FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Adapting Digitalized 3D Models for Interactive Virtual Environments

Bruno Miguel Dias Madeira

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Tiago Pinheiro Neto Jacob

Co-Supervisor: Eduardo Miguel Campos Magalhães

July 24, 2018

Adapting Digitalized 3D Models for Interactive Virtual Environments

Bruno Miguel Dias Madeira

Mestrado Integrado em Engenharia Informática e Computação

July 24, 2018

Abstract

3D digitalized models, such as the ones obtained through photogrammetry, are virtual copies of real object with high visual fidelity. Thus, such models can be used to create very realistic virtual environments. However, they are more rigid than traditional models in the sense that they may contain several visually distinguishable objects that share the same properties with each other (material, texture, geometry). Their meshes are not optimized for performance or realistic animation, and lighting may be baked on the texture, constricting the dynamism of a scene. Consequently, in contrast with their visual presentation, there is a lack of realistic interaction that works against user immersion.

To tackle this problem a novel conceptual workflow is proposed as a possible alternative to traditional workflows for the conception of interactivity on 3D digitalized models and their integration into virtual environments. This workflow is primarily tailored for models of small objects whose interactive components resemble graphical user interfaces. To validate the workflow, a prototype is implemented as an Unity extension that makes use of Blender for heavy, mesh related, tasks. A 3D digitalized model of a MIDI controller is used as test model.

The workflow and the prototype are tested usability wise, using the system usability scale questionnaire and additional queries. The users, that partake in usability tests, are divided into two sets with respect to their experience with 3D modeling and related areas. The data gathered from the experienced users set is used to compare the proposed workflow against a more traditional workflow.

Despite results of these tests not being conclusive due to the lack of valid quantitative data, they indicate that, at least, the proposed workflow has potential to replace or complement more traditional approaches. The user feedback received provides some indications regarding the desired features in integration environments that follow the proposed workflow.

Resumo

Os modelos digitalizados em 3D, como os obtidos através de técnicas fotogramétricas, são cópias virtuais de objetos reais com elevada fidelidade visual. Consequentemente, estes modelos podem ser usados para criar ambientes virtuais muito realistas. No entanto, eles são mais rígidos do que os modelos tradicionais, no sentido em que eles podem conter vários objetos visualmente distinguíveis que compartilham as mesmas propriedades entre si (material, textura, geometria). As malhas não são otimizadas para o desempenho ou animações realistas, e a iluminação pode estar embutida na textura, limitando o dinamismo de uma cena virtual. Assim, em contraste com sua apresentação visual, há uma falta de interação realista que influência negativamente a imersidade em ambientes virtuais.

Para resolver este problema, é proposto um novo fluxo de trabalho como uma possível alternativa a metodologias tradicionais para a conceção de interatividade em modelos 3D digitalizados e para a sua integração em ambientes virtuais. Este fluxo de trabalho é principalmente focado em modelos de pequenos objetos cujos componentes interativos se assemelham a "interfaces gráficas de utilizador" (GUI - Graphical User Interface). Para validar a metodologia proposta, um protótipo, baseado nesta, é implementado como uma extensão do Unity que faz uso do Blender para tarefas pesadas, relacionadas com a malha do modelo. É usado como modelo de teste um modelo 3D digitalizado de um controlador MIDI.

O fluxo de trabalho e o protótipo são testados em termos de usabilidade, usando o questionário de "escala de usabilidade do sistema" (SUS - System Usability Scale) e questionários adicionais. Os utilizadores que participam nos testes de usabilidade são divididos em dois conjuntos de acordo com a sua experiência em modelação 3D e áreas relacionadas. Os dados obtidos do conjunto de utilizadores experientes são usados para comparar a metodologia proposta a um fluxo de trabalho mais tradicional.

Apesar dos resultados dos testes não serem conclusivos devido à falta de dados quantitativos válidos, eles indicam que, pelo menos, o fluxo de trabalho proposto tem potencial para substituir ou complementar abordagens mais tradicionais. O feedback dos utilizadores também fornece algumas indicações sobre as funcionalidades desejadas em ferramentas de integração que sigam o fluxo de trabalho proposto.

Acknowledgements

Firstly, I would like to thank my Supervisors, João Jacob and Eduardo Magalhães, for the dissertation proposition, the freedom and trust entrusted me during this project and their feedback which was essential in the writing of this document.

Secondly to the Graphics, Interaction and Gaming Group at the Faculty of Engineering of the University of Porto which provided a work environment, and some needed connections and tools.

I would also like to thank the online communities of StackExchange, Unity Forum, and Blender Artists Forum, whose unsung heroes have helped me break trough some of the implementation brick walls. I'd like to thank Bastian Molkenthin's for the sunshine2k website and GitHub user kugelrund for making publicly available the segmentation algorithm implementation which was adapted for this project.

Finally, I would like to thank my father for the financial support during my academic endeavors, my mother for the special support given during the final sprint, and my friends for (questionably) helping me keep my sanity during the entire process.

Bruno Miguel Dias Madeira

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Goals . . . 2
- 1.3 Document Structure . . . 4

2 State of the Art **5**
- 2.1 Virtual Reality Environments . . . 5
  - 2.1.1 Immersion . . . 6
  - 2.1.2 Interaction . . . 8
  - 2.1.3 3D Representation . . . 9
- 2.2 Model Digitalization and Integration . . . 12
  - 2.2.1 Digitalization methods and limitations . . . 12
  - 2.2.2 Digitalization Workflow . . . 13

3 Workflow Specification **17**
- 3.1 Model Digitalization . . . 18
- 3.2 Automatic Segmentation . . . 19
- 3.3 Manual Segmentation . . . 20
  - 3.3.1 Segmentation by Painting . . . 20
  - 3.3.2 Enclosing Volumes Segmentation . . . 23
- 3.4 Interactivity and other components' properties . . . 24
  - 3.4.1 Automation . . . 24
  - 3.4.2 Properties Set . . . 26
  - 3.4.3 Information Display . . . 27

4 Implementation **29**
- 4.1 Implemented Workflow . . . 29
- 4.2 Model Digitalization and Refinement . . . 31
- 4.3 Final Prototype . . . 33
  - 4.3.1 External Scripts Tool . . . 33
  - 4.3.2 Disjoint Property Sets (Groups) and Segments (Elements) . . . 34
  - 4.3.3 Automatic Segmentation . . . 36
  - 4.3.4 Manual Segmentation . . . 38
  - 4.3.5 Applying a Segmentation . . . 40
  - 4.3.6 3D Handles and Annotations . . . 41
- 4.4 Initial Prototype Related Experiments . . . 42
  - 4.4.1 Image Based Segmentation . . . 42

5 Validation **45**
5.1 User Tests . . . 45
5.1.1 Non Modelers Validation Test . . . 47
5.1.2 Modelers Validation Test . . . 48
5.1.3 SUS Results . . . 49
5.1.4 Modelers Only Questionnaire Results . . . 49
5.1.5 Shared Questionnaire Results . . . 50
5.1.6 Prototype's Usability Problems and Possible Improvements . . . 52
5.2 Modified Segmentation Algorithm Segmentation Results . . . 53

6 Conclusions and Future Work **57**
6.1 Contributions . . . 57
6.2 Future Work . . . 58
6.2.1 Workflow and Prototype . . . 58
6.2.2 Future Validation Attempts . . . 60

7 Appendix **69**
7.1 External Scripts XSD . . . 69
7.2 Segmentation Binary File Structure . . . 73
7.3 Questionnaire for any user profile . . . 74
7.4 SUS Results . . . 75
7.5 Questionnaire (Modelers Only) Results . . . 77
7.6 Questionnaire (Any User Profile) Questions 4 and 6 Results . . . 79
7.7 Questionnaire (Any User Profile) Remaining Questions Results . . . 81
7.7.1 Question 1 - In Production Problems . . . 81
7.7.2 Question 2 - Workflow Advantages . . . 81
7.7.3 Question 3 - Workflow Disadvantages . . . 82
7.7.4 Question 5 - Suggestions . . . 82
7.8 Original, Untranslated User Tests' Consent Forms . . . 84

List of Figures

1.1 Photogrametric 3D model assets used in Start Wars Battle Front. Taken from DICE 2016 . . . 2
1.2 Oxygen 8 M-Audio MIDI keyboard . . . 3

2.1 AIP-Cube from Zeltzer 1992 . . . 7
2.2 A grasp based interaction device prototype. Extracted from Kry et al. 2008. . . . 9
2.3 Example of a simple 3d model represented in the OBJ format. At the left is presented the model defined by the OBJ at the right. The presented OBJ only specifies the vertices and theirs indices. Image and source OBJ from "OBJ Files - A 3D Object Format" 2018. . . . 10
2.4 Semantic annotation of automatically segmented model parts using OWL. Cutout from Attene et al. 2009. . . . 11
2.5 Retrieval of semantically similar 3d shapes, an example application of semantic representation in CAD context. Cutout from Qin et al. 2016. . . . 11
2.6 Manual retopology of a model using the Mira "Mira Tools" 2018 plugin for Blender "Blender" 2018. The new mesh, highlighted in the image, is snapped to an existing mesh surface. . . . 15
2.7 Composite textures (texture patching related). The right picture shows a virtual reforestation of the scene on the left picture. Taken from Zalesny, Maur, and Van Gool 2001 . . . 15

3.1 Blender's modifiers preview. . . . 17
3.2 Conceptual Diagram of the proposed workflow. . . . 18
3.3 GrabCut approach. Taken from Brown 2008. The white and yellow stripes and dot were user inputed and the red and blue parts are the resulting segmentation. . . . 20
3.4 Autodesk 3d Max 2D and 3D UV texture painting tools. . . . 21
3.5 An example of texture based volumetric painting approach used by Autodesk 3D Max Software. . . . 22
3.6 A "fake" depiction of a "surface" painting approach. . . . 22
3.7 An example of a perspective painting approach used in Blender. . . . 23
3.8 Components movement (green arrows) with respect to the up vector (blue arrow). From left to right are displayed the following types of components: button, knob and thumb stick . . . 25
3.9 The different types of property sets organization. . . . 27
3.10 Digital sketch of 3D billboard annotations over a keyboard. . . . 28

4.1 Diagram of the implemented workflow. . . . 30

4.2 At the left Zephyr's GrabCut mask feature. At the right the Kinect Developer Kit Fusion Explorer demo project. Both taken when attempting to digitize models for this project. . . . 32
4.3 Digitalized model of a tv remote over a table, using Kinect. . . . 32
4.4 The left image is the original captured model with all major background elements already removed. The right picture is the modified version of the model. . . . 33
4.5 Unity's custom inspector showing the script's XML defined parameters. . . . 34
4.6 Prototype's groups editor. . . . 35
4.7 Prototype's elements editor and "same name" warning. . . . 35
4.8 Visual summary of the segmentation algorithm . . . 37
4.9 Segmentation preview via texture of an automatically segmented cube in Unity. . 37
4.10 Final prototype manual segmentation with a spherical brush . . . 38
4.11 Segmentation preview displaying texture bleeding . . . 39
4.12 Popup with additional features. . . . 40
4.13 Prototype's hierarchy mutually exclusive options. . . . 41
4.14 Visual representation of the script on the scene. . . . 41
4.15 3D adjustment tools, implemented using Unity's Handles. . . . 42
4.16 OpenCV image based segmentation of a grayscale version of the model texture (after refinements). . . . 43
4.17 Segmentation from retopologized model displacement and grayscale color textures. 43
4.18 . . . 44

5.1 Diagram of potential users groups. The groups' area is not representative of their population. . . . 45
5.2 Graph of the targeted user groups and test subjects relatively expertise in the fields of 3D Modeling and Game Development (or similar). . . . 46
5.3 Adapted from Bangor, Kortum, and Miller 2009. Shows the prototype score in relation to different metrics. . . . 49
5.4 Automatic segmentation focused on color distance over different LODs using 27 clusters. . . . 54
5.5 Automatic segmentation focused on angular distance over different LODs using 27 clusters. . . . 54
5.6 Comparison between a segmentation using color distance,at the left, and angular distance, at the right, using 15 clusters. . . . 55
5.7 Comparison between a segmentation using color distance,at the left, and angular distance, at the right, using 26 clusters. . . . 55

6.1 Example of geometry distortion. The area surrounding the painted area is shrinked reducing the geodesic distance of near faces. . . . 59

List of Tables

5.1 Frequency of found usability problems 52
5.2 Usability Problems Solutions 53

LIST OF TABLES

Abbreviations

3D	Three-Dimensional
API	Application Programming Interface
CAD	Computer-Aided Design
CPU	Central Processing Unit
FBX	FilmBoX
GPU	Graphics Processing Unit
GUI	Graphic User Interface
HCI	Human-Computer Interaction
HDM	Head Mounted Display
LOD	Level of Detail
MIDI	Musical Instrument Digital Interface
OWL	Web Ontology Language
PBR	Physical Based Rendering
RDF	Resource Description Framework
SOTAR	State Of The Art Review
SUS	System Usability Scale
UE4	Unreal Engine 4
UI	User Interface
VE	Virtual Environment
VR	Virtual Reality
VRML	Virtual Reality Modeling Language
WYSIWYG	What You See Is What You Get
X3D	eXtensible 3D Graphics
XML	eXtensible Markup Language

Chapter 1

Introduction

The goal of digitalized 3D models is to, as accurately, and automatically, as possible, capture the physical properties of real world objects, bypassing the need to manually model them. Typically, although this varies depending on the specific techniques employed, a point cloud model of an object or scene is created, which then can be converted into a 3D mesh. These models contain much of the details present in the real object, such as its texture and surface nuances. For this reason they have been used for movies, preservation of cultural heritage through virtual means, visually realistic video games and other virtual environments (Foster and Halbstein 2014, 7-15).

In the context of this dissertation Digitalized 3D Models refers to 3D meshes with assigned textures, such as the ones described in OBJ or FBX file formats, that are virtual copies of real objects and obtained using either photogrammetric or laser scanning techniques.

The increasing computational power and performance - less energy consumed per number of computations - mentioned in Denning and Lewis 2017 - have resulted in an increase in both the processing power of mobile devices and, consequently, accessibility of technologies that allow for user immersion in VR environments such as the ones presented in Martín-Gutiérrez et al. 2017 (e.g. Samsung Gear). GPUs also have been improving their general purpose and rendering capabilities as mentioned in Nickolls and Dally 2010, allowing for the creation of visual high-fidelity environments (e.g. DICE 2016). Finally, digitalization technology, as mentioned in Foster and Halbstein 2014, 10–14, has also become more accessible. All these advancements make the usage of digitalized models more accessible and pertinent.

1.1 Motivation

Even though digitalized models can be used in real time interactive virtual environments to achieve a high degree of realism from a visual aesthetic standpoint they do not capture information regarding the original model segmentation (i.e. the different individual parts that compose it), physical properties (e.g. stiffness) or how its individual parts relate to each other (e.g. a door knob is connected to the latch). They also have specific limitations due to the way they are conceived.

Figure 1.1: Photogrametric 3D model assets used in Start Wars Battle Front. Taken from DICE 2016

Additionally, they may also impose more work in the development of the virtual environment in which they are used. If the environment also includes non digitalized models these need to be quite detailed in order to blend in. In this case, more technical apt 3D modelers, are also required, as realism requires a combination of strong aesthetic and technical abilities (Foster and Halbstein 2014, 17–20).

Consequently, due to the aforementioned reasons, digitalized models usually lack realistic interactivity, which is important, specially in VR environment (as will be seen in 2.1). And, when they are interactive, they usually require more work than traditional models.

With an increasing accessibility in digitalization and VR technology, new, user friendly, integration tools that match the same level of accessibility are needed for an overall unburdensome integration workflow of these models in VR environments.

1.2 Goals

The broader goal of this project is to improve upon the traditional workflow integration approaches of digitalized models into VE environments, which potentially could contribute their immersiveness by making the integration more accessible (fast, easy, cost efficient) and consequently allowing the integration of more or higher quality digitalized models.

Additionally, this workflow should be VR oriented and focus on interactive objects with a similar interface to GUIs, such as remote controls, gamepads, keyboards, and the like.

The concise way in which this project proposes to improve upon existing workflows is trough the specification of an alternative novel workflow that tries to automate some of the integration tasks and reduce the need of switching environments during the model integration.

To test the proposed workflow a prototype is implemented, with Unity and Blender, and used to integrate the 3D model of an Oxygen 8 M-Audio MIDI keyboard.

Figure 1.2: Oxygen 8 M-Audio MIDI keyboard

Whether the implemented solution provides an improvement over a traditional workflow or not is evaluated with respect to two different types of users. Users who have little or no experience with integration of these models using a traditional approach, 3D modeling oriented, and users who are experienced.

The final goal is to make the integration of 3D digitalized models into VR environments easier and faster for experienced users or doable by unexperienced users, or, in case this is not possible, expose the reasons why.

Through usability tests, the proposed workflow and prototype are evaluated and compared with a traditional workflow. The results obtained should indicate if the specified workflow and its implementation offer, or have the potential to offer, any significant advantages over traditional approaches and if they make the integration process accessible for users without modeling experience, which typically can't use a traditional approach.

