimately be imported in UE4 by checking the 'Skeletal Mesh' checkbox in the Import Settings because otherwise UE4 does not generate a Skeletal Mesh or a Physics Asset for the imported FBX file.

3.2.3.1.4. Metahumans in Vehicle System

The vehicle blueprint that we have implemented returns a world location for each vehicle seat. This way we can precisely position our Metahumans in the vehicle. We implemented the same logic for car doors as location indicators for Metahumans, so that they can first walk to the car door and then be seated on the vehicle seat. Later on we decided to exclude the walking feature, as explained in the Metahumans section.

3.2.3.2. Camera System

We offer four pre-defined camera positions as well as the option to define a new camera position (and rotation) in order to keep the configuration of the simulation as intuitive as possible while granting designers the highest degree of freedom.

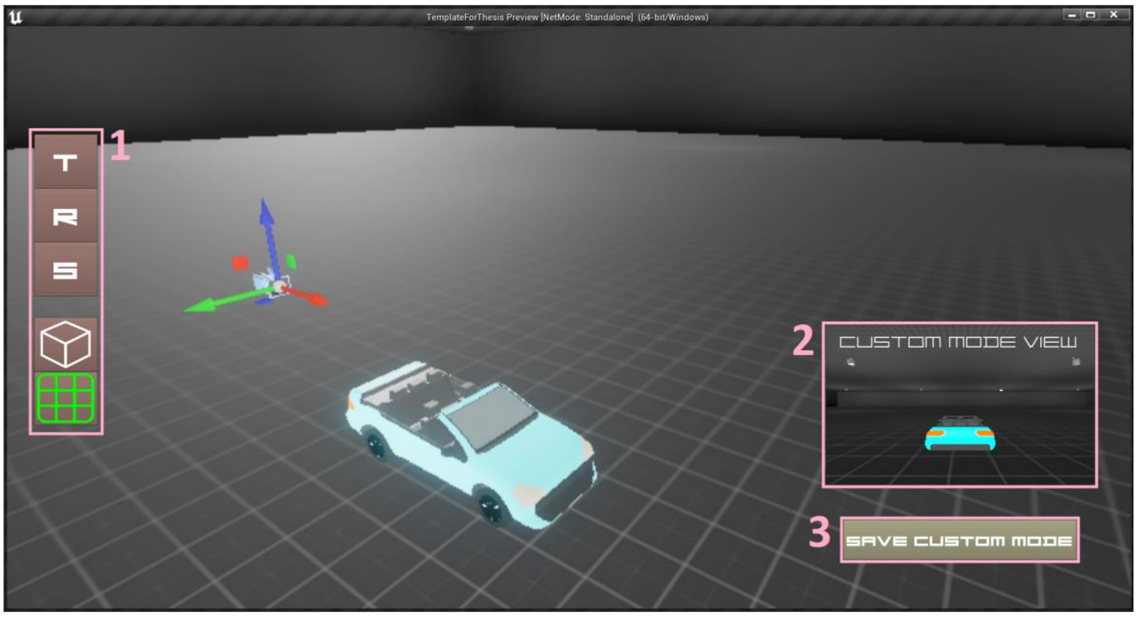

TemplateForThesis Preview [NetMode: Standalone] (64-bit/Windows)
1
T
R
S
2
CUSTOM MODE VIEW
3
SAVE CUSTOM MODE

Figure 17: UE4 Level to Create a Custom Camera Mode

Definition of the markers on Figure 17:

1. Buttons to change the Transformation Mode of the Transform gizmo

2. Up-to-date display of the currently edited Custom Camera Mode

3. Button to finalize configuring the Custom Camera Mode, save it in the next free slot and return to the menu page

3.2.3.2.1. User Interface

As shown in Figure 16, the user interface to configure the camera settings in the vehicle- and camera configuration menu page consists of a drop-down menu (marked with number 2 in Figure 16), where designers can select the camera mode, and a button which takes designers to a second level (marked with number 4 in Figure 16), where designers can create a custom camera mode. The drop-down menu (marked with number 2 in Figure 16) does not contain the Custom Mode options which have not been created yet.

As shown in Figure 17, the user interface which we present designers to edit and create a custom camera mode consists of three main sections.

The first section (marked with number 1 in Figure 17) consists of buttons. These buttons adjust how the 3D transform gizmo manipulates the 3D camera mesh. Button ‘T’ stands for ‘Translation’, ‘R’ for ‘Rotation’ and ‘S’ for ‘Scale’. If designers press on button ‘T’, the transform gizmo is adjusted so that it serves to change the position of the 3D camera mesh. Buttons ‘R’ and ‘S’ function in the same principle as the button ‘T’.

The top first button, with a cube image on it, of the remaining two buttons of this section (marked with number 1 in Figure 17) serve to toggle the (World/Local) Space attribute of the transform gizmo. The cube image on the button indicates that the current mode of the transform gizmo is Local Space. The image is toggled to a 2D-Earth image if the current mode is World Space.

The latter button of the remaining two buttons, with a green matrix-like image on it, serves to toggle the 'Grid Snapping' function of the transform gizmo. "When grid snapping is enabled in Unreal Engine, an Actor will move, rotate, or scale in increments of a specific value." [51] Our simulation does not provide an option to define the 'step size' of the incrementation when 'Grid Snapping' is active. The green color on the matrix-like image on this button indicates that the 'Grid Snapping' function is active. The color of this image is toggled to gray if 'Grid Snapping' is deactivated.

In this level the keyboard keys 'W', 'E', 'R' serve as hotkeys for buttons 'T', 'R', 'S', respectively.

The second section (marked with number 2 in Figure 17) serves as a display of the currently edited camera mode. The preview of the currently edited custom camera mode is rendered on the user interface by applying the implementation explained in the 'Displaying a View from the Level in the User Interface' section. We decided to integrate this display in our user face in order to increase the accuracy and efficiency of designers by simultaneously previewing the result of their configuration.

The third and final section (marked with number 3 in Figure 17) is a single button. When clicked, it saves the relative transform data of the recently created camera mode in the GameInstance Blueprint and opens the vehicle- and camera configuration menu page. The option to choose the recently created custom camera mode is eventually added in the drop-down (marked with number 2 in Figure 16).

3.2.3.2.2. Pre-Defined Camera Modes

The four pre-defined camera modes that we present are: First Person View Mode, Driver-Shoulder View Mode, Third Person View Mode and the Top-Interior View Mode. Each mode has its own advantages in creating and strengthening the atmosphere, which the designers aim to make the users engage in.

The Third Person View makes it possible for designers to enhance the environmental effect of the simulation and to attract the user's attention to the car's exterior features. This camera mode functions according to the camera systems of traditional third-person action games, where the camera follows its target (in our case the car) at an upper angle from behind. Additionally, we utilized Unreal Engine’s ‘Spring Arm’ actor-component [52], so that the line-of-sight of the camera is never interrupted during the simulation.

The First Person View emphasizes the driving experience by supporting a view angle which arguably puts the designers in the driver’s shoes.

The Driver’s-Shoulder View, while being similar to the First Person View, presents the driving experience from the perspective of a passenger who is sitting on one of the back seats of the car. The characteristic of this camera mode might possibly be explained as putting the designers in the place of a child who is excitedly watching the road through the front seats.

The Top-Interior View almost fully emphasizes the features of the dashboard. It points the user’s focus to the control panel in order to display the interior properties of the vehicle. It can be said that this mode is a modified version of the Driver-Shoulder View, being more technical and less immersive.

3.2.3.2.3. Custom Camera Modes

We provide designers the option to configure the camera location and rotation for a custom camera mode. Designers can create and save up to four custom view modes and use them in the same way as they use the pre-defined camera modes. If the option to create a custom camera mode gets chosen, our simulation presents designers a different level with a car mesh and a camera mesh, where they can edit the camera’s location and rotation by using a runtime transform gizmo, as well as see the view of the camera model in the most recent rotation and location. In case if designers choose to create a custom camera mode they do not lose any progress from their previous configuration in the menu page.

We utilized Robin Wood’s blueprint demo ‘Runtime Transform Gizmo’ [53] in order to provide designers the freedom to edit the transform of an object in 3D space. By doing so, we were able to present a system where designers do not have to register any numerical input in order to edit the location and rotation of a 3D object. We can possibly say that this is the most intuitive way to edit the location and rotation of a 3D object in a 3D space and therefore this approach is probably an accelerator for designers’ workflow.

3.3. Back-End Implementation

In this section we cover the features which we did not give place to while describing the menu sections. We also explain some of the core implementation that we have realized in order to make our project function as a whole. The implementation are utilized multiple times throughout the project and the features are not directly related to a specific menu page, thus we decided to rank these topics in their own section.

3.3.1. Game Instance Blueprint

We implemented a single 'GameInstance Blueprint' [54] (we named it BP_GameInstance) in our project in order to store and access global variables which are needed by multiple blueprints. It can be said that the GameInstance Blueprints in UE4 are implemented according to the principles of the 'Singleton Design Pattern'. "In software engineering, the singleton pattern is a software design pattern that restricts the instantiation of a class to one 'single' instance. This is useful when exactly one object is needed to coordinate actions across the system." [55]

We especially utilized it in order to store information which will be needed after Levels get switched. In Unreal Engine 4, when a new Level is opened, every Blueprint from the previous Level gets killed and new Blueprints are instantiated in the new Level, therefore the necessary information which was obtained in the previous Level gets lost. In order to prevent this we save the essential information, i.e. data about the configuration that takes place in the menu which will be required when the configuration phase gets finalized, in BP_GameInstance and later on obtain this information by accessing BP_GameInstance.

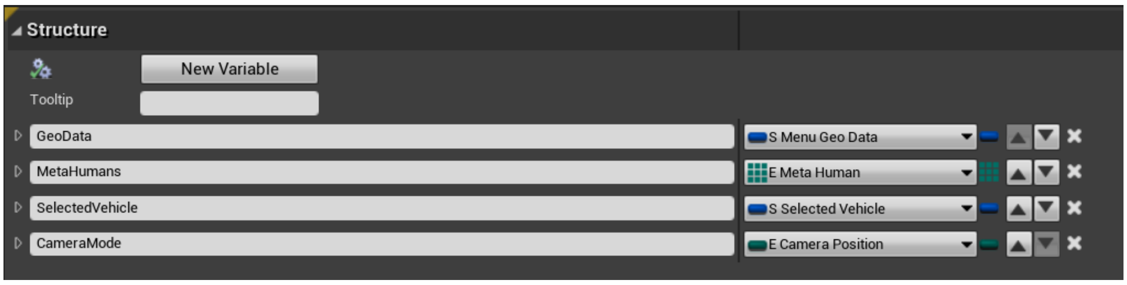
We also employ BP_GameInstance in order to execute the intercommunication of Blueprints, of which it is debatably hard to access and store an instance of. An example to such pair of Blueprints would be a Level Blueprint [56] and a Widget Blueprint [57]. We faced this scenario while implementing the 'Selected Camera Mode and Vehicle Display' (marked with number 3 in Figure 16). We had to notify the Level Blueprint when designers change the selected camera mode or the vehicle mesh in order to apply the recent changes on the Actors in the Level. Instead of attempting to access and store an instance of the Level Blueprint in the Widget Blueprint and vice versa, we decided to implement 'Event Dispatcher's [58] in BP_GameInstance. We then implemented a 'Custom Event' [59] in the Level Blueprint, bound it to the Event Dispatcher and finally called the Event Dispatcher in the Widget Blueprint. We benefited this way of distributing information among blueprints for displaying the change in selection of vehicle mesh and camera mode.

3.3.2. Data Structures

We utilized multiple Structs [60] and Enums [61] in order to organize and save information or define and separate data types.

The most fundamental usage of the data structures was to create a container for our save system. As shown in Figure 68, we divided the information about the configuration in the menu in segments which correspond to the menu pages.

Each time when a menu page gets switched, the most recent configuration in that menu page gets saved in the corresponding data container: the registered date, time and weather condition as well as the selected coordinates and the zoom factor are saved in 'GeoData' Struct, the selected Metahumans is saved in the E_MetaHuman (the Enum which we created in order to define the different Metahumans which our simulation provides) array, the selected driver is saved directly in a class variable in BP_GameInstance, the selected vehicle and the selected camera mode are saved in S_SelectedVehicle Struct and E_CameraPosition Enum, respectively.



GeoData	S Menu Geo Data
MetaHumans	E Meta Human
SelectedVehicle	S Selected Vehicle
CameraMode	E Camera Position

Figure 18: Struct to Save Configuration Data from the Menu

The rest of the data structures that we have created and utilized can be further examined in the Appendix section.

3.3.3. Displaying a View From an UE4 Level in the User Interface

We needed to display the view of an actor in a Level on the user interface for multiple scenarios throughout the project. We implemented the 'Futuristic Earth Image' in the Time- and Weather Configuration menu page (Figure 10), the 'Selected Vehicle- and Camera Mode Display' in the Camera- and Vehicle Configuration menu page (marked with number 3 in Figure 16) as well as the 'Display of the Currently Edited Custom Camera Mode' (marked with number 2 in Figure 17) by realizing this feature.