The following list presents, in an organized manner, the main objectives. Firstly, in point one, are listed objectives related to the workflow specification, whose qualities should be present in the implemented prototype. Then, points two, three and four are aimed at uncovering whether the workflow as the potential to be applied or fails to improve the integration process.

1. Specify a new, more integrated, workflow for VE integration with the following characteristics:
    1.1. Abstract the user from modeling, and other specialized tasks
    1.2. Automate common VE tasks
    1.3. Minimize context switching
    1.4. Make the integration process clearer
    1.5. Oriented towards interactive objects
    1.6. Allow to add VR interactivity through different VR interaction paradigms

1.7. Produce results with enough quality as not to compromise the integration

2. Integrate a model using the proposed workflow to understand if the workflow allows to completely integrate models without using specialized software and uncover possible limitations. (related to points 1.1, 1.2, 1.3 and 1.7)

3. Examine if the proposed workflow makes the integration process accessible to non modelers. (from a usability standpoint)

4. Examine if the proposed solution improves or can complement the integration process for modelers.

4.1. Understand what are the advantages and disadvantages of the proposed workflow

1.3 Document Structure

After this introductory chapter, this document has 5 more chapters: a state of the art review; the proposed workflow specification; implementation; validation; and conclusions and future work .

The state of the art is divided into the the Virtual Environments and Model Acquisition and Integration sections.

The Virtual Reality Environments section (2.1) focuses on exposing key concepts in understanding the relationship between VR environments and dynamic digitalized models.

The Model Acquisition and Integration section (2.2) gives a brief overview regarding the process of digitalization, focusing on the possible limitations and high level procedures of photogrammetric and laser scanning approaches to better understand their impact on the overall integration workflow.

The Workflow Specification chapter 3 presents the conceptual workflow proposal and the core features it should, or not, implement. This chapter also complements the state of the art as it exposes some of the many implementation possibilities in a critical light, arguing what are the advantages and disadvantages of some.

The Implementation chapter 4 exposes the concrete implementation of the workflow. This also entails the capturing and refinement of the MIDI keyboard model, but more importantly exposes the final state of the implemented integrated environment prototype and explains its implementation details.

The Validation chapter 5 describes the targeted user groups for the usability tests, the tests protocol and what data was collected during these, and then presents the main findings of these tests. It also presents the results obtained with the adapted segmentation algorithm.

The final chapter, Conclusions and Future Work 6, summarizes this projects findings and contributions, and discusses possible future improvements or related works.

Chapter 2

State of the Art

The chapter is divided into 2 sections. The purpose of this chapter is to provide an overview of important concepts and informations of areas of study that are related with the subject of this document.

In the Virtual Reality Environments section (2.1) are introduced key concepts in literature. By adapting the Zeltzer 1992 API-Cube is established an intuitive conceptual model to understand how an isolated dynamic digitalized model influence immersion. This model, implicitly, also suggests an intuitive delimitation to the term "dynamic", used in this documents title, and how it relates to user immersion. In the Interaction subsection (2.1.2), the concept of interaction is expanded. Finally, the 3D Representation subsection (2.1.3) explains how a virtual environment is represented depending on its complexity.

In the Model Acquisition and Integration section(2.2) is explained how photogrammetric and laser scanning techniques differ in terms of results and limitations, thus providing a basic understanding of how they can influence the integration workflow. After this the photogrammetric digitalization workflow commonly used in game development contexts is summarized. Note that this workflow only allows to obtain a visual representation of an object or scene, and does not add any dynamism to it (besides, possibly, dynamic illumination readiness).

2.1 Virtual Reality Environments

Virtual Environments, or using a more specific term VR (Virtual Reality) environments[1], are defined in Milgram and Kishino 1994 as the polar opposite of a real environment, i.e. a scene composed only by virtual (non-real) elements. Other types of environments, that are neither purely "real" or "virtual" such as augmented reality or augmented virtuality environments[2], are defined as "Mixed Reality" (MR) environments.

[1] Which according to Milgram and Kishino 1994 is the common nomenclature used for a Virtual Environment, i.e. completely synthetic environment, in which the user is highly immersed, hence "Reality". This is the definition that the term will connote throughout this document. However, the term may be used with different meaning in related literature.

[2] Using the same terminology referenced in Milgram and Kishino 1994.

The elements present in virtual environments may be copies of real elements, as is the case with digitalized models, however these models can't completely reflect the original state of the real object or scene from which they were obtained, they are "lookalike" virtual objects and not a real time video rendering of the real objects. For this reason, digitalized models and virtual environments are not exclusive terms. For example, a 3D simulation might possess a VR environment made of digitalized models only. The details of the terminology could be explored further, one could ask if models that imitate the state of their real counterparts to a certain degree could be considered a form of mixed reality, however, these are questions that fall outside the scope of the typical applications of digitalized models and consequently won't be explored in this document.

Zeltzer 1992 presents a taxonomy that identifies the main components of VR environments, which are:

- **Autonomy**.
  Defined as "... a qualitative measure of the ability of a computational model to act and react to simulated events and stimuli, ranging from 0 for the passive geometric model to 1 for the most sophisticated, physically based virtual agent". E.g. soft bodies and other models properties in physical simulation or AI controlled agents.

- **Interaction**.
  The degree to which the user can interact with the virtual environment. E.g. opening a door in VR. In this example, whether the door is animated or has a physical simulated force is a matter of autonomy. The variable that the user can change, which is related to the interaction component, is the state of the door which is continuous in the second case.

- **Presence**.
  Quantifies and qualifies sensory input and output. E.g. stereo audio channels allows for a richer experience than a single audio mono channel.

2.1.1 Immersion

A key aspect of VR environments is the user's immersion. The definition of immersion varies depending on the author and context. The components defined by Zeltzer 1992, although proposed as a way of classifying types of virtual environments, are very useful to define immersion in the context of this document. Ultimately, excluding additional factors not present in the model, a VR environment is more immerse the bigger its euclidean distance is from the origin point on the AIP-Cube[3].

An interesting characteristic of these components is that they can be easily be adapted to evaluate how realistic a digitalized model, when already integrated on a VR environment, is. The following examples illustrate how each of the components can be used in this context:

[3] 3D Coordinate system that measures the Autonomy, Interaction and Presence components along its axis. Zeltzer 1992

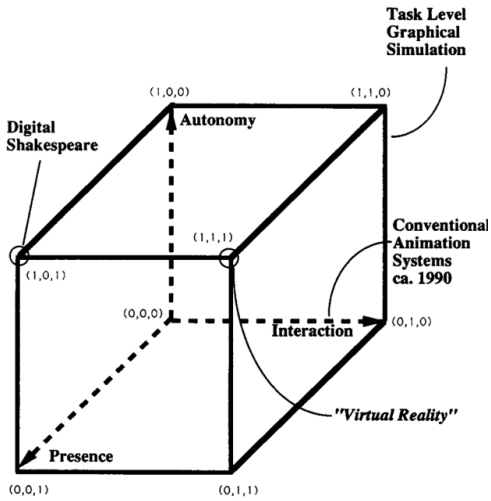


Figure 2.1: AIP-Cube from Zeltzer 1992

- Autonomy - The usage of physical properties such as mass that change the model dynamic behavior.
- Interaction - The ability to interact with mechanic components such as buttons or levers.
- Presence - The quality of the mesh and textures, and possible sound.

This is very intuitive, as when used this way, the model is evaluating a subset of the environment instead of its totality. At max granularity, an "adapted AIP-Cube" has the potential to be a tool for evaluating the degree of immersion provided by a single model, without taking into account the influence of the surrounding VR environment.

More recent proposals, such as Ermi and Mäyrä 2005 and Gordon 2007, which are focused on video games and not particularly on VR only, define immersion as being the result of many additional factors and distinguish different types of immersion. One of these is typically related to the sensory stimuli, "sensory immersion" in Ermi and Mäyrä 2005 and "spatial involvement" in Gordon 2007, and are very similar to the presence component defined in Zeltzer 1992. The interaction and autonomy can also be found in these and other components. Ultimately, such model proposals are not of much relevance, as the AIP-Cube components adapted to digitalized models provides a more pertinent, simpler and more intuitive, taxonomy to characterize the models. When using a more complete definition of immersion details such as the perceived quality of voice acting of virtual entities or the aesthetic appeal of a scene can be considered and, in context of characterizing digitalized models, they provide unneeded complexity. The only additional factors related to immersion that are relevant to consider are the ones that relate to how the user interacts with the environment, this will be further explored in section 2.1.2. Surveys, as the one present in Huiberts 2010[4], show, indirectly, that there is some perceived value regarding the usage (and correct

[4]Some of the answers presented in the appendix 8 of Huiberts 2010 are directly related to components defined in Zeltzer 1992.

usage) of these components, which serve as evidence to legitimize the theoretical models and the relevance of related projects.

In a generic sense immersion is a quality that is usually considered a positive aspect. This last statement is backed-up by surveys, such as the one presented in the previous paragraph. However, when dealing with VR, some restrictions to the sense of immersion might be important to allow a safe experience. Safety is both important in the physical sense, e.g. the used hardware should not strain or compromise the user posture, and in the psychological sense, e.g. the experience may be too much realistic and trigger panic or stress responses.

2.1.2 Interaction

As previously mentioned, interaction can be considered one of the components used to characterize a VR environment. However, that definition does not entail many of the important factors that influence the immersion, that easily come to mind when mentioning the term in a broader, more generic, context. To properly understand the concept of interaction and all its different connotations, the following points need to be taken into account:

- Interaction techniques available to the user in the VR environment context. E.g. manipulation of objects orientation or navigation of a scene techniques. Some can be specific to specific devices. A recent SOTAR of these techniques is Jankowski and Hachet 2013, which only focus on interactions available in general-purpose hardware and not specific devices such as HMDs. It also divides interaction techniques in 3D space into three components, in conformity with previous literature, which are: Navigation; Selection and Manipulation; and System Control.

- Manipulation Device. The hardware from which user input is received to apply interactions. It may also provide some sort of output. May be a mouse, tablet, haptic glove, video capturing devices or HMD that sense head movement or track eye movement.

- Algorithmic aspects. The specific algorithms used to process a device input data. In Rautaray and Agrawal 2012, 21–22, for example, are listed the advantages and disadvantages of using certain vision based hand gesture recognition techniques from a low level perspective such as execution time and robustness. These properties affect the interaction by influencing, for example, its responsiveness and precision.

All these points are relevant to discuss immersion. Is through them, that it is possible to discern the qualities of a VR environment interaction in its broader sense, qualities such as:

- Precision. E.g. High precision tracking allows the user to interact within more constricted spaces.

- Responsiveness. E.g. If the user input lag is too high it may cause difficulties interacting with the environment.

- Scope of Applicability. Some techniques or devices work better under specific conditions (e.g. well illuminated environments or big open spaces).
- Skeuomorphic qualities. E.g. A driving simulator uses input devices that mimic the vehicles components and the input data they provide is used in the virtual environment to mimic the consequences that an action would have on a real car.

With particular focus on the last point, a intuitive way to classify the interaction is through the use of metaphors, in Jerald, LaViola, and Marks 2017 the following metaphors for classification of VR interactions are enumerated:

- Grasping metaphors: hand based (e.g. Jacobs and Froehlich 2011) and finger based techniques (e.g. Talvas, Marchal, and Lecuyer 2013).
- Pointing metaphors: vector-based (e.g. Pierce et al. 1997) and volume-based techniques.
- Indirect metaphors: indirect control-space (e.g. Simeone 2016) and proxy techniques (e.g. Pierce, Stearns, and Pausch 1999).
- Bimanual metaphors: symmetric and asymmetric techniques (e.g. Cho and Wartell 2015).
- Hybrid metaphors: e.g. HOMER (Bowman and Hodges 1997).

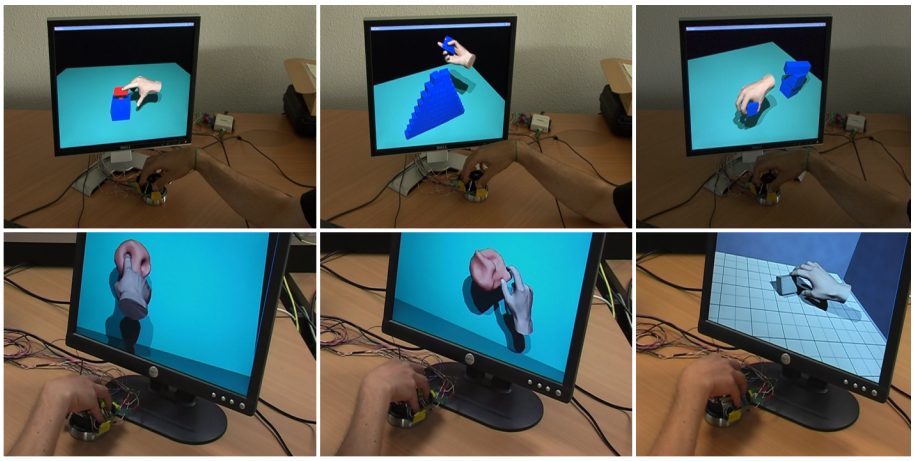


Figure 2.2: A grasp based interaction device prototype. Extracted from Kry et al. 2008.

2.1.3 3D Representation

At its simplest, a VR environment, or any virtual environment, is represented only by the 3D structure, i.e. geometry and texture, of its elements. 3D structures can be stored in formats such as OBJ, which are commonly used in modeling software.

The previous definition, even for a simple VR environment is over simplified. Sometimes low-level information can be stored in very different formats. For example, terrains can be stored as

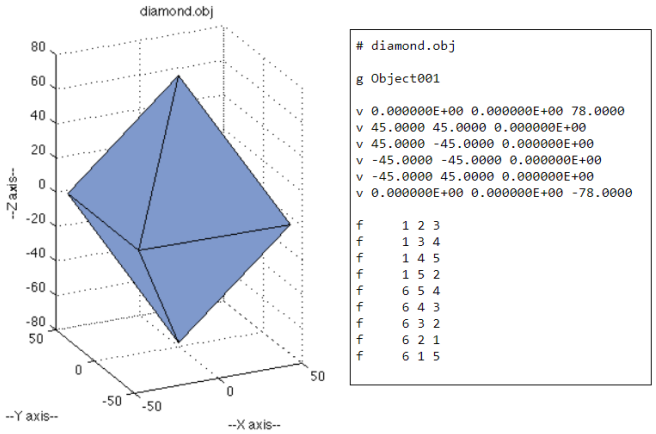


Figure 2.3: Example of a simple 3d model represented in the OBJ format. At the left is presented the model defined by the OBJ at the right. The presented OBJ only specifies the vertices and theirs indices. Image and source OBJ from "OBJ Files - A 3D Object Format" 2018.

heightmaps or voxels. However, for the sake of simplicity, since this document focus is on digitalized models whose geometry is encoded as a mesh in order to be integrated on VR environment, such formats won't be addressed in this document.

As the complexity of the environment increases additional information is required. Referring to the API-Cube as a simplified way to evaluate immersion, it is needed to specify the properties of its elements so that they can be **autonomous** and **interactable**. Formats such as VRML and X3D can store this kind of features, such as events and behaviors through scripts or how the models are rendered (e.g. specify backface culling for solids) . In the case of video games or other commercial interactive applications, such formats are usually replaced with a custom in-house format, specifically designed to be used by a specific editor or game engine.

However, the already mentioned formats may not be enough. Sometimes virtual scenes may be too complex to be modeled in useful time or by a person that doesn't have experience with 3D modeling or 3D graphics. There may also exist a need to represent the environment in a text format, include annotations or add some high-level meta data that dictates relationships between its elements. In such cases, a semantic, high-level, representation of the environment is required. Such representation may be used to: generate a scene procedurally based on a high-level descriptions; enrich the environment by establishing relationships, constraints, and other high-level information that can be used in a variety of ways (e.g. simulations); used only to retrieve information about the 3d environment[5]. The OWL is an example of a format that can be used for high-level representations, as it allows to define classes and properties, instantiate them, and reason over these abstractions and instances.

This last type of representation is commonly referred in literature as ontology-based representation. Despite its usefulness in representing high-level concepts, it also has been used for

[5]This high level information may not be "baked", i.e. instantiated, and, in such case, needs to be extracted from lower level information.

low-level modeling. Flotyński and Walczak 2017 is a state of the art review on this type of representation. In it are mentioned examples of both usages in academic literature. Regarding the level of abstraction, it proposes the following classifications:

- Concrete Representation. The lowest possible level of representation. The concepts are comparable to the ones used in 3d content, such a geometry, texture and similar structural information. Can also incorporate VR/AR environment related concepts, similarly to X3D.
- Conceptual Representation, which does not contain concepts present in concrete representations and focus on higher level concepts of a specific knowledge domain.
- Hybrid Representation, which, as the name suggests, incorporates both concrete representation related concepts and other specific knowledge domain concepts.

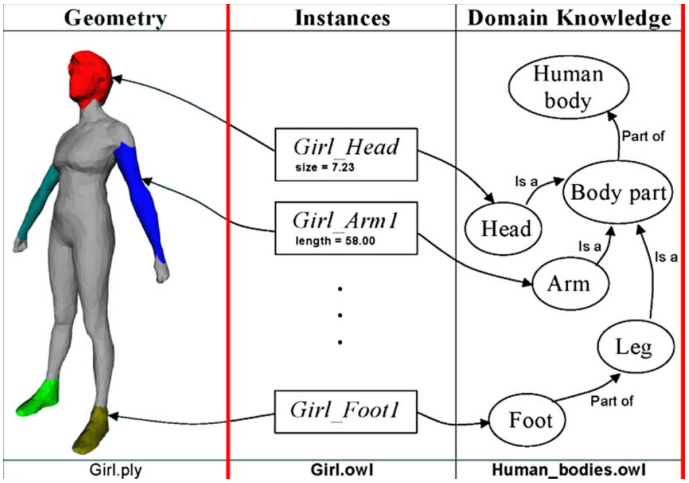


Figure 2.4: Semantic annotation of automatically segmented model parts using OWL. Cutout from Attene et al. 2009.

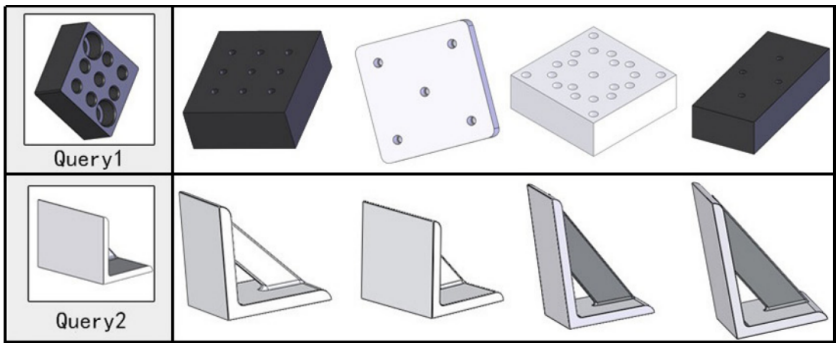


Figure 2.5: Retrieval of semantically similar 3d shapes, an example application of semantic representation in CAD context. Cutout from Qin et al. 2016.

2.2 Model Digitalization and Integration

In this section the digitalization processes that can be used to obtain an object's geometry will be explained from a practical standpoint, focusing on the implications of using certain digitalization techniques.

Then, the digitalization workflow, consisting of all the additional steps for the integration of a non interactive model in a VE besides the acquisition of the model's geometry is explained.

2.2.1 Digitalization methods and limitations

The acquisition of a model can be achieve through several means, such as **laser scanning** or **photogrammetry**.The first works by measuring the direction and travel time of laser pulses that are fired at a target object. The later uses images as input. Usually, multiple photographs of the target object in different perspectives are used, and, by locating common points it is possible to estimate their position in a 3D space.

It should be noted that both approaches are not restricted to a single algorithm or equipment choice. Regarding photogrammetric algorithms, they are varied and some require additional user input besides visual sources (images or video). The exposition in this section only focus on explaining common limitations which are relevant in the model integration workflow, and will not mention specific algorithms or technical details in the procedures.

In academic works, laser scanning approaches, specially for large objects (or scenes), are often referred as being more expensive than photogrammetric approaches, without offering significant advantages over them. These points can be found in diverse works that compare the two approaches, such as Radosevic 2010, Skarlatos and Kiparissi 2012, and Kolecka 2011.

The photogrammetric approach is also considerably more flexible due to:

- The range of use. As stated in ADAM Technology 2008, "Photogrammetry is very flexible when it comes to range. The same technique can be used on any scale", changing the lens allows the captures at different distances without any significant impact on the results.

- The capturing device being independent of the software/techniques used. There is a high diversity of cameras available with different characteristics and price ranges.

- The diversity of algorithmic solutions. Multiple solutions have been proposed and developed, some of them have been developed for the capturing of a specific type of objects/scenes. For example, the Manhattan-world Stereo Furukawa et al. 2009 focuses on the capture of highly structures architectural scenes. Independently of the capturing hardware, users can select the software or algorithms that better fit their needs.

- Semi-automatic solutions. Some solutions, such as the ones available in PhotoModeler (see "PhotoModeler How it works" 2017), require additional user input besides raw images. These methods allow to fine tune the generated model or avoid some limitations of completely automatic solutions.

Both methods have difficulties dealing with non-lambertian[6] surfaces and thin structures.

Photogrametric approaches have difficulty dealing with homogeneous surfaces, without a texture or diffuse illumination. It is also required that the photos are taken in a way that the target model is focused and that the background doesn't blend with it. After the capture phase, the produced module is typically not scaled with respect to some unit measure, and a reference is needed for this. (Furukawa and Hernández 2015)

On the other hand, laser scanning is simpler to use and produces results faster. Depending on the used hardware, it has the potential to produce more precise geometry as can be seen in the results and comparisons in Radosevic 2010.

Typically, laser scanning requires the usage of additional sensors in order to capture the model's texture. When capturing it with this method or with photogrammetry, the illumination may influence the final result. This is specially true for the photogrammetric approach and sometimes the only way to remove illumination artifacts (e.g. shadows, highlights) is by editing the texture after the model is captured.

2.2.2 Digitalization Workflow

Depending on the target object to be digitalized and on the used equipment there are documented procedures that should be followed in order to obtain good results. These procedures can be relative to the acquisition process or to post model capture refinements.

The photogrammetric model integration workflow in virtual environments has been documented in game development contexts. Noticeably, the game engines UE4 and Unity have made public documentation regarding the integration of these models, Jover 2017 and Oh 2015 respectively. It also possible to find such information in game development and similar forums and blogs, such as Azzam 2016, which often are good places to learn about practical guidelines and details regarding the models' integration.

In the aforementioned documentation, the workflow can be divided into the following.

1. Capture. How to prepare the scene, what tools to use and how to correctly use them.

2. Image processing. Prepare images to be used. Change image format, apply color corrections and similar operations.

3. Mesh creation. Create the model mesh, bake its normals, possibly simplify its geometry and prepare UV layout.

4. Material Creation. Creation of the different textures (e.g. bump map from high frequency geometry) and assignment of the material properties (e.g. reflectiveness). Important step in the creation of realistic environments, specially when using PBR. Can be somewhat automated using tools such as Substance B2M[7].

[6] Which surface illumination is not uniform, as in the case with most plastic and metallic materials. On the opposite spectrum, materials such as wood are uniformly illuminated.

[7] See "Substance B2M - IMAGE-TO-MATERIAL GENERATOR" 2018.

5. Lightning removal. Removal of shadows and ambient illumination from the model's texture. Only needed if the original scene lightning is not meant to be used, as in the case in dynamically illuminated scenes for example. Sometimes can be solved by avoiding direct lightning sources in the capture environment in the capture phase.

6. Tileable material. Adjust the texture so that it can be repeated over and over. Mainly used on environments or generic materials which the complete coverage of the mesh with a single texture would be prohibitive.

Although this might be considered as a good workflow, the mentioned documents are not comprehensive, and only give developers practical guidelines that allow to obtain good results in most cases. Additional procedures and more detailed knowledge are sometimes needed, such as:

- Mesh reconstruction[8]. Fix missing parts in the digitalized model or structural problems that arise from scanning a damaged object. (Stavrou et al. 2006)

- Texture patching. Textures may contain holes, i.e. missing data, that needs to be patched. Or, to produce the desired texture, different elements from different textures may need to be combined seamlessly.(Zalesny, Maur, and Van Gool 2001; Efros and Freeman 2001; Wei et al. 2008; Celaya-Padilla et al. 2012; Bhatia 2014; Raad and Galerne 2017)

- Mesh retopology. Recreate the original mesh or parts of it, organizing the mesh in a different way. This can be done for different reasons, such as: make used animations consistent and believable; avoid loss of detail on prominent features when simplifying the mesh; allow usage of dynamic LOD. (Pietroni, Tarini, and Cignoni 2010; Foster and Halbstein 2014)

- Texture Mapping. If the original model is reconstructed from scratch for mesh retopology the mapping of the texture on the model is lost and the it needs to be baked onto the new mesh with minimal distortion and information loss. If a tileable material is used then it should be mapped onto the mesh correctly, regarding some criteria (e.g. texture continuity on mesh corners). (Turk 2001; Wei and Levoy 2001; Singh and Namboodiri 2012; Zhang et al. 2003)

The usage of such procedures varies depending on the nature of the models, experience of the user and available tools. For example, retopology can be applied manually by reconstructing the model in a modeling software or automatically via an algorithm usually implemented in similar software such as MeshLab[9].

There might be some procedures used by field professionals that do not fall within the scope of document. The reason for this is the lack of information regarding these techniques, which in some cases are kept in secrecy for competitive or legal motives.

The workflow, when dealing with laser scanners is more streamlined due to its capture process being simpler. After the capture phase, it is similar to the photogrammetric workflow.

[8]The term may sometimes be used as a synonym to retopology. This is not the case here.
[9]see "MeshLab - Features" 2018; Pietroni, Tarini, and Cignoni 2010.

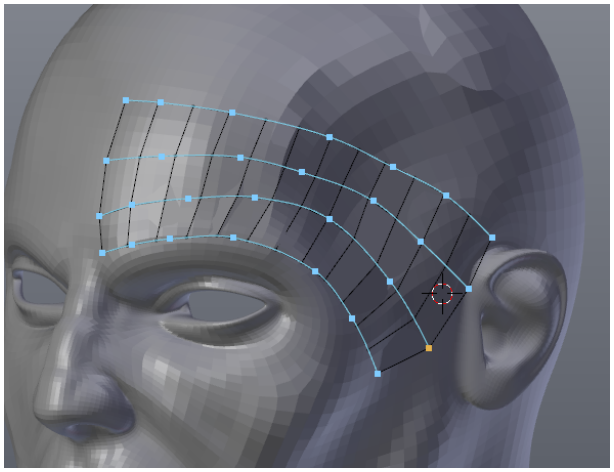


Figure 2.6: Manual retopology of a model using the Mira "Mira Tools" 2018 plugin for Blender "Blender" 2018. The new mesh, highlighted in the image, is snapped to an existing mesh surface.

Figure 2.7: Composite textures (texture patching related). The right picture shows a virtual reforestation of the scene on the left picture. Taken from Zalesny, Maur, and Van Gool 2001

Chapter 3

Workflow Specification

This chapter aims to present the proposed workflow on an abstract level and clarify core concepts related to this.

To devise a more streamlined workflow an environment is needed. This environment can use both higher level functionalities, required to build an integrated environment, such as the ones present in game engines, and low level functionalities, such as the ones implemented in modeling software. Such functionalities can be embedded in the used environment natively, through libraries, or through external software or services.

The environment should then combine these two types of functionalities creating a hybrid integrated workflow that should automate some of the steps needed to make a model interactive and remove the need of context switching. Additionally, a WYSIWYG environment without the context switching allows the integration steps and final result to be clearer. An example of WYSIWYG usage in software related to 3D model preparation are the Blender's modifiers, which result can be previewed before they are applied to the model. Figure 3.1 illustrates this: in a) the model without modifiers is shown; in b) a modifier is added and its result is previewed; on c) a stack of modifiers is added and the final result is previewed.

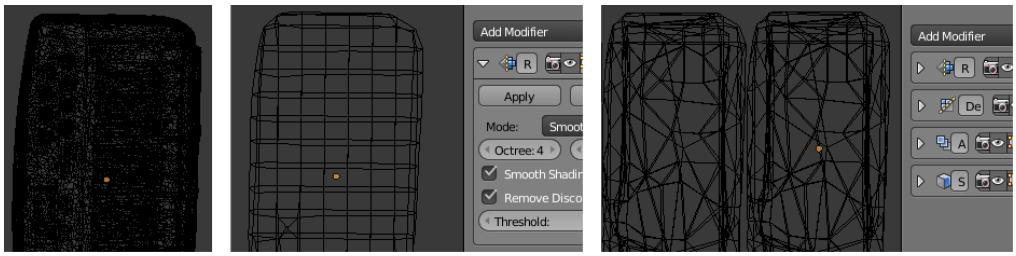


Figure 3.1: Blender's modifiers preview.

In the proposed workflow, ideally, WYSIWYG should transverse all the step of the integration process, thus making it as clear as possible from start to finish.

The following diagram presents a conceptual diagram of the proposed workflow's tasks flow and the different needed environments in relationship to these tasks. The main differences between the more traditional workflows are the planning and automation related activities. The most relevant tasks here presented will be detailed further in the following sections of this chapter.

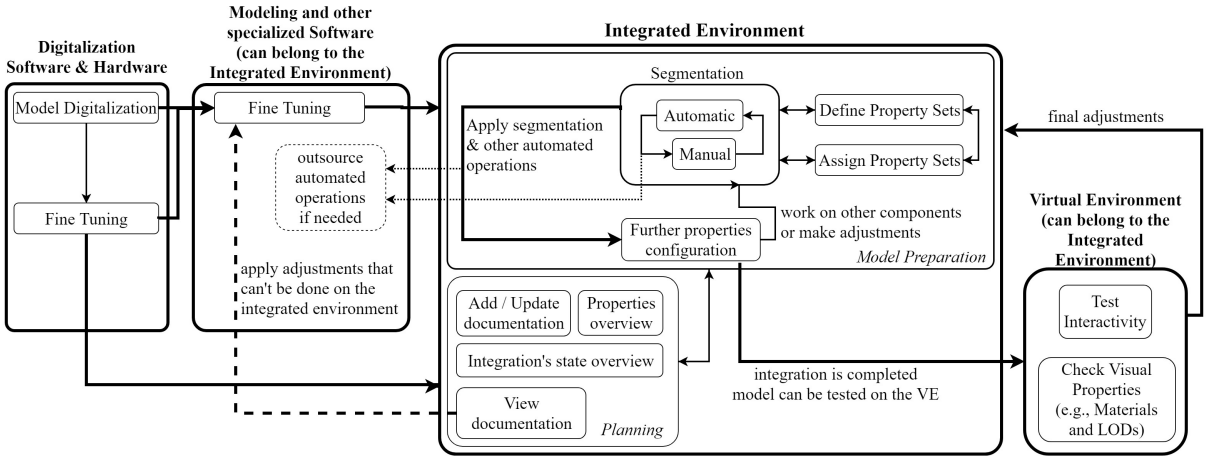


Figure 3.2: Conceptual Diagram of the proposed workflow.

Note that the proposed workflow is missing some key tasks related to the model's integration. For example, material preparation related tasks could, by adding more granularity in the diagram, be presented in the integrated environment or in the environment before it. Because it is not clear where these tasks would fit better, since the scope of the project doesn't allow for an extensive exploration of workflow variations, their order in the workflow was left unspecified.

3.1 Model Digitalization

Unequivocally, the first step in the model integration workflow is the digitalization of the model. This can be done through different techniques as was mentioned in the 2.2.

Because an interactive object is usually composed of different components binded together, if they are digitalized together, which sometimes is unavoidable, it is necessary to separate them later. To avoid this, if possible, the different model's individual component should be digitalized separately.

After obtaining the digitalized model, additional procedures, such as automatic removal of unwanted geometry or other minimal model preparation procedures such as delighting and retopology are not specified in this proposal. There are some tools used that allow some automation in these departments but their application is not here considered. This creates a small gap in the conceptually ideal workflow by isolating the initial preparation related procedures from the rest of the integration process.

3.2 Automatic Segmentation

Unlike a 3D manually produced mesh, a digitalized model mesh is unified and does not discriminate different components, which is needed in order to add interactivity to the model. Therefore, automatic segmentation of the model has the potential to speedup the model preparation process considerably, by allowing the user to skip the process of modeling or separating the different components. However, automatic segmentation can't infer missing geometry. So, completely or partially occluded components may still need to be manually modeled.

Additionally, independently of the qualities of the used algorithm, there is an additional problem common to all automatic segmentation algorithms. This problem is their agnosticism regarding following points:

- **The original object components physical properties.**
  E.g., a model might have bumps that are very similar to buttons for aesthetic purposes which are not intended to be segmented. As will be explained in examples further below, this can be solve by a mixed, not completely automated, approach.

- **The lack of information about the user's intention.**
  Just because a model contains some interactive component that doesn't mean that the component is relevant for the context in which the model will be used on the VE. In such case, it might be preferable to leave the component as is, without applying any form of segmentation. E.g. a digitalized room contains many potentially interactive objects, such as a computer or a chalk board, but in the context in which it will be applied, for instance a fire simulation, its only pertinent to attribute interactivity to a door or fire extinguisher.

From the examples above it becomes clearer that a fully automated segmentation can sometimes produce unusable results despite working "correctly". Plus, even when the results are usable, it is also possible that a fully automatic segmentation is inefficient due to the time it takes being mostly spent on undesirable segmentations or due to the visual noise that these segmentations produce by flooding the user attention with irrelevant data. Therefore, from all the mentioned limitations and possible problems that can arise from them, we can conclude that a model integration workflow can't be complete with automatic segmentation alone. At best the segmentation needs to be semi-automated and possibly iterative.