We achieved this by using a SceneCapture2D [44] and a RenderTarget [45]. “A Scene Capture 2D makes an image of the scene on every frame.” [44] The RenderTarget is used to store the image which is produced by SceneCapture2D. The RenderTarget is eventually utilized by a creating a Material and using the RenderTarget as its output. Ultimately, we use an instance of this material to render the preview of the vehicle on an Image [15] in the user interface.

3.3.4. Hotkeys for Real-Time Controls

We implemented hotkeys so that designers can change the configuration that they have executed in the menu, during the simulation. These controls contain changing the weather condition, camera mode and the daytime.

Weather Condition Hotkeys:

- (N + 1): Sets the weather condition to ‘Sunny with Cloudy Sky’
- (N + 2): Sets the weather condition to ‘Sunny with Clear Sky’
- (N + 3): Sets the weather condition to ‘Slightly Foggy’
- (N + 4): Sets the weather condition to ‘Heavily Foggy’
- (N + 5): Sets the weather condition to ‘Slightly Rainy’
- (N + 6): Sets the weather condition to ‘Heavily Rainy’
- (N + 7): Sets the weather condition to ‘Slightly Snowy’
- (N + 8): Sets the weather condition to ‘Heavily Snowy’

Camera Mode Hotkeys:

- (C + 1): Sets the camera mode to ‘First Person View’
- (C + 2): Sets the camera mode to ‘Driver’s Shoulder View’
- (C + 3): Sets the camera mode to ‘Top-Interior View’
- (C + 4): Sets the camera mode to ‘Third Person View’
- (C + 5): Sets the camera mode to ‘Custom View 1’

- (C + 6): Sets the camera mode to ‘Custom View 2’
- (C + 7): Sets the camera mode to ‘Custom View 3’
- (C + 8): Sets the camera mode to ‘Custom View 4’

The custom camera modes are initially set to the same position as the ‘Driver’s Shoulder View’, therefore using the hotkeys for a custom camera mode which had not been configured will result the camera moving to the ‘Driver’s Shoulder View’ location.

Daytime Hotkeys:

- (T + UP): Increments daytime by one hour
- (T + DOWN): Decrements daytime by one hour

3.3.5. Mapping Terrain Coordinates into Latitude/Longitude Format

Our simulation contains an externally generated and realistic terrain.

The terrain is generated depending on the coordinate- and the zoom factor inputs that are registered in the first menu page. The generated terrain has its coordinates in the [-1,1] interval. We implemented a function that maps the ‘terrain coordinates’ of the interval [-1,1] to real-world coordinates in Latitude-Longitude format. To calculate the coordinate mapping, the function takes the real-world coordinates of the middle point of the terrain and the terrain’s edge length (in km) as input. It then converts a third input, a point on the terrain in terrain coordinates [-1,1], to real world coordinates and returns it as output. This function can also easily be modified in order to interpret real-world coordinates as terrain coordinates.

We utilized Chris Veness’s implementations [62] in order to achieve geographical accuracy while calculating; the distance between two real-world locations, the initial bearing while traveling from one real-world location to another and the destination location while traveling from a real-world location with a certain bearing and distance. We needed to execute these calculations in order to compute the real-world coordinated of a point on the generated terrain, as these calculations return the necessary intermediate results. Chris Veness realized these computations in JavaScript, therefore we re-implemented them in UE4’s Blueprinting language.

3.3.6. Areas of Improvement

- The maximum number of selected Metahumans does not change depending on the selected vehicle. We decided to set the maximum number of selected Metahumans to a constant value (five) because otherwise we would either have to make designers register the vehicle meshes, which they have added to the project, in a specific data structure where they would also enter the maximum number of Metahumans allowed in that specific vehicle or we would have to implement a system which would obtain the maximum number of Metahumans just by examining the vehicle mesh. Implementing such a system in a complete way would probably not be possible when both the time limitation and our workload are taken into consideration. Besides the difficulties of having implemented such system, the theoretical reliability of exactly spotting the seat places of the vehicle and therefore precisely placing the Metahumans on the car seats remains questionable because the dimensions of a vehicle do not exactly indicate where each car seat is located.

- In the final version of our project each available vehicle is utilized by using the same steering wheel model. The first reason for this is that the vehicle model and the steering wheel model have to be two separate models so that the steering wheel's transform can be manipulated separately independently of the vehicle model. The second and actual reason which have caused this problem is that we have only utilized a single vehicle model for our simulation. Thus, because we did not have two different steering wheel models, we did not invest the time to actually change the utilized steering wheel depending on the selected vehicle model.

  The solution is to extend our implementation in such way, that, just like it has been implemented for the vehicle models, steering wheel models which are located in a certain project folder and follow a certain naming convention must be read in the beginning of the runtime, saved in the GameInstance blueprint and ultimately be assigned to the corresponding vehicle according to the final vehicle selection.

4. Results and Discussion

In this section, we will firstly present the results of the user experience questionnaire (UEQ) we conducted. Secondly, we will discuss about the final version of our system and present our insight on where our system stands in terms of usability, satisfaction of current demand in the industry and completeness.

4.1. User Experience Questionnaire (UEQ)

We utilized UEQ-S [63], the short version of the UEQ, in order to evaluate users' experience with our product. We conducted the questionnaire on 14 participants. The participants mainly consist of engineers and engineering students from multiple disciplines and mainly between the ages 20-25.

In this section, we will introduce the results of the questionnaire and briefly describe their place in evaluating our user interface in reference to the UEQ-S tool. [63] The term 'items' refers to the eight dimensions of the questionnaire. These dimensions are the perceived; supportiveness, easiness, efficiency, clearness, excitement, attractiveness of a product, as well as whether the product has been perceived as an inventive, and leading edge product.

4.1.1. Main Results

The scale means, as well as the mean and standard deviation for each item, are computed here.