One or more manual, complementary, procedures[1] are needed. The automatic segmentation can be integrated with such procedures differently, depending on the desired workflow and the characteristics of both the segmentation procedure and the manual procedure. For example, if an algorithm chooses the number of segments automatically and has a tendency to divide the model in more segments than required, a cleanup procedure can be done afterwards to merge different segments into one or it can be modified receive areas of interest given which will evaluate individually. However, if the algorithm required the user input to decide the number of segments,

[1] Such procedures may not necessarily be a form of segmentation, despite this being the approach chosen for this project's implementation.

the latest procedure would require that for each area of interest the user specified the desired number of segments. Such additional input could possibly make the workflow cumbersome but also more flexible. Both algorithms, and possibly more, could be available to the user, and this would add complexity to the workflow.

Note that the selection of areas of interest mentioned above does not necessarily imply a manual segmentation, in the sense that the user is not really segmenting the model but only providing complementary data to the algorithm. Thus the use of the more generic term procedure, which might not be in-itself a segmentation procedure.

Perhaps, a possibly more interesting solution to the agnosticism problem is the graph-cut approach presented in Brown 2008. This segmentation requires that the user sketch over what is desired to be segmented, in a similar fashion, interaction wise, to the GrabCut image segmentation method. This differs from the previous presented examples as in this case the user input is obligatory, being data that is required by the algorithm.

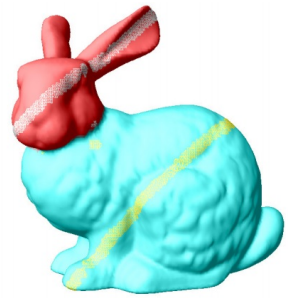


Figure 3.3: GrabCut approach. Taken from Brown 2008. The white and yellow stripes and dot were user inputed and the red and blue parts are the resulting segmentation.

3.3 Manual Segmentation

Manual segmentation is the procedure that allows the user to select components of interest in the model and cleanup automatic segmentation mistakes. The term "manual" can be misleading. The manual segmentation can, similarly to automatic segmentation, automate some of the segmentation behind the curtains as long as the user obtains instant feedback and the results obtained are very clear. The idea behind the naming is that, unlike automatic segmentation, it requires continuous user engagement thus requiring an intuitive tool that provides instant feedback. In this sense, unlike the black box like behavior of an automatic segmentation, manual segmentation can be characterized by its WYSIWYG related qualities.

3.3.1 Segmentation by Painting

Segmentation by painting mimics the the process of a painting software but instead of working on an image only takes into account a 3D model. This can be done in two ways.

One is by associating colors to the vertexes or faces of the model, akin to Zbrush's polypainting functionality. The other is by using a UV mapped texture, present in most 3D modeling softwares. This approach can create visual artefact's and slow the painting process and, as will be explained in the implementation section 4.3.4.2, it is not ideal for segmentation. It also required that the model contains its vertices's UV coordinates data.

The painting approach can also used as a way to provide input for a semi-automatic segmentation algorithm.

3.3.1.1 Painting Styles

Independently of whether the painting uses a texture or not, the way the user interacts using the a painting brush can vary.

The painting can occur in 2D UV space or in a 3D model space. Most complete modeling softwares include both modes. Drawing attention again to the fact that the style of interaction is independently of whether a texture is used or not, note that even a 2D UV space can be used for poly painting, despite this not being common practice in modeling software as it is not the most useful feature in a modeling context and possibly requiring the usage of texture for preview purposes on a 2D interface anyway.

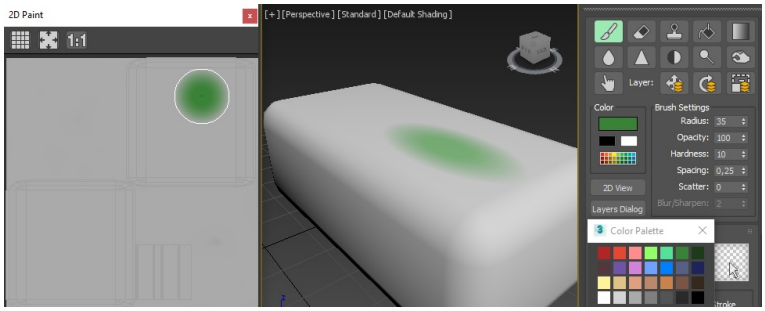


Figure 3.4: Autodesk 3d Max 2D and 3D UV texture painting tools.

The 3D model space brush paradigm can further be categorized into different types of painting interaction types. The following are some examples of painting paradigms:

- **Volumetric based painting.**
  Depicted in figure 3.5, this was the method implemented in the final prototype and considers a volume that encloses the geometry to be painted. It can be seen as a variant of the enclosing volumes segmentation explained in section 3.3.2 that provides instant feedback and uses a perfect 3D sphere, or similar shape fast when used in computations, over the model as a brush.

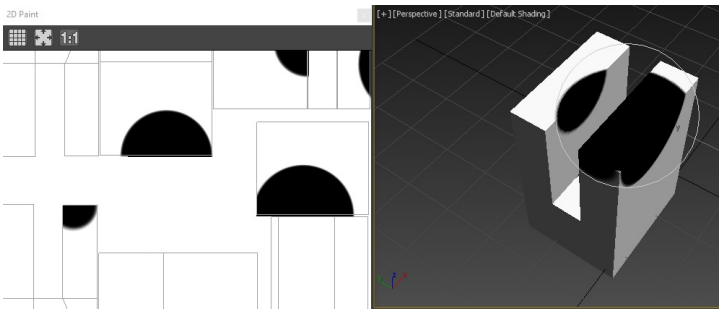


Figure 3.5: An example of texture based volumetric painting approach used by Autodesk 3D Max Software.

- **3D point to UV space brush.**
  This approach maps the point on the model pointed by the user to the model's UV coordinates and uses a 2D circular brush in this space. This approach can be problematic, particularly so in digitalized models, as painting directly in the UV space can result in geometry that far away from the pointed 3d position to be painted due to being mapped closely to it in the UV space. Consequently, on an usability level this approach is possibly the worst. See Fernández 2018 as an example of this approach.

- **"Surface" based painting.**
  This painting paradigm is similar to the 3D volumetric approach but also considers the geodesic distance to the 3D pointed point on the model. This may or not be desirable depending on what is to be segmented. The figure 3.6 tries to depict this type of segmentation[2].

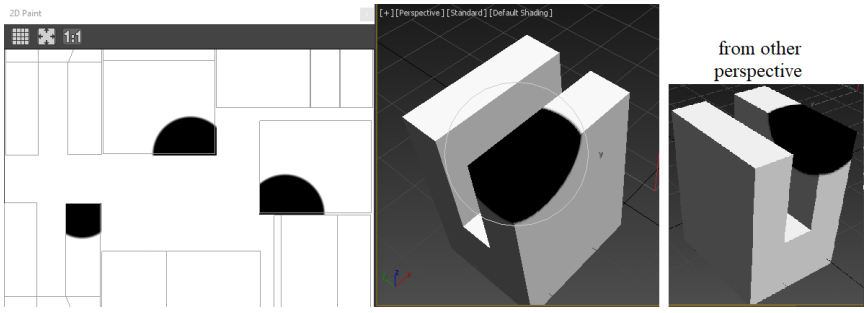


Figure 3.6: A "fake" depiction of a "surface" painting approach.

- **Perspective based Painting.**
  The utility of this approach in a segmentation context is very limited. It could be used to select specific surfaces areas, such as one of plane of component that closely resemble a cuboid, but outside of very specific cases does not offer any advantage over a volumetric approach. Figure 3.7 displays this type of approach. The image on the left shows the

[2]The actual segmentation depicted on the image is volumetric but it is presented with a specific perspective to try to convey the idea behind this approach.

painting brush targeting the model and painting it blue. The image on the right shows the same model viewed from behind, and it can be seen that the rear part of the painted elevation and its projected "shadow", relative to the previous perspective, remain unpainted.

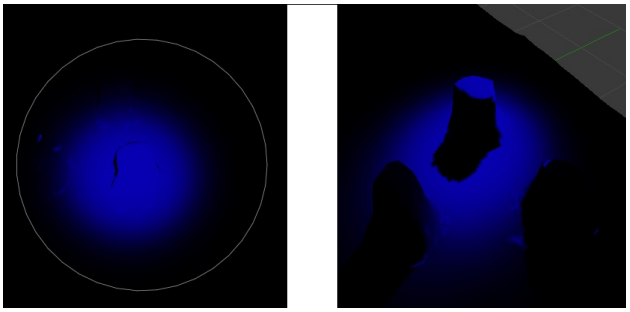


Figure 3.7: An example of a perspective painting approach used in Blender.

3.3.2 Enclosing Volumes Segmentation

Another way to segment the model is through the usage of user defined enclosing volumes. However the result of their segmentation and the way through which feedback is provided to the user, unlike the previous mentioned method, may vary greatly. It can also be combined with the previously mentioned approaches in different manners, also dependent on the concrete implementation.

One on the factors that affects this type of segmentation is what problem is being solved when applying the segmentation of a volume. Does the segmentation just selects all the geometry of the target model inside the enclosing volume or does it do something more, such as taking into account the normals direction to filter out unwanted geometry. If the user desires to segment components that correspond to elevations (bumps) int the geometry of the model, then, taking into account the normals can prevent the selection of other irrelevant geometry that is in the proximities of the target component, thus giving the user more leeway when defining the volume and consequently making the interaction easier in this specific case.

Another factor is how the used algorithms and implementation limit the solution. Due to the potential geometry density of digitalized models, in order to compute the result of the segmentation fast, similarly to the volumetric Brush, an octree of the model or similar techniques may be required to obtain an instant result. Also, a point in a polyhedron problem, a possible subproblem to solve depending on the approach used, is an old documented problem with many solutions already proposed (e.g, Lane, Magedson, and Rarick 1984; Liu et al. 2010; Li and Wang 2017) whose performance can vary greatly from each other. A solution that can instantly compute the segmentation allows the usage of an interface that can immediately preview the result while a slower solution could work in the background until the result is computed or, in case it is so slow that can't provide feedback in a timely manner, do not even allow to preview the result. In the last case scenario, the WYSIWYG qualities of the segmentation are compromised, and an integration with other manual segmentation approaches that provide instant feedback could result in less unintuitive and cumbersome segmentation process.

3.4 Interactivity and other components' properties

The interactive components of a model, e.g. buttons, levers, knobs, and so on, can be characterized by their properties.

These properties can have different levels of granularity.

At an atomic level they can be either related to the interactivity of a component or not. Some examples of interactivity related properties are: the resistance of a lever to the applied force; the angle of rotation of a door knob; the weight of ball that can be thrown. Some examples of properties nor related to the component's interactivity are: its rendering material; levels of detail; ab enclosing volume and its position on the model's hierarchy, used for rendering acceleration purposes.

At a less granular level properties can define multiple smaller properties. For example:

A property named physical material can define the rendering material, and the density of the component which can indicate whether the component floats or be used as a meta property used to calculate the component's weight.

Another property named spring back knob could define the spring physical properties of a door knob, its angles of movement, and triggers for when the knob is pulled down or released.

Some of these properties (but not the properties' values, though sometimes this might also be useful) should be defined before the components are segmented.

The reasons for this are:

- **Some automations can be applied in a more streamline fashion if done during or immediately after applying a segmentation.**
  E.g. the previously mentioned density property can be used to calculate the weight of the component while applying the segmentation.

- **The properties can work as contracts for behaviors whose implementation might vary.**
  E.g., a property defines that a component is a button but the concrete implementation of how it detects whether its being pressed may vary. The same is true for post segmentation mesh retopology operations and other geometry related operations.

- **Document intention.**
  E.g. , a certain mechanical contraption controls all the roller shades and light intensity in a room. Even before this contraption is properly segmented, might be useful to connect it to the roller shades and lightning so that its intended functionality is not forgotten later.

3.4.1 Automation

Ideally, to simplify the integration process, the components' properties' values[3] should be attributed automatically. Similarly, geometry, material and texture modifications, can, in some cases, be automated.

[3] These can be a numeric values, vectors, textures, enumerators, or any other type of data.

A few of the properties that can be configured automatically are:

1. Topology
2. Levels of detail
3. Material/Texture (e.g. creation of normal maps)
4. Basic physical properties. (e.g. weight)
5. Physical components (e.g. hinge rotation axis)

Some automations are best executed during or immediately after segmenting a component to avoid breaking the integration process into additional unnecessary steps.

The rest of this section will expose concrete automation example related to point 5, which allow to automate part of the interactivity related properties' values, and is of particular relevance to for models with GUI similar components.

Components such as buttons, knobs and thumb sticks which, immediately after segmentation, when they still lack geometry at the base, can use the segmented mesh to calculate the direction of the up vector of the model transform. This direction is the release direction of a button, the rotation axis of a knob and a thumb stick resting orientation. The normalized vector up, $\hat{N}$, can be calculated with a sum of the triangle normals with respect to their triangle area. Alternatively, is also possible to construct the geometry of the missing base and apply the same vector.

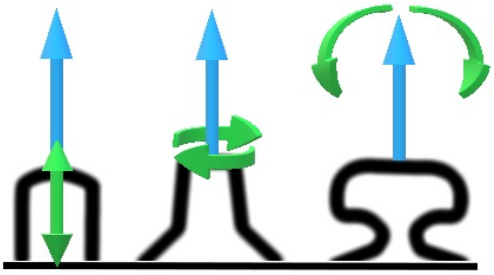


Figure 3.8: Components movement (green arrows) with respect to the up vector (blue arrow). From left to right are displayed the following types of components: button, knob and thumb stick

The formula 3.1 shows how to compute $\vec{N}$, not normalized, using vertex normals and formula 3.2 how to calculate the same without using the vertexes' normals. Note that any constant divisions from intermediate calculations can be discarded as the resulting vector magnitude is not relevant.

$$\vec{N} = \sum_{i}^{t} \mid v_0^i v_1^i \times v_0^i v_2^i \mid (n_0^i + n_1^i + n_2^i) \tag{3.1}$$

where t is the number of triangles of the mesh. v_a^b is the vertex a of the triangle of index b and n_a^b is the normal of the vertex.

$$\vec{N} = \sum_{i}^{t} \max_{x} d(\frac{v_0^i + v_1^i + v_2^i}{3}, x) \tag{3.2}$$

where $x \in \left\{ v_0^i v_1^i \times v_0^i v_2^i, -v_0^i v_1^i \times v_0^i v_2^i \right\}$. $d(c,n) = \mid c + n - \bar{v} \mid$, where $\bar{v}$ is the vertexes centroid.

3.4.2 Properties Set

A way to make the workflow faster and possibly simpler and more expressive is by allowing to group similar component features together into property sets. Similarly to mesh proxys in modeling software, prefabs, blueprints or templates in game engines, a property set is shared between multiple components. Therefore the user can change the configuration of multiple components by editing a property set alone.

A simpler approach to implement and interact with, regarding property sets, is by making these disjoint, i.e. a component can only have one and only one property set associated. This avoids some of the nuances mentioned in the next paragraph and consequently makes the workflow more linear, but also more rigid and in some cases more cumbersome.

If a component can have multiple property sets associated, i.e. non disjoint sets, then the following points need to be taken into account:

- **The ambiguity of property assignment.**
  To avoid ambiguity the property sets assigned to a component, these can not define properties that are equal, very similar in nature or in which one encompasses the other. Or, alternatively,they can but a rule is given to select which property takes precedence.

- **Display approach.**
  The visual representation of the property sets has more information, thus being potentially less simple to interpret, and can't use some of the representation solutions that the disjoint approach can. The way information in conveyed to the user must be more carefully though to avoid information cluttering and be intuitive enough to understand.

It is also possible to organize groups of elements by a common property value instead of using property sets. In theory this is a subset of the possible usages of non disjoint property sets in which only one property is defined, this would remove any ambiguity in property values assignment. Such approach if used alone could make the property assignment cumbersome, but if implemented as a form of "syntactic sugar" over an existing non disjoint property set system, with the right user interface, could be useful.

Figure 3.9: The different types of property sets organization.

3.4.3 Information Display

The integrated environment should also display visual auxiliary icons, annotations and controllable, or not controllable, gizmos over a 3D view of the model to be integrated, preferably on the VE scene where it is used in case of interaction with other models.

The objectives of this additional information over a view of the 3D model are:

- **Annotate task that can't be done using the integrated environment.**
  E.g., modeling of occluded geometry.
- **Annotate components that still need work.**
  E.g., mesh simplification and bump texture creation.
- **Document the integration process.**
  E.g., explain why some of the model's components were, and should be ignored in the future due to the context in which it will be applied.
- **Display current state of the integration.**
  E.g., show the different components already configured.
- **Display properties of the models components and allow fast adjustments.**
  E.g., adjust the default resting position of a spring back knob.
- **Make connections between components explicit.**
  E.g., highlight the component or object that is affected by the pressing of a button.

Although not specified in this proposal, a visual language that expresses the different nature of these informations could be created based of color codes, symbols, icons, or other visual design elements. This would require a taxonomy to classify all the different types of visual information, possibly not only regarding the points listed below but conveying additional information such as, for example, indicating that a specific annotation content is about texture related subjects.

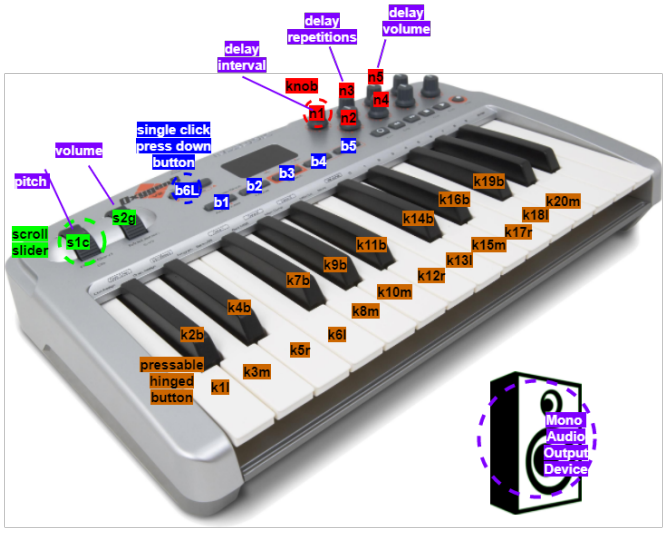


Figure 3.10: Digital sketch of 3D billboard annotations over a keyboard.

Chapter 4

Implementation

This chapter is dedicated to detail the implemented workflow, which, unlike the conceptual workflow specification, is limited by the integrated environment prototype.

Thus, this chapter begins with an overview of the workflow, followed by an explanation of the digitalization and refinement process of the MIDI keyboard, and finally the exposition of the final prototypes' features.

This chapter will also present unused models digitalization attempts in section 4.2.

4.1 Implemented Workflow

The diagram below is similar to the one previously presented in figure 4.1 in chapter 3 but with respect to the implemented prototype. It replaces the unnamed environments by the ones used in the implementation, shows only tasks available in the final prototype and also display which of these were used to integrate the digitalized model.

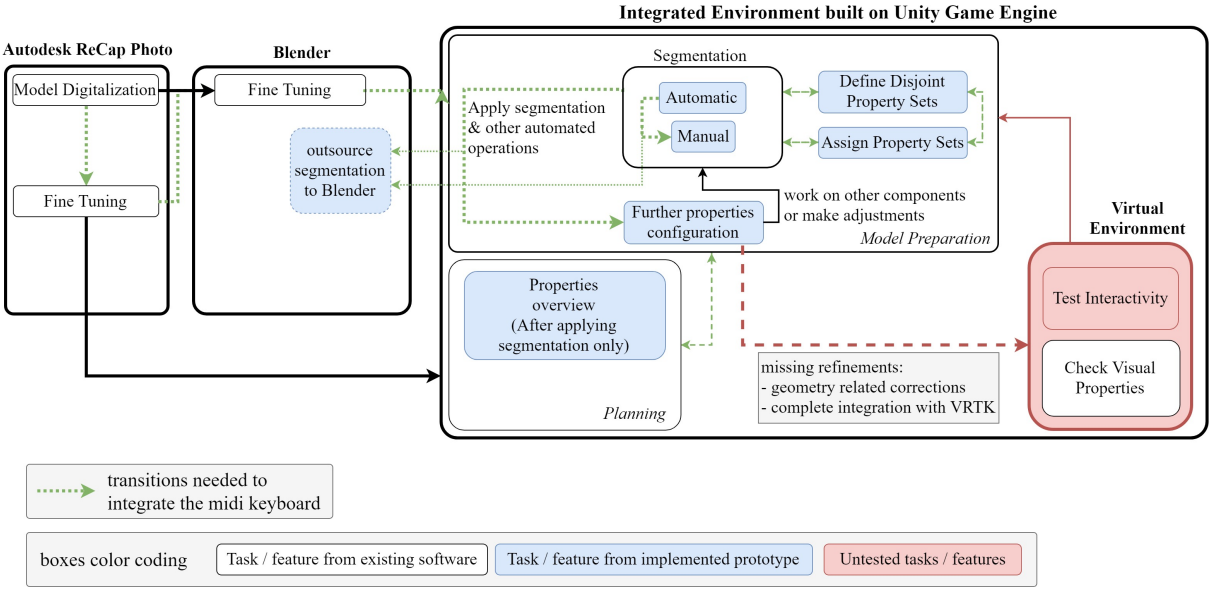


Figure 4.1: Diagram of the implemented workflow.

The following list explains the technologies that appear in the workflow diagram and how they were used.

- **Autodesk ReCap Photo**
  Photogrammetry software to create 3D models from photographs via cloud service. Also allows to apply basic refinements after the model acquisition such as hole filling, retopology, update object orientation, and export the model.

  It was used to capture the MIDI keyboard.

- **Blender**
  "Blender is the free and open source 3D creation suite. It supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. Advanced users employ Blender's API for Python scripting to customize the application and write specialized tools; ..." ("Blender - About" 2018)

  It was used to refine the model mesh and texture before importing it to Unity in an initial phase. Then it was linked to the implemented integrated environment in Unity in order to implement some of the needed automations via the External Scripts Tool that is explained in section 4.3.1. The main reasons for selecting Blender has the external tool that complements the integrated environment are:

  - Already acquired basic knowledge of the Blender capabilities and modeling experience with the software

  - The Python scripting language and API. There are multiple abstractions available for python which can be integrated in blender and potentially speed up the development. Additionally I was already familiar with python and only had to learn the Blender API.
  - The adapted segmentation algorithm was implemented as a Blender extension

- **Unity**
  Unity is a game engine with a C# API for scripting and multiple tools for the creation of VEs ("Unity - Features" 2018). The integrated environment was built on Unity due to the following reasons:

  - Some of its tools, such as the 3D scene editor, have the desired WYSIWYG qualities
  - The C# API allows to extend the editor easily, without changing the source code of the engine, potentially reducing development time
  - Can deploy for most modern platforms with native support for many VR platforms (Oculus Rift, Gear VR, Playstation VR, Microsoft HoloLens, Steam VR/Vive and Google Daydream)
  - Many useful abstractions are available via the Unity Asset Store and open source repositories, which potentially reduce development time
  - Blender friendly. Can import models directly from the Blender format, which allows to implemented automations with Blender directly without needing to change the model format. This is useful to speed up development in a prototype context.
  - Already acquired experience during other projects reduced the C# API learning curve

- **VRTK**
  VRTK (Virtual Reality Toolkit) is an Unity asset which contains a diverse set of VR related abstractions. It supports SteamVR and Oculus SDK and allows to add common VR functionalities to a VE such as common VR movements (e.g., teleportation) and interactive properties such as grabable objects or pressable buttons. ("VRTK - Virtual Reality Toolkit" 2018)

4.2 Model Digitalization and Refinement

The model used to test the developed prototype is a digitalized version of a MIDI keyboard obtained through photogrammetry using the Autodesk ReCap Photo software and services. After computing the geometry from a collection of photos the software also allows to cleanup the model by filling holes, calculating the desired orientation with respect to a surface and basic retopology.

Due to the specularity of its components, environment lightning and possibly other factors, the captured model is somewhat defective, appearing to be melting. This is not ideal and so there were some additional attempts to capture other models but without success.

For these attempts additional software was used, namely 3DF Zephyr and Microsoft 3D scan and Kinect Developer Toolkit. Hardware wise, the first requires only a camera for photo acquisition and the other the Microsoft Kinect.

Zephyr, similarly to Autodesk ReCap, reconstructs the model from a set of photos. The main differences being that it does it locally, not as a online service, and that it allows to create masks, via GrabCut, over the photos, to segment the object of interest. Although not a single model was deemed worth using, possibly because of the nature of the model and the lack of setup experience, by using the mask feature Zephyr was able to create geometry from sample photos which without using a mask, in both Zephyr and Autodesk ReCap Photo, no useful result was obtained.

Microsoft 3D Scan and the Kinect Developer Kit were used with a Kinect to devise a non photogrammetric approach for obtaining a model geometry. From all the digitalization attempts, Kinect provided the worst results. It was observed that although it worked relatively well for large objects, e.g. a table, small objects geometry details are completely lost (see 5.3). This is particularly problematic since the target models have buttons, triggers and similar small-scale components which need to be properly identifiable in order to be interactive. I would also argue that the capture process is more cumbersome than the previous mentioned methods, as it requires the user to slowly move in order to capture the different parts of the object or to join scans from different perspectives in external software.

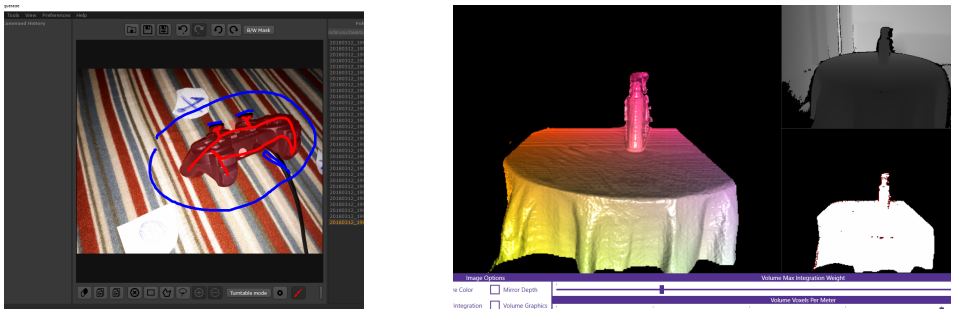


Figure 4.2: At the left Zephyr's GrabCut mask feature. At the right the Kinect Developer Kit Fusion Explorer demo project. Both taken when attempting to digitize models for this project.

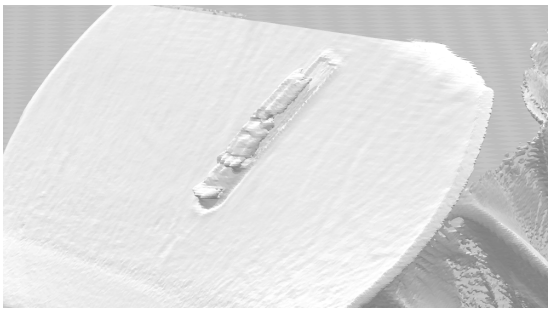


Figure 4.3: Digitalized model of a tv remote over a table, using Kinect.

The MIDI keyboard model, although suffering from a somewhat faulty geometry, allowed to test the proposed workflow without any major problem. The original captured model was slightly modified in modeling software. The base of the keyboard was separated from background geometry, not belonging to the keyboard; automatic retopology was applied; the texture was slightly painted to soften illumination effects.

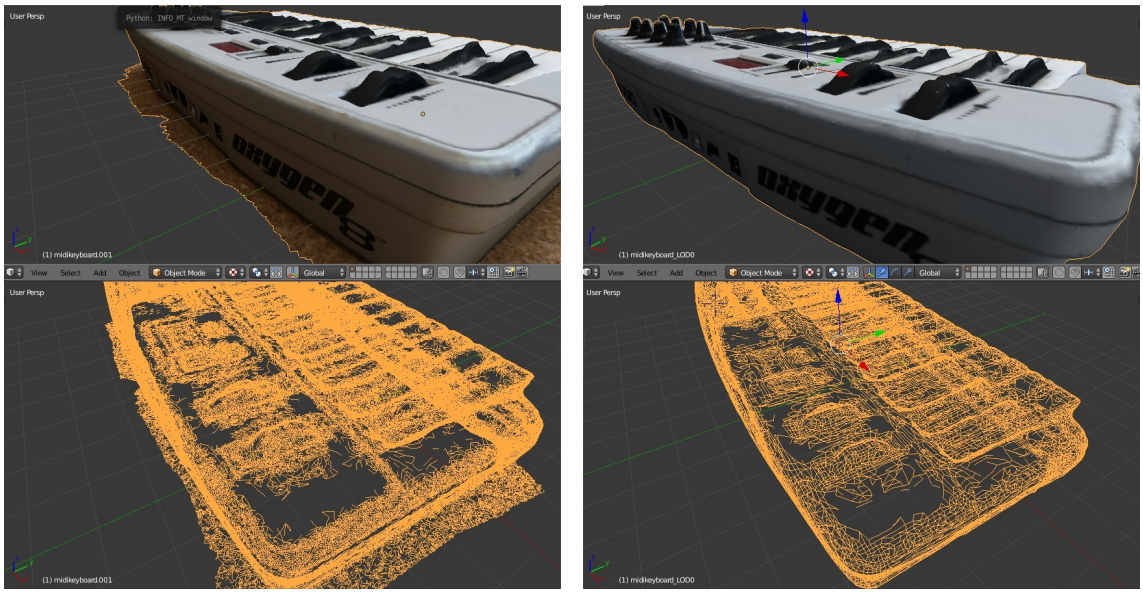


Figure 4.4: The left image is the original captured model with all major background elements already removed. The right picture is the modified version of the model.

Ideally, although not implemented in the prototype, this cleanup should be done, within possibilities, automatically. In this particular model, automatic segmentation and retopology are possibly useful. Not having explored this part of the workflow throughly, which would require more time, additional models, and a better understanding of common procedures, it is not clear what the limits of automation are. This is, however, a very pertinent question to the proposed workflow, as without any automation the workflow can't be properly executed by someone that doesn't have a minimal modeling knowledge.

4.3 Final Prototype

This section aims to clarify the final state of the prototype and the details of implementation of its main features. Each on one the following subsections is dedicated to explaining a particular feature of the prototype. Accompanying the text are images that display the mentioned features.

4.3.1 External Scripts Tool

So as to process the mesh on a 3D modeling software, which contains useful, geometry specific operations, a XML file, that follows the schema specified in the XSD 7.1, lists available operations.

Each operation can have, theoretically, according to the XSD model, multiple scripts for different target software, but in practice only background calls for blender were implemented into the final prototype. These scripts were written in python and placed in specific folder inside the Unity project which is dedicated to store all the operations scripts.

From the scripts metadata it is possible to generate a form with the input required by the script. Each script also contains a dependencies field to document specific required libraries not available by default in the target software. This could be improved further by adding automatic installation of these features.

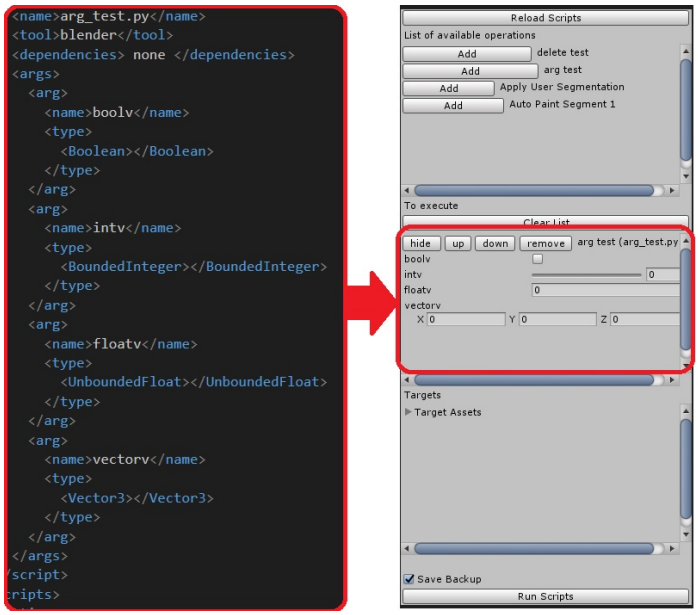


Figure 4.5: Unity's custom inspector showing the script's XML defined parameters.

Currently, the only use of this component is to allow the user to run segmentation related scripts in blender from Unity without changing context. However, the implementation allows the running of a queue of operations over a queue of objects which could easily be used to, for example, create multiple LODs meshes for a groups of objects.

It is also possible to make a backup of the objects before executing the operations but the feature is not advanced enough to allow an historic like management of different versions of the model.

4.3.2 Disjoint Property Sets (Groups) and Segments (Elements)

The prototype uses disjoint property sets, named as groups in the tool.

A group has an identification name, a boolean that indicates whether the material of the segments belonging to the group should be a distinct material from the source model, and a possible list of scripts to be added by the user. These scripts must be Unity's MonoBehaviour scripts and can be assigned by manually dragging them to a groups script list in the GUI.

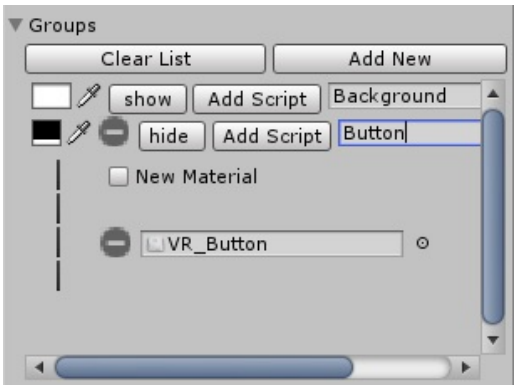


Figure 4.6: Prototype's groups editor.

The group named Background in figure 4.6 is not deletable and its the default group attributed to new segments until specified otherwise.

The segments, named elements in the editor, also have their own editor tool implemented. Each segment has a name, a color a group to which it belongs. The elements editor tool allows to create and remove new elements, change their group and name.

Due to the way information is meant to be used later on by the tool and by the Blender scripts, elements and groups must have distinct names. A warning is displayed on the editor when this condition is not met.