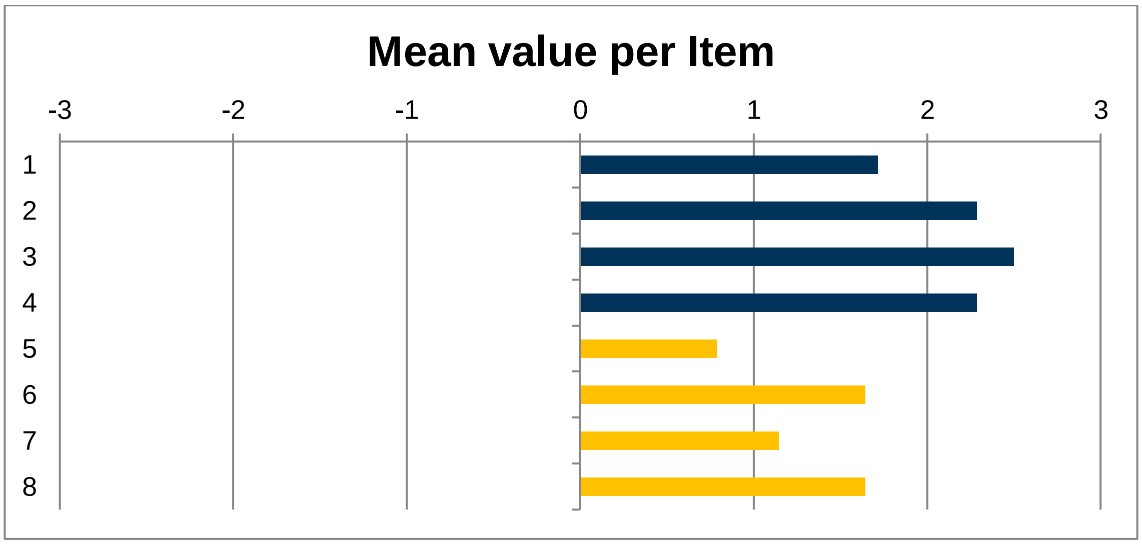


Figure 19: Mean Values of the UEQ [63]

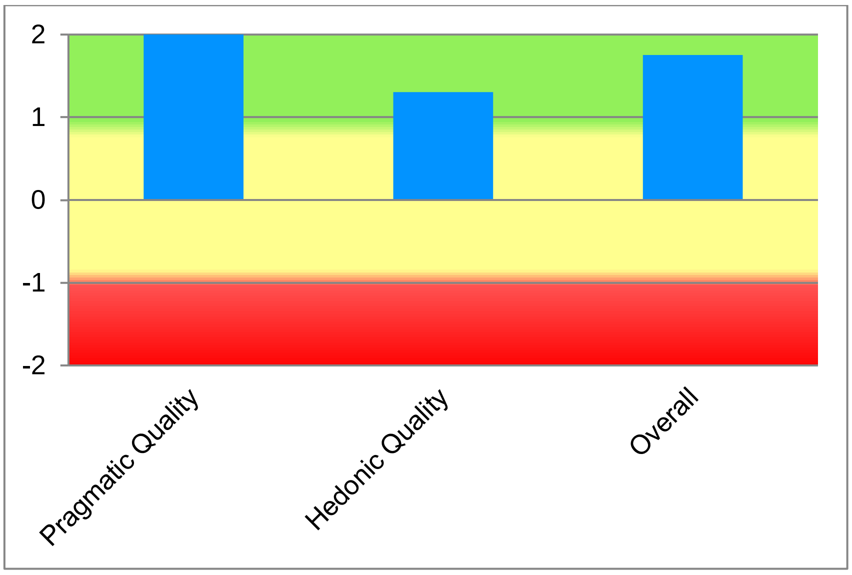


Figure 20: Pragmatic, Hedonic Overall Results of the UEQ [63]

4.1.2. Benchmark

The benchmark compares the evaluated product to the products in the benchmark data set in an easy-to-understand manner. This data collection contains information from 21175 individuals in 468 studies of various products. These benchmark data are currently based on the full UEQ... Data from the full UEQ benchmark can be used as an approximate approximation for the short UEQ, UEQ-S, benchmark because the short version's scale values are a pretty good estimate of the full version's comparable values.

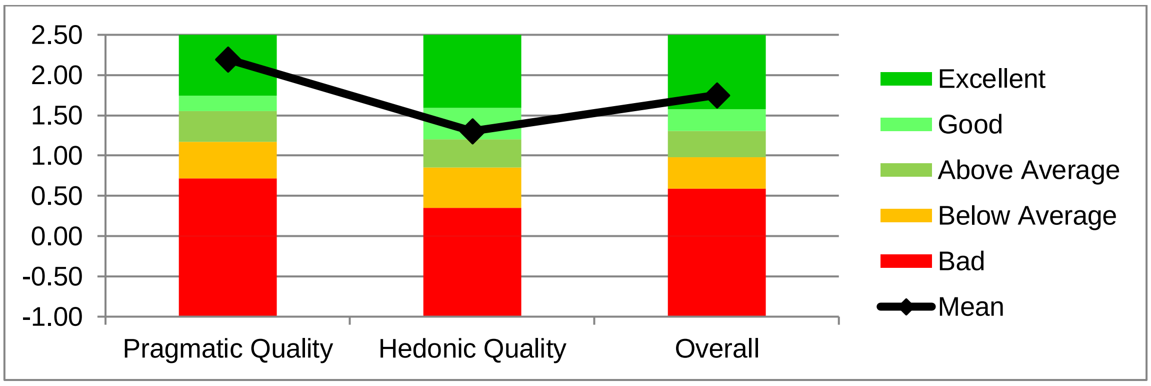


Figure 21: Benchmark Results [63]

4.1.3. Confidents Intervals for Items and Scales

Here, the scale means and the mean of each item's confidence intervals are determined. The confidence interval is a measure of the precision with which the mean is estimated.

Confidence interval (p=0.05) per item						
Item	**Mean**	**Std. Dev.**	**N**	**Confidence**	**Confidence interval**	
1	1.714	1.267	14	0.663	1.051	2.378
2	2.286	1.069	14	0.560	1.726	2.846
3	2.500	0.650	14	0.341	2.159	2.841
4	2.286	1.139	14	0.596	1.689	2.882
5	0.786	1.528	14	0.800	-0.015	1.586
6	1.643	1.082	14	0.567	1.076	2.210
7	1.143	1.292	14	0.677	0.466	1.820
8	1.643	1.447	14	0.758	0.885	2.401

Confidence intervals (p=0.05) per scale						
Scale	**Mean**	**Std. Dev.**	**N**	**Confidence**	**Confidence inter-val**	
Pragmatic Quality	2.196	0.722	14	0.378	1.818	2.575
Hedonic Quality	1.304	0.779	14	0.408	0.895	1.712
Overall	1.750	0.663	14	0.347	1.403	2.097

It may perhaps be said that the results of the questionnaire are positive regarding the users' experience with the user interface and satisfaction of participants during their interaction with the interface.

As can be seen in Figure 19, items 5 and 7, namely *boring vs exciting* and *conventional vs inventive*, have received the lowest mean values. This could be interpreted as our user interface being boring and conventional. This possibly depends on the visual aspects of the user interface. Therefore it probably be wouldn't wrong to claim that our user interface needs more aesthetic work. We suggest; animating the indicators which display the menu-state, and presenting the existing visual effects more effectively.

On the other hand, the biggest challenge with conducting this questionnaire was that it couldn't be succeeded to use the survey on automotive aesthetic designers. Since our participants do not qualify as automotive aesthetic design experts, the reliability of our questionnaire's results remain as a research question.

4.2. Reviewing our System

All in all, we might be able to say that our system’s positive contributions outweigh its shortcomings. In this section, we will address some of the design rules and general principles of usability and user experience that we provided in Section 1, namely Background Research. We will question whether our system has satisfactorily applied these rules and suggest further research in cases where it has not successfully implemented these guidelines and principles.

Firstly, we can probably claim that our system has achieved an acceptable level of coloring, information grouping, and navigation. We have given place to explanation of these issues in section 1.1.3, namely Design. As we can see in Figure 6, the Color Palette, the density of blue is fairly low, and the colors used for ‘Undo’ and ‘Proceed’ features, are not conflicting colors with their purposes.

Secondly, having split our menu content into separate menu pages has possibly helped us organize the information sequentially, while more specific information is organized functionally on a single menu page. As can be seen in Figure 13, our menu pages are organized in square-bounds, where functionally relevant controls are displayed together. Additionally, designing separate menu pages has also helped increase the performance of our system in navigation, as it arguably helps users in keeping track of their progress.

Moreover, our system lacks an ‘Undo’ action in the UE4 Level illustrated in Figure 17, where custom camera modes can be created. Without an ‘undo’ facility, designers are forced to ‘reverse’ their actions manually when a user error occurs. Nevertheless, we can state that providing ‘back’ buttons in menus and hotkeys in the simulation have contributed to our system’s performance in error handling. The significance of regarding errors during design is emphasized in both ‘Schneiderman’s 8 Golden Rules on Interface Design‘ and ‘Normans Principles 7 Principles of Transforming Complex Tasks into Simpler Ones’.

We might be able to say that our participants’ responses to the questionnaire showed us how learnable and robust our system is for the users. We could first claim that our system is learnable. We think this is due to the labeling of buttons and information, and the structure of the multi-paged layout of our user interface. Almost none of our participants had a problem executing the task after having it initially performed.

It could be evaluated as a shortcoming in terms of robustness that our system does not provide many tooltips. In this context, we used button images and labeling, which are, in our opinion, the most standardized symbols in the design industry. Our system definitely lacks a certain level of flexibility. Currently, there nothing that users can customize. The menu layout and capabilities are enforced to be executed in a certain order and have their own constraints. This also indicates that the 'system potential', section 1.2.4., of our system will not be high, as it leaves a small rom for creativity to designers. More research must be conducted in order to automatize more aspects of a user interface and simulation, and therefore offer more customization opportunities to the user.

In terms of the usability of our system, our system can perhaps be evaluated as guessable, and as discussed above, learnable. We selected icons and images for the purpose of indicating the purpose of using the menu page or specific menu element. We intend to provide a visual clue to the function of our interface's elements, which might be a contribution to our system's guessability.

But in terms of experienced user performance and system potential, our product probably has shortcomings due to the lack of variety of ways to interact with our system. Because of this, there might possibly be less room for creativity and thus less potential to achieve with our system.

Finally, we only conducted a summative evaluation, mentioned in section 1.1.4., of users' interaction with our system. This means that, our results could only provide an insight where did not have the resources to redesign and to iterate anymore. We use this insight to formally review our system and suggest further research on certain issues. Therefore, a formative evaluation and redesign could be effective when applied to our system.

All things considered, we can perhaps claim that our system satisfies some of the industry's demands, such as realism, usability, and interactivity, but it needs further research on flexibility, where more aspects of the system are automated and users are granted more excessive and flexible control over the system.

Summary

In conclusion, we have possibly succeeded in creating an innovative and contributive design evaluation tool by using state-of-the-art technologies and implementing a tool that might perhaps cover the shortcomings of similar state-of-the-art tools. But further research and development seems necessary so that the system can be employed by experts in industry. These include granting designers more extensive control over how the system interacts with the user, designing a more innovative user interface by avoiding making conventional design choices, and redesigning while considering error recoverability. Executing a formative evaluation of our system and redesigning according to the evaluation results also remains a research area. The impact of applying inferences from expert opinions to the design of the system and then building a new prototype out of it would probably be large.