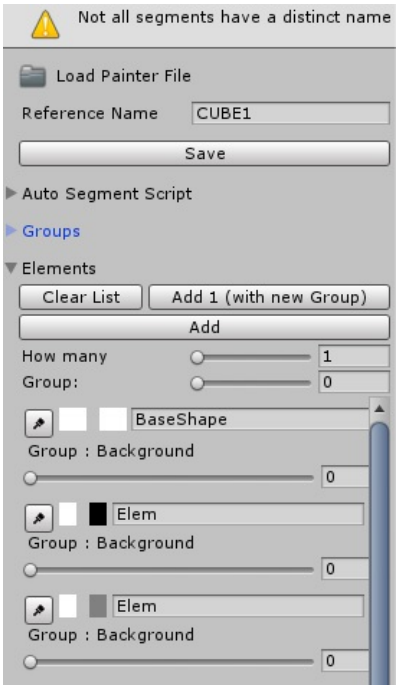


Figure 4.7: Prototype's elements editor and "same name" warning.

4.3.3 Automatic Segmentation

The automatic segmentation algorithm used in the prototype is a modified version of the kugelrund 2013 implementation of the Liu and Zhang 2004.

Both the original and modified algorithm use a Spectral Clustering Algorithm[1] approach. This approach requires building a matrix with weights for each vertex pairs which is not compatible with dense meshes as it requires excessive computation resources. Since most digitalized models, unless manually optimized, usually have a very dense mesh, the mentioned implementation can't process it efficiently. To solve this, in the modified version, a mesh that has a high number of vertexes is simplified by applying the decimate modifier, effectively reducing the vertex count to a pre-specified amount.

After the segmentation of the simplified model is calculated, each face is given a material with a color representing the segment to which it belongs. The result is then baked into a UV texture in the original model.

There are many ways to simplify a mesh automatically. Blender, for instance, contains several modifiers that can be used for retopology. The resulting mesh from the retopology process may not allow to do a bake targeting the original mesh that is artifact free. This can happen independently of the technique used because of the loss of important geometry or due to the radical way it differs from the original topology. The decimate approach leaves some of the geometry intact, thus minimizing the risk of rendering the bake result unusable. However its still possible to lose important details which results in artifacts.

Additionally, although not integrated in Unity on the final prototype, an additional feature was added. The original implementation only takes into account the geometry of the mesh, and uses as features the geodesic distance[2] and angular distance[3]. In the modified version a a color based feature was added. The feature is calculated by computing the average color of the faces and computing the distance between them using the Delta E 2000 Gaurav, Wencheng, and N., n.d. Although this difference is not necessarily the best for segmenting the model, and I conjecture that a better approach would require to take into account the histograms of colors on the model texture similarly to some Computer Vision segmentation algorithms, it is intuitive to interpret as it tries to match the human perception of color difference Mokrzycki and Tatol 2011.

The following diagram visually illustrates the algorithm with the added modifications blue colored.

In the prototype the automatic segmentation is initialized by Unity and executed by Blender in the background by using the External Scripts Tool. The final result is not a set of meshes but a preview texture of the segmentation which is displayed on the model being edited on the Unity scene.

[1] Read Von Luxburg 2007 for more details regarding this clustering approach.
[2] distance between the center of two triangles passing through the edge.
[3] the angle between the normals of the triangles.

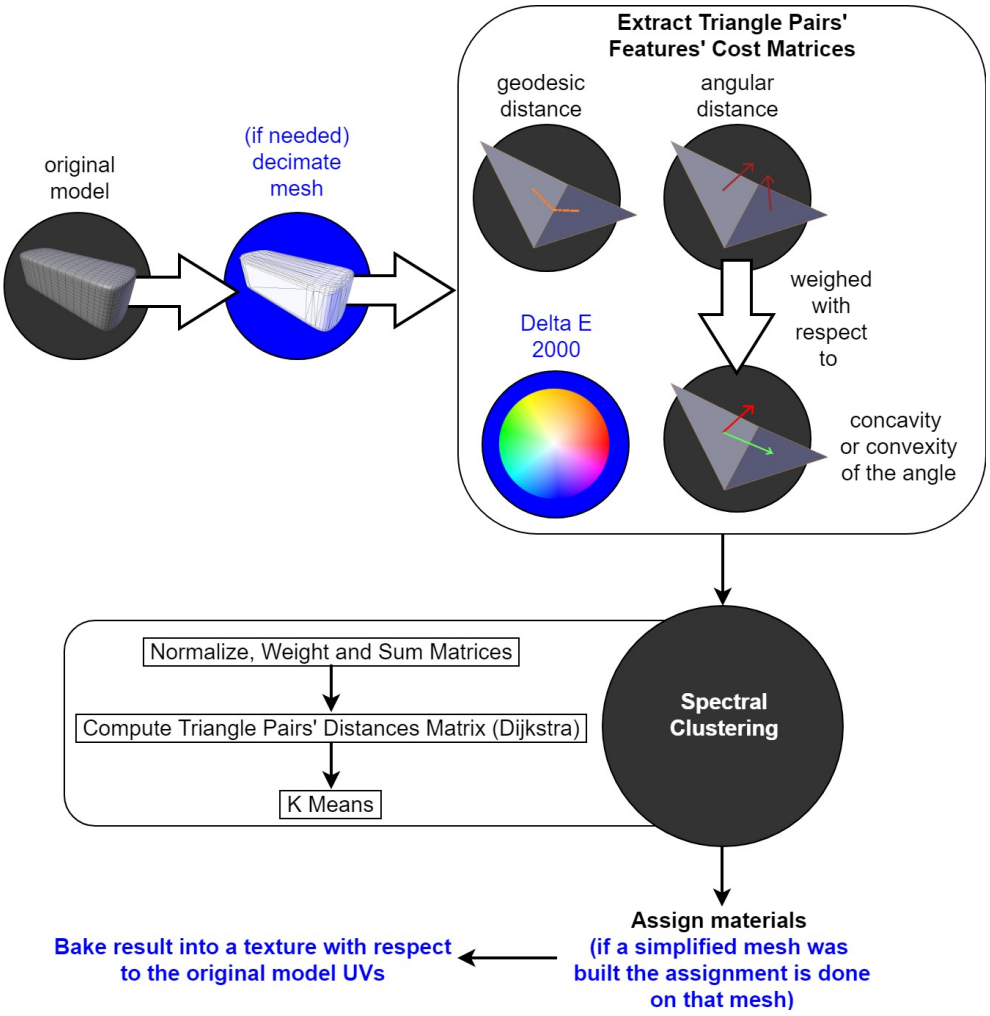


Figure 4.8: Visual summary of the segmentation algorithm

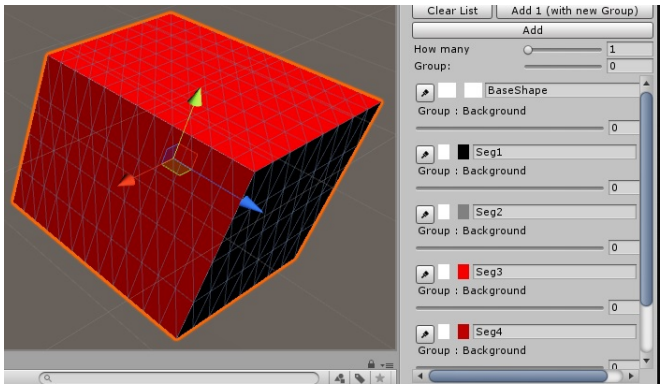


Figure 4.9: Segmentation preview via texture of an automatically segmented cube in Unity.

4.3.4 Manual Segmentation

To complement the automatic segmentation a manual segmentation tool was implemented. Although there was some experimentation with some other methods, the only one integrated on the final prototype was a volumetric based texture painting brush complemented with some additional auxiliary features.

4.3.4.1 Spherical Brush Details

The brush, which is spherical, is drawn over the pointed 3D point on the model with some transparency to allow to see what is beyond it.

Its size can be changed via a slider or key shortcuts. Although the slider is linear, which is not ideal for switching between big and small brush sizes intuitively, the key shortcuts also update the brush in a logarithmic fashion allowing the user the rapidly change between different brush sizes. The scale of the model transform on the Unity Scene is also used to make the brush size proportional to the model. This means that the editing process is almost independent of the scale of the model on the scene so that the user can work on it equally at any scale. The only minor problem is if the model scale is not equally changed along all the axis. The brush uses the average of the scale along the three axis but there is no way to flexibly adapt it to extreme changes along one or two dimensions.

The final implementation iterates over all triangles and updates all the ones that are "contained" within the brush. The selection condition is that, considering an additional imaginary vertex in the center of the triangle, three of the four vertex of the triangle are contained within the brush volume.

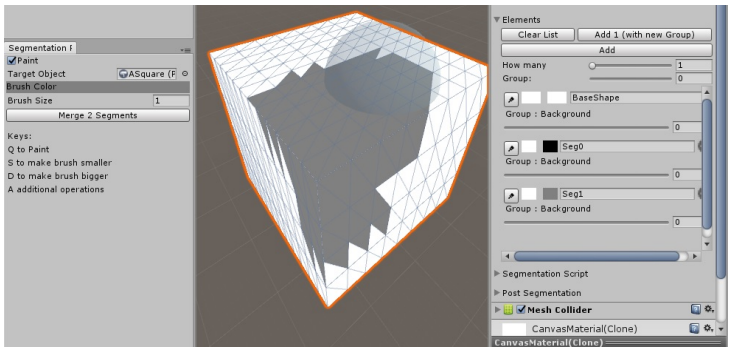


Figure 4.10: Final prototype manual segmentation with a spherical brush

4.3.4.2 Texture Segmentation Problems

Since the segmentation is previewed on a UV mapped texture, an additional iteration over this texture is required per selected triangle. Because of these additional texture iterations, depending on the texture size and on the number of triangles selected by the brush, there may be some lag which may make the tool harder to use. Since the current implementation must abide by Unity's structure, I'm not sure what all the concrete possible solutions are. In abstract, the way to enhance

the performance of the painting operation is by using an alternative way to draw the color without relying on the texture which may need to be arbitrarily big and thus mapping one triangle to an arbitrary number of pixels. Ideally, a triangle should only be associated to one color and no additional data. Another problem using a texture to preview the segmentation is texture bleeding. This happens because the texture can't be used to map the UVs correctly if it's resolution is not high enough, but, if the resolution is too high then the previously mentioned performance problem occurs. Besides increasing the texture size, it is also possible to minimize the problem by editing UVs, which, unless is done automatically via a script, is not ideal as it requires the user to switch context and might be too hard to do since a digitalized model mesh usually has a topology very different from that of a manually modeled 3D model.

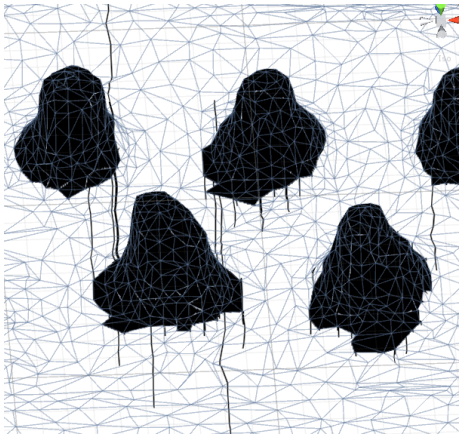


Figure 4.11: Segmentation preview displaying texture bleeding

The mentioned points are solvable by replacing the texture-based painting by a poly-based painting segmentation. This would also allow to work with models that do not have UV data. The number of triangles of the digitalized model also as the potential to be a problem but on the used models it had no effect. If at some point this iteration becomes too heavy, Classic 3D acceleration techniques may be a possible way to solve the problem. For example, building a spatial tree (e.g. octree) would allow to only iterate over a section of the geometry.

4.3.4.3 Complementary Features

To help the user cleanup useless segments created by the automatic segmentation or by user mistakes, it is possible to merge or delete elements via the painting related windows.

The user can also point to the model and open a popup window with information regarding the element to which the pointed triangle is associated with. This popup contains the following list of operations:

- Set the segment color as the brush color.

- Delete the element being pointed by painting it white and removing it from the list of elements.
- Merge all the elements of the group into one single element.
- Delete all the elements from the group to which the element belongs. (Can't be done on the zero index group)

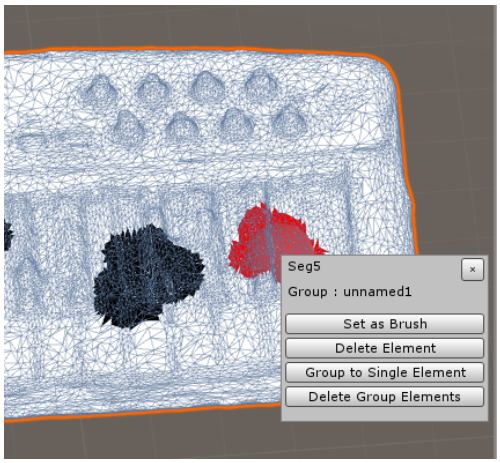


Figure 4.12: Popup with additional features.

4.3.5 Applying a Segmentation

All the segmentation techniques mentioned so far are only previewed with a texture but not actually applied to the model.

To apply a segmentation, the preview texture is exported to a binary file as an array of labels that correspond to the user defined segments. This file also stores all the information regarding the groups and segments. The user can then use this file as source parameter for an external script responsible for applying the segmentation. By saving to a binary it is also possible to save multiple versions of the segmentation, iteratively, and compress them to save memory space[4]. The binary file structure is presented in 7.2, in the annexes section.

The following are the additional parameters which allow the user to define other details regarding the segmentation application:

- Keep the zero index mesh (white colored) intact. Can be used to manually remove the separated components later.
- Define how the origin point of the component. A better implementation would ideally allow this to be defined per groups and possibly overwritten by special individual segment.

[4]This compression is not currently implemented.

- Define how the hierarchy of segmented components is built over the existing hierarchy.

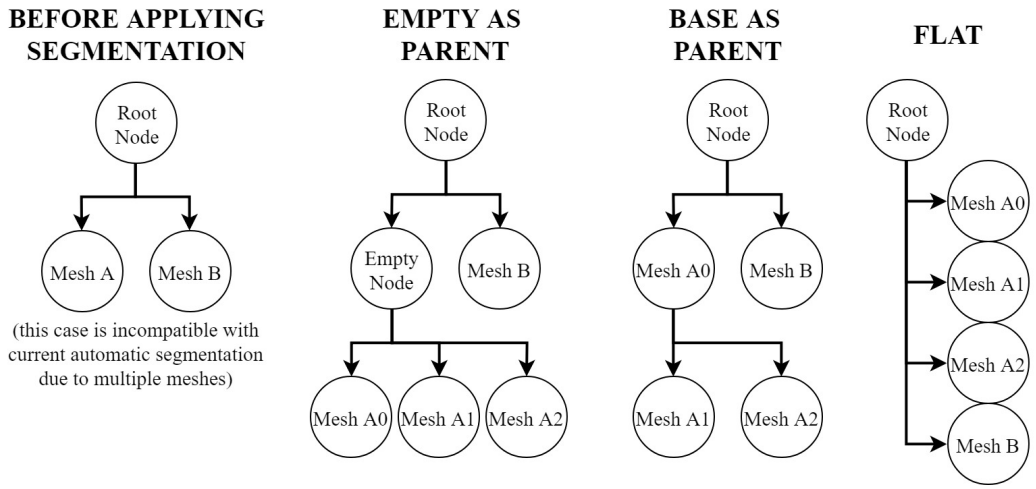


Figure 4.13: Prototype's hierarchy mutually exclusive options.

4.3.6 3D Handles and Annotations

To allow the user to overview the different components and quickly adjust their properties a custom abstract Unity inspector was implemented.

In Unity, an inspector is the form like window through which the developer can interact with scripts instances on a Unity scene that allows to adjust values of in-scene objects without losing context. A useful feature of custom inspectors is that they can draw visual gizmos onto the Unity 3D scene to display information or allow the user to set variables.

The implemented abstraction implements and forces extended classes to implement specific methods which are recognized by other components in the prototype and used to draw a visual representation of the component without this being focused on the editor. Thus, it is possible to have a customized view of all the model's components.

In the final prototype a simplified 3d representation of the script values and an alternative text view are implemented.

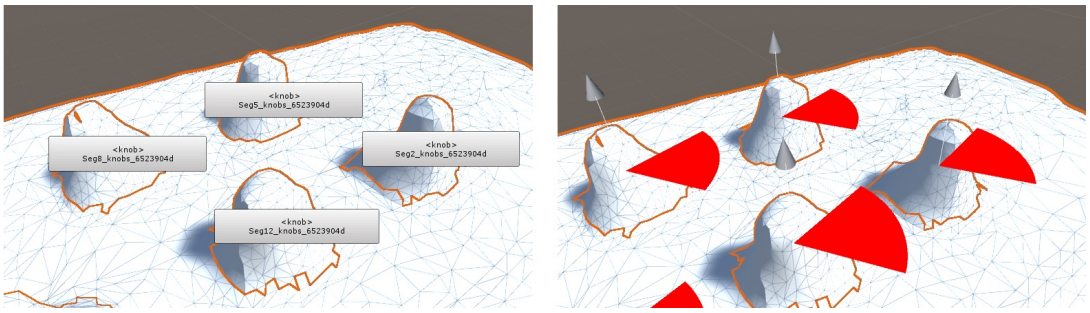


Figure 4.14: Visual representation of the script on the scene.