Appendix

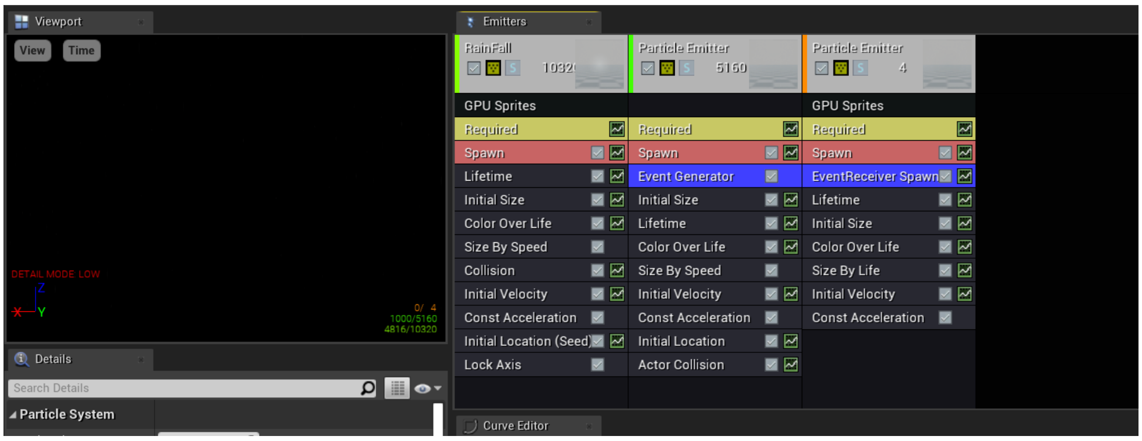


Figure 19: Particle System for Rainy Weather

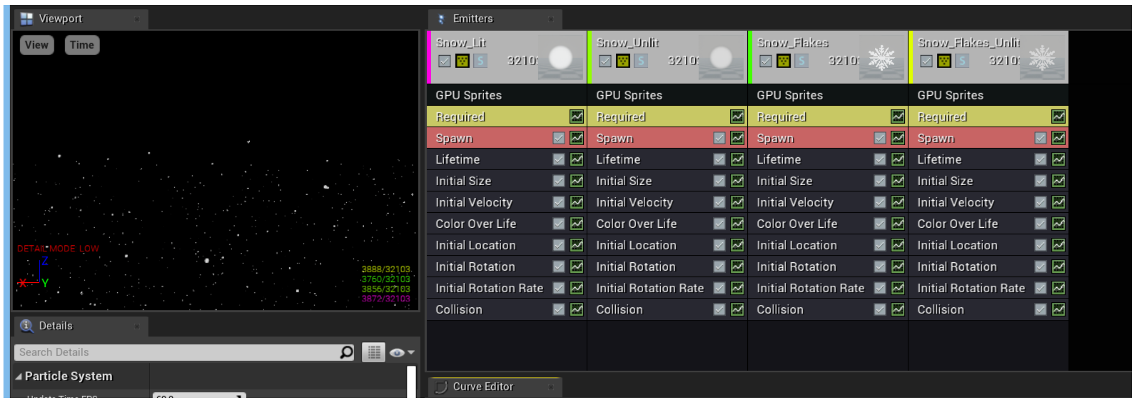
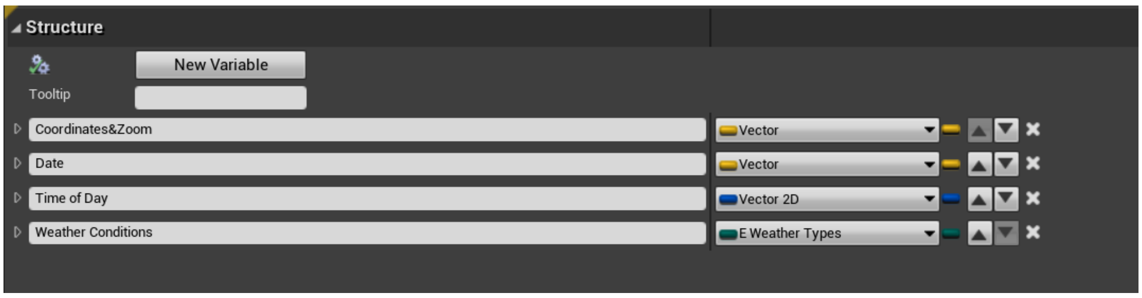
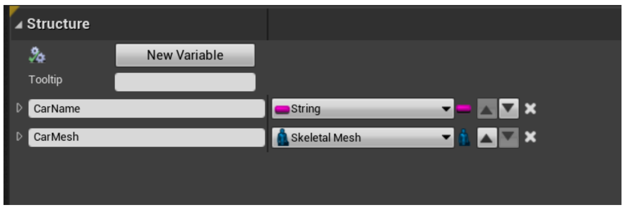


Figure 20: Particle System for Snowy Weather



Coordinates&Zoom	Vector
Date	Vector
Time of Day	Vector 2D
Weather Conditions	E Weather Types

Figure 21: Struct for Saving the Geographical Configuration Input

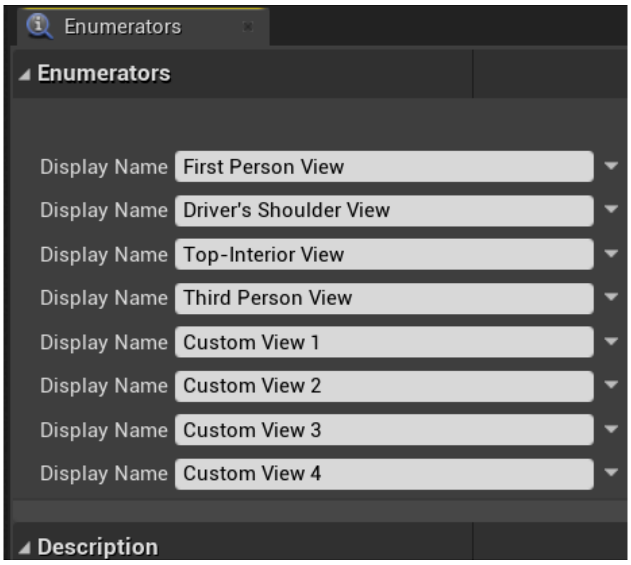

Structure

New Variable

Tooltip

CarName	String
CarMesh	Skeletal Mesh

Figure 22: . Struct for Saving the Selected Vehicle Input

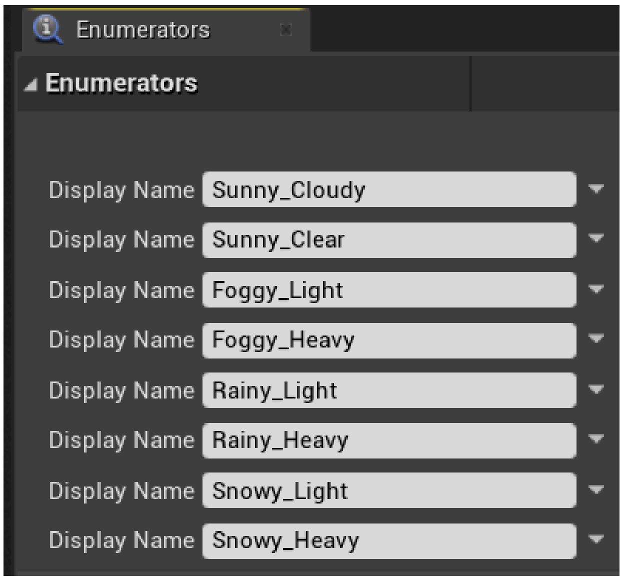

Enumerators

Enumerators

Display Name	First Person View
Display Name	Driver's Shoulder View
Display Name	Top-Interior View
Display Name	Third Person View
Display Name	Custom View 1
Display Name	Custom View 2
Display Name	Custom View 3
Display Name	Custom View 4

Description

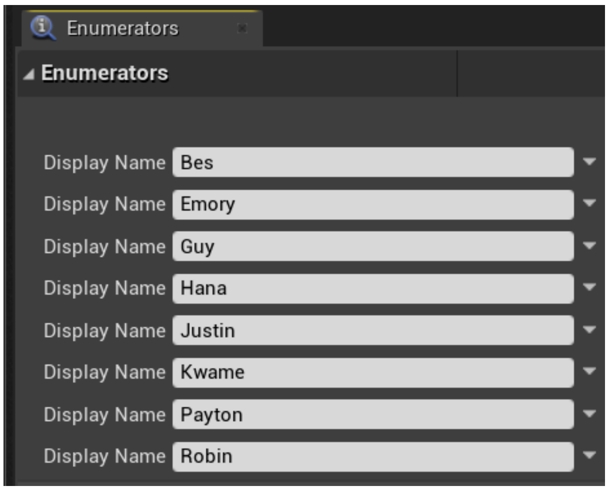
Figure 23: Enum for Defining Camera Modes