Additionally, custom 3D adjustment tools are available when the component is on focus.

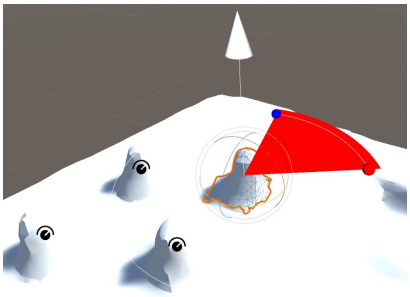


Figure 4.15: 3D adjustment tools, implemented using Unity's Handles.

Initially, this visualization was to be implemented over VRTK, but due to development constraints the VRTK components did not make it into the final prototype.

4.4 Initial Prototype Related Experiments

During the experimentation phase of the project were explored and developed solutions, some of which weren't included in the final prototype such as: volumetric based segmentation, material and texture automations, and image based segmentation.

This section will expose the details of the image based segmentation approach explored. Note that a complete functional solution using this approach was not implemented. By exposing the explored processes the potential of such approaches, has an alternative to 3D segmentation algorithms, such as the one used in the final prototype, will become clearer.

4.4.1 Image Based Segmentation

A tempting solution to the segmentation problem was the possibility of using image processing algorithms, available in OpenCV[5], to segment the texture of the model, which is almost a guaranteed asset in photogrammetric models. This approach would allow to segment elements which geometry its not prominent, such as the keyboard led display (see figure 4.16), and is typically faster and more scalable than 3D segmentation algorithms.

The following figures 4.16 and 4.17 display in the rightmost image the segmentation resulting from using OpenCV contours over the remaining images. The contours are filtered according to their area, aspect ratio of their enclosing rectangle, and convexity to avoid useless segmentations.

[5]Open Source Computer Vision Library

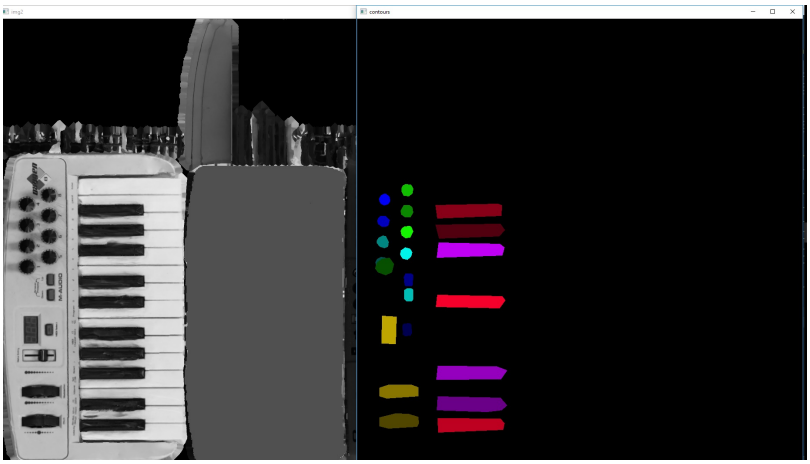


Figure 4.16: OpenCV image based segmentation of a grayscale version of the model texture (after refinements).

To segment elements that do not have a distinct color from their surroundings but have prominent geometry a displacement texture can be calculated from a retopologized model[6] since most types of the components to be segmented, similar to GUI components, are easily identified within this texture.

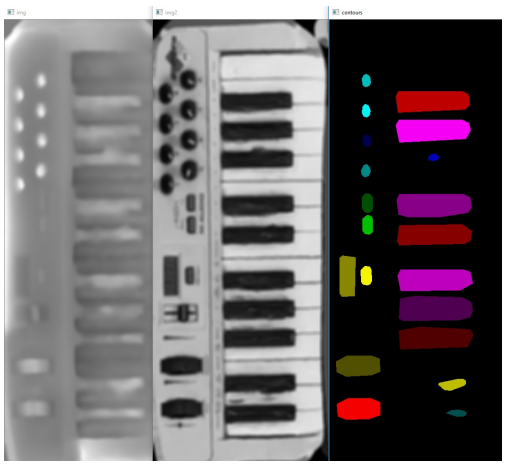


Figure 4.17: Segmentation from retopologized model displacement and grayscale color textures.

The problem with this approach is that the UV layout is not continuous, in the sense that some of the triangle UV form isolated islands away from the rest of connected geometry. This, as the figure below depicts, doesn't allow to directly apply a segmentation using the textured generated with OpenCV contours because it does not map the desired segmentation correctly onto the model.

[6]This retopology can create problems later as mentioned in REF

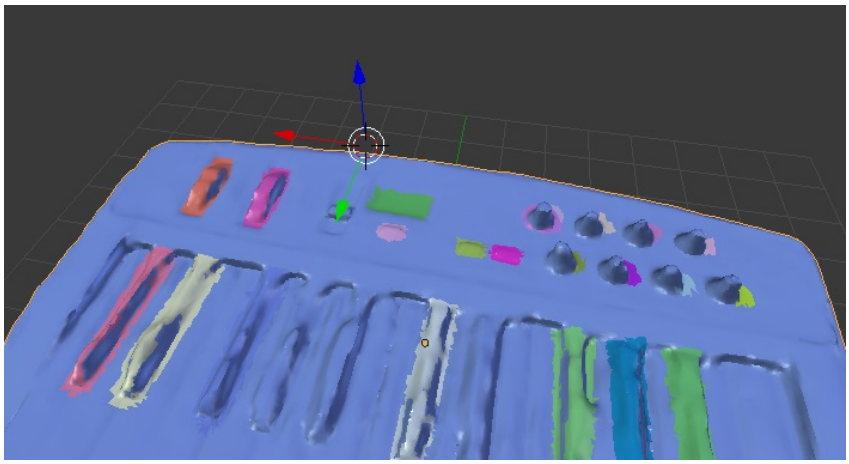


Figure 4.18

A possible workaround to this problem is by applying an operation similar to the OpenCV's closing but that respects the model's geometry and UV layout. Because such operation would require to research, or devise, and implement an additional algorithm in Blender, or in Unity,this was considered too costly to implement when compared to the possibility of adapting the segmentation algorithm which ended up being the chosen approach.

Chapter 5

Validation

This chapter will present the user tests done to validate the proposed workflow and implemented prototype. Then the results obtained from these tests are exposed and interpreted. Possible improvements to the current prototype are also suggested based on the users feedback.

Finally, in section 5.2, are presented the tests done to the adapted segmentation algorithm used in the final prototype.

5.1 User Tests

To validate the tool the target user must be taken into account. The broad characteristics that are relevant to consider regarding a user's profile are his/her experience with VE content creation, namely 3D modeling and Game Development[1]. The user can be seen as belonging in any of the four groups depicted in figure 5.1 , depending on his/her experience on those fields.

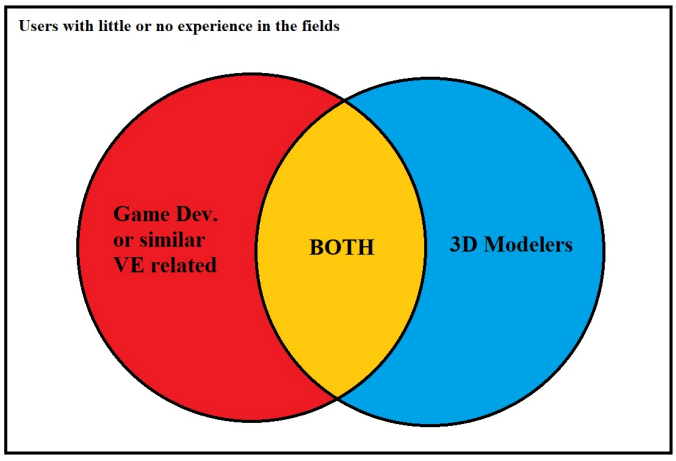


Figure 5.1: Diagram of potential users groups. The groups' area is not representative of their population.

[1]Not necessarily Game Development but any VE content creation related tools that focus on the interactivity of a VE and not the modeling of the objects therein present.

To validate the prototype, it would be necessary to evaluate it thoroughly with the different user groups to completely understand its impact. However, that would require an extensive evaluation. With respect to this project's scope, this is not feasible. Thus, the validation herein presented is not by any means extensive or conclusive. It should, however, provide preliminary results and a testbed foundation that can be improved upon.

The validation was done by targeting 2 different groups of users. The graph in figure 5.2 shows the "modelers" target group colored blue and the "non modelers" group colored red.

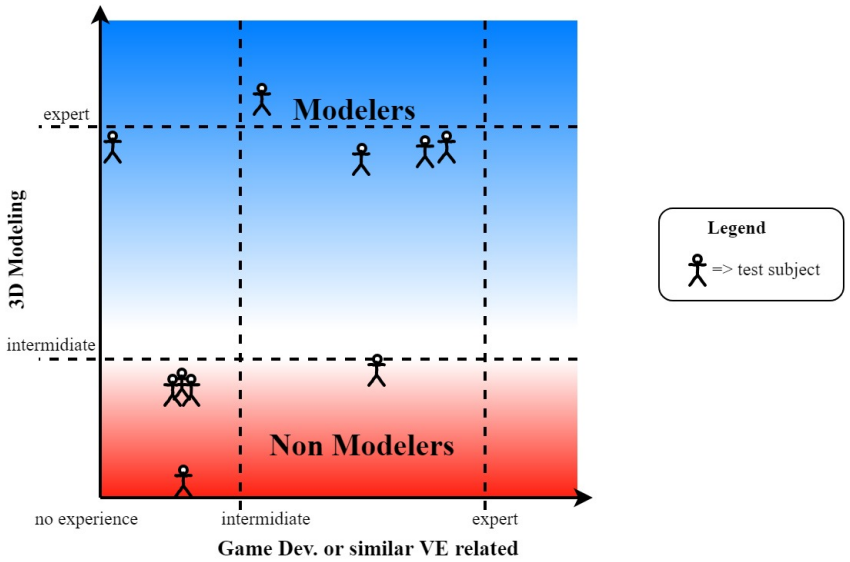


Figure 5.2: Graph of the targeted user groups and test subjects relatively expertise in the fields of 3D Modeling and Game Development (or similar).

For the non modelers group a usability test was devised in which the users had to explore all of the prototype features and evaluate the prototype, usability wise. For the modelers groups another test was made to compare the traditional approach to the proposed one. Regarding this last test group, there are a number of problems that invalidate an objective quantitative analysis with the gathered data, as will be explained later, being the qualitative feedback received possibly the most relevant data gathered from the tests. Both groups were required to fill a questionnaire regarding the workflow, the usefulness of certain features and the tested prototype. They were also required to fill a form regarding their user profile which was sent to potential testers before scheduling the tests. This form also included a video explaining the basic features of the tool.

To gather the maximum quantity of data during the tests, a keylogger was used in conjunction with video and audio recording of the laptop screen of the machine used to do the test and of the target subject. The video and audio recording of the subject were optional components for a more in depth analysis of the tests and the other were deemed as essential. To register the users' consent, two forms were given before starting the test and a duplicate was sent to the users with the test observer signature. The forms template is available in the annexes 7.8.

5.1.1 Non Modelers Validation Test

All the test subjects belonging to the Non Modelers groups have had minimal experience in game development or 3D modeling, but never enough experience to be considered modelers.

The subjects were given a paper with a task-list but, to accelerate the process, in most cases, the tasks were given verbally with complementary remarks. Due to the time completion of the pilot tests being long, some hints were given to compensate for the users' lack of knowledge regarding the Unity interface or general usage of the implemented tool to further accelerate the tests.

The test established the objective of segmenting the previously presented MIDI keyboard model knobs and explore some of prototype functionalities. And it was divided into the following steps:

1. Add Painter Target component to the model;
2. Set Reference Name field to: user<user's id>;
3. Open the "Automatic Segmentation" and, without changing the "texx" and "texy" fields, experiment executing some automatic segmentations until a good enough result is achieved. (If the automatic segmentation, correctly delimited the 8 knobs, two were removed before starting the next step);
4. Create a new group named "Knobs" and add the script "VR_Knob" to the group;
5. Delete non relevant elements (not knobs) and add new elements for the missing knobs;
6. Set the knob Elements as belonging to the "Knobs" group;
7. Segment, without using automatic segmentation, the remaining knobs;
8. Save the result and apply the segmentation;
9. Explore the Post segmentation section;
10. Set a knob angle interval of 90 degrees.

Not all the steps need to be executed in the presented order. The users were guided in order to make the interaction more straight forward. For example, a user that immediately started checking the elements after applying the segmentation was told to execute step 10 before mentioning step 9 to avoid needless context changes. Steps 5, 6 and 7 were also dynamically ordered according to the users interactions and feedback.

So as to evaluate the prototype with this category of users, the SUS (System Usability Scale) questionnaire was used[2]. In the section 5.1.3, further below, the results are summarized and interpreted.

Regarding the final questionnaire, which was shared with the modelers groups, the results are summarized and interpreted in the 5.1.5 section.

[2]See Brooke et al. 1996 and Brooke 2013.

5.1.2 Modelers Validation Test

The modelers test, regarding the Unity tool, was a subset of the non-modelers test which required the testers to only separate the knobs of the MIDI keyboard model. Before this however, they had to model the knobs using a modeling software, imitating the procedure they would typically follow to integrate the model. Note that they modeled the knobs and did not isolate some geometry as was done with the Unity tool. This is because, conceptually, some retopology operations are automatic procedures in the proposed workflow. Unlike non modelers, the modelers test focuses more on validating the potential of the workflow and not so much on the usability of the prototype, despite this being a factor that sometimes influences the perception of the users.

To compare the workflows a questionnaire was given only to the modelers to measure the perceived qualities of both approaches, results are presented in 5.1.4.

The keylogger data and task completion were also initial thought to be useful to make a quantitative analysis of the number interactions and task completion times. However, this idea was scratched due to the following factors influencing the results:

- **Environment problems.**
  Examples: Mac users struggled with windows keyboard shortcuts; some Blender definitions were not exactly as the testers were accustomed.

  These kinds of environment differences can skew the collected data.

- **Task failure.**

  In some cases the task of segmenting the model components on Unity was not successfully completed. It is not clear whether the data received upon applying segmentation should be used for comparison. Note that because it is more important to evaluate the workflow, prototype related problems should not be taken into account. It is always possible to use only successful tests but this shrinks further the already small test population.

- **Arbitrary rigor.**
  During the pilot tests not enough information was gathered regarding the different ways to model the knobs. This resulted in the different modelers opting to use unanticipated solutions with different levels of rigor. For example, one of the modelers used an array modifier to guarantee that the knobs have the same distance between each other. This is a somewhat more rigorous approach than manually dragging knob proxys to their respective position and also has an impact on the number of interactions and execution time. Another similar example is of one of the modelers not deciding immediately by them self that the procedure was complete, and continued to apply some extra adjustments.

  It's not clear whether this is part of natural modeling process and should be taken into account, adding more variation to the results, or whether the test script should define with greater precision what needs to be done.

5.1.3 SUS Results

The questionnaire answers are available in the annexes 7.4.

The final score of 67,5 indicates that the prototype usability and learnability is acceptable but not particularly good.

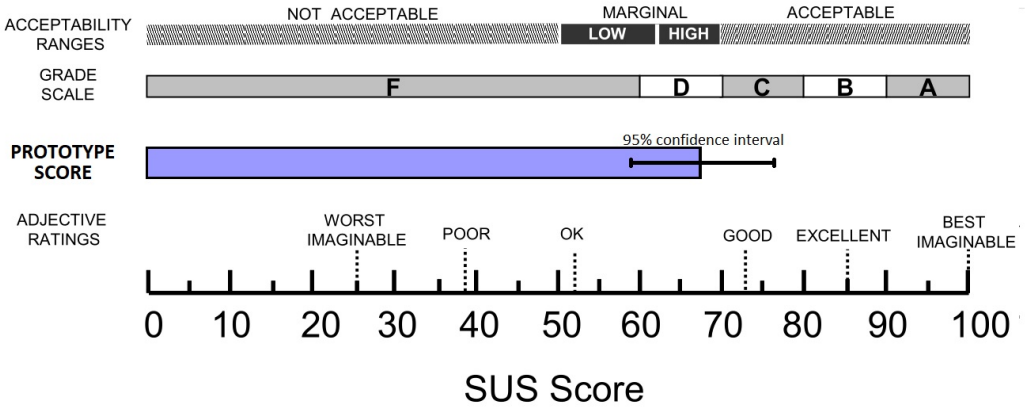


Figure 5.3: Adapted from Bangor, Kortum, and Miller 2009. Shows the prototype score in relation to different metrics.

This result may, however, be misleading. This is due to two factors.

Firstly the lack of experience of the users in specialized software, which possibly makes them more impressionable may yield biased results. This was subjectively observed in some of their reactions during the test.