Enumerators

Enumerators

Display Name	Sunny_Cloudy
Display Name	Sunny_Clear
Display Name	Foggy_Light
Display Name	Foggy_Heavy
Display Name	Rainy_Light
Display Name	Rainy_Heavy
Display Name	Snowy_Light
Display Name	Snowy_Heavy

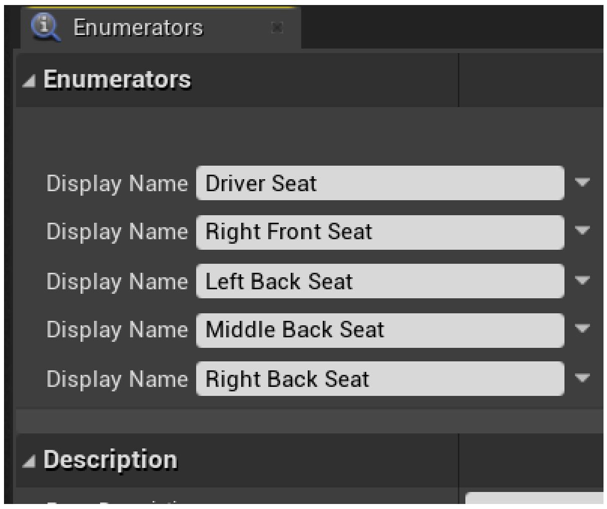
Figure 24: Enum for Defining Weather Types


Enumerators

Enumerators

Display Name	Bes
Display Name	Emory
Display Name	Guy
Display Name	Hana
Display Name	Justin
Display Name	Kwame
Display Name	Payton
Display Name	Robin

Figure 25: Enum for Defining the Provided Metahumans


Enumerators

Enumerators

Display Name	Driver Seat
Display Name	Right Front Seat
Display Name	Left Back Seat
Display Name	Middle Back Seat
Display Name	Right Back Seat

Description

Figure 26: Enum for Defining Car Seats

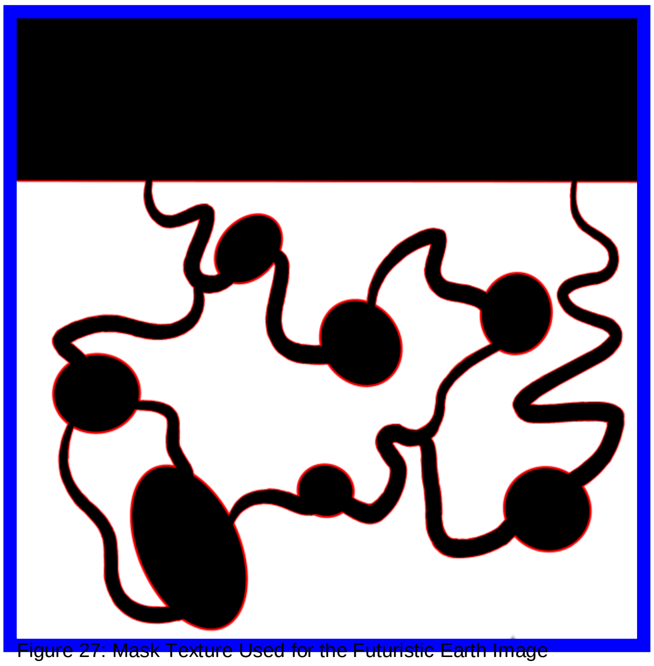


Figure 27: Mask Texture Used for the Futuristic Earth Image

References

[1] P. W. Jordan: An Introduction to Usability. CRC Press Taylor & Francis Group, 2002.

[2] A. Dix, J. Finlay, G. D. Abowd and R. Beale: Human-Computer Interaction. Pearson Education Limited, 1993.

[3] R. Harper, T. Rodden, Y. Rogers and A. Sellen: Being human: human-computer interaction in the year 2020. Microsoft Research Ltd, 2008.

[4] S. K. Card, T. P. Moran and A. Newell: The Psychology of Human-Computer Interaction. CRC Press Taylor & Francis Group, 1983.

[5] ACM Special Interest Group on Computer–Human Interaction Curriculum Development Group. ACM SIGCHI curricula for human–computer interaction. Technical report, ACM, New York, 1992.

[6] N.Bevan, J. Kirabowski and J. Maissel: What is Usability?. In Proceedings of the 4th International Conference on HCI, Stuttgart, September 1991.

[7] J. R. Lewis: Usability Testing. In book: Handbook of Human Factors and Ergonomics, Third Edition (pp.1275 - 1316) Edition: 3rd , Chapter: 49, Publisher: J. Wiley. 2006.

[8] N. Bevan: What is the difference between the purpose of usability and user experience evaluation methods ?. In Proceedings of the Workshop UXEM, 2009.

[9] J. Maxfield, P. M. Dew, J. Zhao, N. Juster and M. Fitchie: A Virtual Environment for Aesthetic Quality Assessment of Flexible Assemblies in the Automotive Design Process. SAE International, 2002.

[10] A. Kulkarni, A. Kapoor, M. Iyer and V. Kosse: Virtual prototyping used as validation tool in automotive design. 19th International Congress on Modelling and Simulation, Perth, Australia, 2011.

[11] A. Warell, K. Young: Interior aesthetics: an experience-focused approach for the design of brand-specific automotive identity. Int. J. Vehicle Design, Vol. 55, Nos. 2/3/4, 2011.

[12] Y. Lin, W.J. Zhang: Aesthetic Design for Automobile Interiors: Critical Issues and Conceptual Framework . IEEE International Conference on Systems, Man and Cybernetics, 2004.

[13] Epic Games, Unreal Engine 4’s Material. https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Materials/

[14] Epic Games, Unreal Engine 4’s Material Domain. https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/MaterialProperties/1_5/

[15] Epic Games, Unreal Engine 4’s Image. https://docs.unrealengine.com/4.27/en-US/API/Runtime/UMG/Components/UImage/

[16] Epic Games, Unreal Engine 4’s Text Block. https://docs.unrealengine.com/4.26/en-US/BlueprintAPI/Utilities/Text/

[17] Free-To-Use Text Fonts. https://www.dafont.com/de

[18] Epic Games, Unreal Engine 4’s Material Instance. https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Materials/HowTo/Instancing/

[19] Epic Games, Unreal Engine 4’s Material Blend Modes. https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Materials/MaterialProperties/BlendModes/

[20] Epic Games, Unreal Engine 4’s Panner Material Expression Node. https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Materials/HowTo/AnimatingUVCoords/

[21] Epic Games, Unreal Engine 4’s Tessellation. https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/MaterialProperties/1_8/

[22] Epic Games, Unreal Engine 4’s World Position Offset. https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/MaterialNodes/1_10/

[23] Wikipedia. Voronoi. https://en.wikipedia.org/wiki/Voronoi_diagram

[24] Inigo Quilez. Equidistant Voronoi Edges. https://www.iquilezles.org/www/articles/voronoilines/voronoilines.htm

[25] Epic Games, Unreal Engine 4’s Geographically Accurate Sun Positioning. https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/LightingAndShadows/SunPositioner/

[26] Epic Games, Unreal Engine 4’s Particle System. https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/ParticleSystems/UserGuide/

[27] Epic Games, Unreal Engine 4’s Exponential Height Fog. https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/FogEffects/HeightFog/

[28] Epic Games, Unreal Engine 4’s Volumetric Clouds. https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/LightingAndShadows/VolumetricClouds/

[29] NASA, Global Maps: Land Surface Temperature. https://earthobservatory.nasa.gov/global-maps/MOD_LSTD_M

[30] Epic Games, Unreal Engine 4’s Scroll Box Widget. https://docs.unrealengine.com/4.26/en-US/API/Runtime/UMG/Components/UScrollBox/

[31] Epic Games, Metahuman. https://docs.metahuman.unrealengine.com/en-US/

[32] Non Player Character (NPC). https://en.wikipedia.org/wiki/Non-player_character

[33] Epic Games, Metahuman Creator. https://www.unrealengine.com/en-US/metahuman-creator

[34] Epic Games, Metahuman Creator: Blending. https://docs.metahuman.unrealengine.com/en-US/UserGuide/Face/

[35] Quixel, Bridge. https://quixel.com/bridge

[36] Quixel, Bridge. https://help.quixel.com/hc/en-us/articles/115000613105-What-is-Quixel-Bridge-

[37] Epic Games’ Marketplace, MCO Mocap Basics. https://www.unrealengine.com/marketplace/en-US/product/28fc3cc4332541e3b0037d67a65e5d6d

[38] Adobe, Mixamo. https://www.mixamo.com/#/

[39] Epic Games, Unreal Engine 4’s Animation Library. Two Bone IK. https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/SkeletalMeshAnimation/NodeReference/SkeletalControls/TwoBoneIK/

[40] Kemna, Tom. Inverse kinematics for fingers explanation and example in Unreal Engine 4.21. https://www.gamedeveloper.com/programming/inverse-kinematics-for-fingers-explanation-and-example-in-unreal-engine-4-21

[41] Epic Games, Unrel Engine 4’s Character Movement Component. https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Networking/CharacterMovementComponent/

[42] Epic Games, Unreal Engine 4’s NavMesh. https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/NavMesh/

[43] IsaraTech, UE4 – Get Assets by Path in Blueprints with the AssetRegistry. https://isaratech.com/ue4-get-assets-by-path-in-blueprints-with-the-assetregistry/

[44] Epic Games, Unreal Engine 4’s SceneCapture2D. https://docs.unrealengine.com/4.26/en-US/Resources/ContentExamples/Reflections/1_7/

[45] Epic Games, Unreal Engine 4’s RenderTarget. https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/RenderTargets/

[46] Epic Games, Vehicle User Guide. https://docs.unrealengine.com/4.26/en-US/InteractiveExperiences/Vehicles/VehicleUserGuide/

[47] Blender. https://www.blender.org

[48] Blender, Scene Properties: Unit Scale. [https://docs.blender.org/manual/en/latest/scene_layout/scene/properties.html

[49] Blender, Camera Properties: Clip End. https://docs.blender.org/manual/en/latest/render/cameras.html

[50] Blender,Armature.https://docs.blender.org/manual/en/latest/animation/armatures/introduction.html#your-first-armature

[51]Epic Games, UE4’s Grid Snapping. https://docs.unrealengine.com/4.27/en-US/Basics/Actors/ActorSnapping/

[52] Unreal Engine 4, Actor-Component: Spring Arm. https://docs.unrealengine.com/4.26/en-US/InteractiveExperiences/UsingCameras/SpringArmComponents/

[53] Wood, Robin. UE4RuntimeTransformGizmo. https://github.com/robinwood3d/UE4RuntimeTransformGizmo/blob/master/README.md

[54] Epic Games, Unreal Engine 4’s GameInstance Blueprint Class. https://docs.unrealengine.com/4.27/en-US/API/Runtime/Engine/Engine/UGameInstance/

[55] Wikipedia. Singleton Pattern. https://en.wikipedia.org/wiki/Singleton_pattern

[56] Epic Games, Unreal Engine 4’s Level Blueprint. https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Types/LevelBlueprint/

[57] Epic Games, Unreal Engine 4’s Widget Blueprint. https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/UMG/UserGuide/WidgetBlueprints/

[58] Epic Games, Unreal Engine 4’s Event Dispatcher. https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/UserGuide/EventDispatcher/

[59] Epic Games, Unreal Engine 4’s Custom Event. https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Events/Custom/

[60] Epic Games, Unreal Engine 4’s Structs. https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/GameplayArchitecture/Structs/

[61] Microsoft’s Documentation on C++, Enumerations. https://docs.microsoft.com/en-us/cpp/cpp/enumerations-cpp?view=msvc-170

[62] C. Veness: chrisveness/geodesy · GitHub.

[63] User Experience Questionnaire (UEQ). https://www.ueq-online.org/

[64] J. H. Oetjens, N. Bannow, M. Becker, O. Bringmann, A. Burger, M. Chaari, S. Chakraborty, R. Drechsler, W. Ecker, K. Grüttner, Th. Kruse, C. Kuznik, H. M. Le, A. Mauderer, W. Müller, D. Müller-Gritschneder, F. Poppen, H. Post, S. Reiter, W. Rosenstiel, S. Roth, U. Schlichtmann, A. von Schwerin, B. -A- Tabacaru, A. Viehl: Safety Evaluation of Automotive Electronics Using Virtual Prorotypes: State of the Art and Research Challenges. DAC '14: Proceedings of the 51st Annual Design Automation Conference, 2014.