Secondly, the number of tests executed might be insufficient. In Tullis and Stetson 2004, a higher number of users is required to correctly identify the preferable website using SUS or other questionnaires. Although the target application domain is not the same, this, at the very minimum, indicates that the presented results shouldn't be used for comparisons with possible future iterations of the prototype or similar tools. This significantly reduces the usefulness of the gathered data in future research.

Taking into account that the prototype is missing important features, such as an undo operation, the SUS results point that at least a usable tool that follows the proposed workflow in not completely infeasible. It remains uncertain whether the usability of a tool that follows the proposed, or similar, workflow plateaus at some point regarding usability metrics due to limitations originated from the workflow.

5.1.4 Modelers Only Questionnaire Results

The modelers only questionnaire tries to compare the user perceived speed, ease-of-use, intuitiveness and pleasantness of a traditional approach versus the proposed approach. The users were asked to score each of these qualities for each approach from 1 to 5, where 5 indicates a positive appreciation and 1 a negative appreciation.

The findings obtained through this questionnaire are presented in rest of this section but, because the numbers of testers is low, the results obtained should be interpreted with caution and the findings should not be taken as valid assertions but rather as educated guesses.

The approach used with the prototype scored worse than a traditional approach with respect to all the measured qualities, averaging a total score of 9,5 out of 16[3] per user while the traditional scored an average of 12,75. This indicates that the prototype or the workflow are usable but are not yet an attractive solution when compared with a traditional approach. It is also important to take into account that the results are possibly influenced by the fact that the users are using the system for the first time and that there could be some resistance to the new paradigm, which could be further aggravated by the unpolished state of the prototype. When considering the modelers feedback during the tests, the reasons behind the lower scores given seems to be influenced by some of the prototype's usability problems presented in 5.1.6, which were mentioned during the tests.

It is also worth noting that the speed and ease-of-use qualities were more positively and consensually (less varied answers) classified than the intuitiveness and pleasantness on both approaches. A possible conjecture to explain this is that these qualities are, by nature, more volatile than the others, being more influenced the users' test experience or their personal subjective perspectives on the matter.

The graphs with the questionnaire results are available in 7.5, on the annexes.

5.1.5 Shared Questionnaire Results

The first three questions are related to the workflow.

Regarding the first question on the final "general" questionnaire - "What problems do you think might occur if this "workflow" were applied in a professional production context?" - there was not much feedback from non-modeler users. The single feedback provided by one of the testers was that the used tool could be problematic without proper documentation. Although it is tempting to regard such feedback as a "mistake", because it seems to be more focused on the implemented tool and since the user hasn't the background to backup any claims regarding a production context, there might be some truth in this statement. If the methodologies present deviate considerably from standard practices, there won't be substantial available information regarding them which could render them as not reliable. As some of the modelers pointed out, the process suffers from: lack of intuitiveness; lack of control over technology; a possible high learning curve. All these points also point to the need to have proper, comprehensive documentation or possibly specialized training (e.g., workshops).

Another problem, indicated by modelers, is that they might need to use modeling software anyways due to some functionalities not being present on the integrated environment. This problem has already, to some degree, been anticipated by the workflow specification, and was confirmed with the received feedback. Also, as pointed out by one of the modelers, even if something can

[3] Score calculated summing the values attributed for each measured quality minus 4. This considers the values with an offset of minus one so that the minimum possible score is zero and the maximum is 16.

be done on the integrated environment alone, if it's not possible to produce exact results then, due quality requirements, a modeling tool might need to be used.

Lastly regarding positive aspects of the workflow, it was pointed as being: "potentially faster"; more integrated - "removing the need to switch between different work contexts"; and "allowing variety rapidly". Regarding this last point, mentioned by one of the modelers, the following example was presented by the user: "... if I want a set of cracked keys on the keyboard, for example, I can define a group of keys and obtain more variety more rapidly ...". This seems to be a similar comment to another non-modeler user's feedback on the same question which mentions "ease of doing multiple tasks on multiple objects in a single iteration".

Question 4 results, regarding the perceived importance of features that a tool used to follow the proposed workflow may have.

Results seem to indicate that non-modelers prefer segmentation via painting over enclosing volumes and consider the automatic segmentation as the less valuable form of segmentation. Contrarily to these, modelers seem to value segmentation via painting the least and value automatic segmentation the most. Most modelers mentioned the preference for the volume based approach over the painting approach, not only on the questionnaire but also verbally. The reason why they prefer the automatic segmentation over others in contrast with the non modelers user group is not clear, some modelers even mentioned some of the difficulties of the automatic approach.

Regarding the groups organization[4] there doesn't seem to be any preference of disjoint groups over non-disjoint. Some users did mention they'd prefer the second, but the global results do not reflect this.

For non-modelers automatic retopology is mostly only somewhat important while modelers seem to lean more over the very important option. I conjecture that the non modeling group probably does not fully understand the implications of not automating this task to some degree, and their point of view is more revealing of their lack of knowledge than the usefulness of the tool. The fact that one of the non-modelers users did not even answer this question also seems to point into this possibility.

Question 5 results will be omitted in this section as they are a complementary to the analysis of the prototype usability. They are available in the annexes 7.7.4.

Question 6 results show that both modelers and non modelers see some potential in the proposed workflow.

The shared questionnaire is available in 7.3 and graphs with the results of questions 4 and 6 are presented in 7.6.

[4]Conceptually equivalent to property sets explained in the 3.4.2 section. Named changed to reflect the prototype nomenclature.

5.1.6 Prototype's Usability Problems and Possible Improvements

Upon reviewing the test videos the following problems presented in table 5.1 were detected. The possible solutions, some of which suggested by users, are listed below in table 5.2.

Table 5.1: Frequency of found usability problems

ID	Frequency	Mentioned or observed problems
1	high	need for undo
2	high	color codes are confusing (some colors are too similar)
3	high	group sliders are confusing and provide little feedback
4	high	automatic segmentation is cumbersome (e.g., values reset)
5	high	group assignment is cumbersome
6	high	could not completely understand how to setup automatic segmentation parameters for better results
7	medium	users expected some color related UI elements to be clickable
8	medium	cleanup of extra undesired segments is cumbersome
9	medium	add element options are confusing
10	low	need for a key shortcut to close utilities popup window such as escape key
11	low	it is not clear what is the target color when opening the utilities popup near an edge
12	low	slow feedback when deleting elements (always observed but only 1 complaint)
13	low	"Set as Brush" option is not clear
14	low	some triangles are difficult to paint
15	low	focus should change when pressing enter to validate a parameter value

Regarding point number 8 in table 5.2, the possibility of changing the scope of the segmentation to only process a section of the model is already doable in the current prototype but with some caveats. To do this, the user is required to apply a manual segmentation over the area of interest, and then apply an automatic segmentation over the resulting mesh. Unfortunately, this approach has three problems: it requires the user to already understand the full workflow of the tool; it is not very intuitive as it requires the user to add an additional Painter Target component to the segmented mesh; and, most importantly, breaks the desired utility of the tool because, with the current implementation, the usage of multiple Painter Targets on a single object doesn't allow to preview all of its components simultaneously and also requires the user to save multiple files for each Segmentation Painter used.

Table 5.2: Usability Problems Solutions

ID	Possible Improvements
1	Implement a stack of user requested operations and their values, serving as the basis for Undo/Redo features.
2	Use higher contrast colors or textures instead. Add labels over the segmentations. Show elements' names when hovering over them. Keep the same segment colors when deleting elements.
3	Use drop-down selectors similar alternative.
4	Allow a historic of used values accompanied by a picture of the result. Solving problem 6 should also improve its usage.
5	Use a painting tool similar to the one used to paint segments but targeted for groups. This tool, instead of painting the texture, sets the group of the elements contained inside the paintbrush, excluding the background element. Allow the user to change the group directly on the utilities popup window.
6	Abstract the user from the parameters and allow them to change them from sample results. Add comprehensive documentation in addition to the already present parameters descriptions.
7	Make brush color changeable by clicking the color label in Painter Window and also by clicking the element color on the Elements section in the Painter Target Inspector. Changing the current layout can also help.
8	Solving problems 5 and 12 can make it less cumbersome. Changing the scope of the automatic segmentation can also prevent the need for this step entirely.
9	Change the layout and rename options.
10	Add keyboard shortcut(s).
11	When such selection is done, the user is asked what was the desired target color. Showing the color on the utilities popup window could also work.
12	Improve the core painting system implementation.
13	Rename option. Add a picker icon to the button.
14	Relax selection conditions. Use only distance to the center of the triangle as criteria instead of requiring the volumetric to enclose multiple points of the triangle.
15	Correct focus behavior.

5.2 Modified Segmentation Algorithm Segmentation Results

The modified segmentation algorithm allowed to segment the keyboard, whose mesh was too dense to be segmented by the original algorithm. By using different levels of details was verified that the segmentation does not suffer significant changes and can be applied correctly using the modified version if its not given priority to the geodisic distance.

This can be observed in figures 5.4 and 5.5. The first shows the results of a color distance focused segmentation over different levels of detail and the second figure shows an angular distance focused segmentation.The numbers above the models indicate the number of triangles that make up the mesh of the model. The leftmost model is the original, non decimated, model.

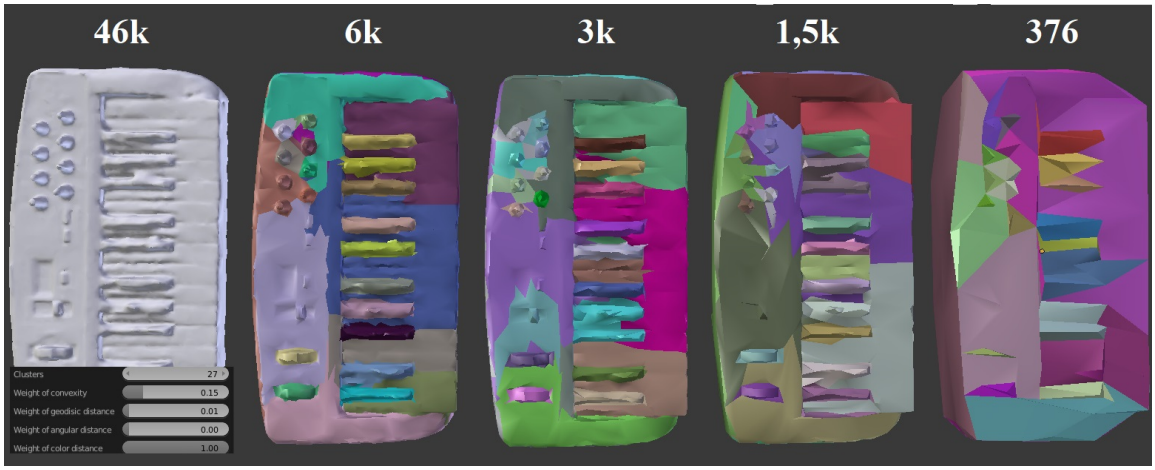


Figure 5.4: Automatic segmentation focused on color distance over different LODs using 27 clusters.

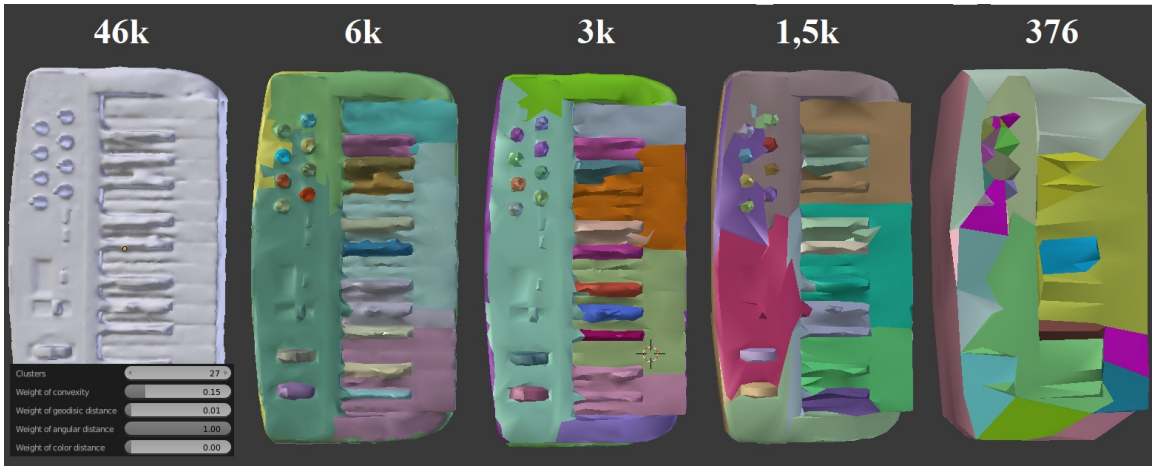


Figure 5.5: Automatic segmentation focused on angular distance over different LODs using 27 clusters.

From the above figures can be seen that the segmentation quality only starts to break down below the one thousand number of triangles. Of course the quality deterioration also depends on the model geometry and number of components, but since the used model already as a fair number of components and seems to work relatively well this is an indication that this solution will probably be able to be applied in most models of similar nature, with GUI like components, without being limited by the of their geometry density.

The Delta E 2000 color distance allowed to better segment the keyboard black keys, however it did not help detect some of the interactive components such as buttons or led display. This doesn't mean that the algorithm can't ever find them but unless the color contrast is high it is less likely to do so. Besides this, although this is not a problem with the particular used model, the decimate procedures may eliminate required geometry for such segmentations by: merging geometry outside the component with the component; or merging triangles of different colors that when viewed as one change the average color significantly.

It was observed that the buttons' shadows, which were not removed, had an impact on a color focused segmentation. When comparing a color distance focused approach with a angular distance focused approach with different numbers of segments can be seen that the first better segments the back keys and that the second better segments the buttons. This can be seen in the following figures, 5.6 and 5.7.

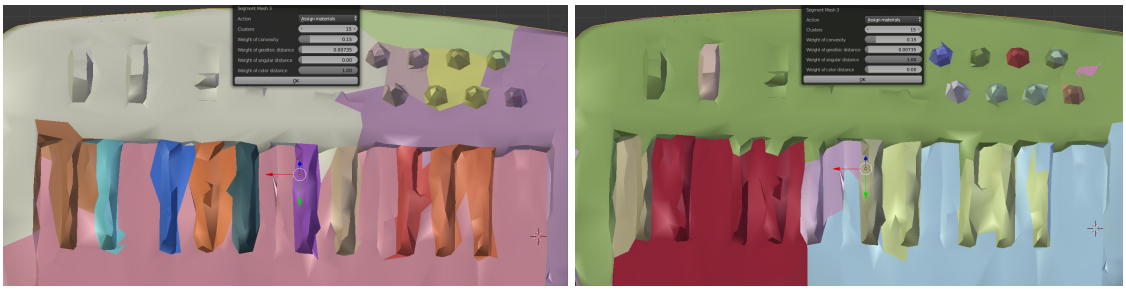


Figure 5.6: Comparison between a segmentation using color distance,at the left, and angular distance, at the right, using 15 clusters.

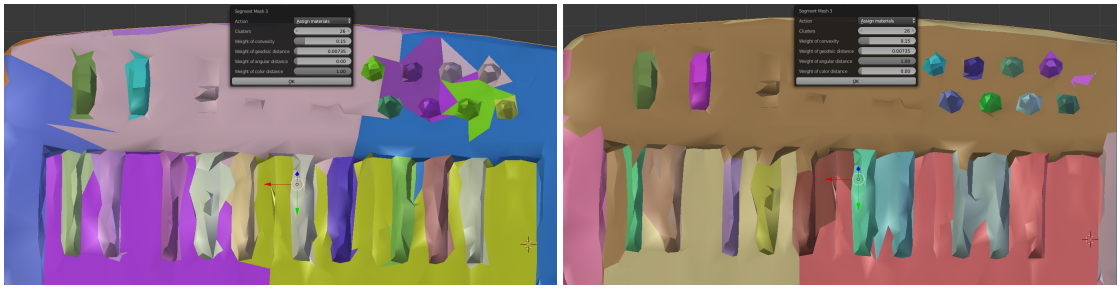


Figure 5.7: Comparison between a segmentation using color distance,at the left, and angular distance, at the right, using 26 clusters.

Unfortunately, combining the angular and color distances doesn't guarantee that the result will yield the best of both approaches. Sometimes it helps identify new important components without losing important segmentations, other times it loses important segmentations from both approaches. This is highly dependent on the model's geometry and texture.

Chapter 6

Conclusions and Future Work

The proposed overflow specification tackles all the points listed in the objectives 1.2 to some degree with exception of the point 1.6. This last point is explored in the SOTAR and was left incomplete in the prototype, which allows the usage of abstractions such as VRTK but doesn't fully integrate any with the rest of the prototype. Some topics could have received more attention and are later mentioned as possible future work in 6.2.

Regarding the Midi keyboard integration, the model was almost completely integrated using the proposed workflow, missing some geometry and material adjustments and interactivity tests in a VE.

The usability of the prototype was evaluated as being somewhat good for the users with no modeling experience which indicates that the workflow has some potential for this group of users.

The feedback of the modelers group indicates that the workflow and prototype require more improvements to offer clear advantages over a traditional flow.

The following section6.1 and 6.2 are dedicated to explicitly present this project's main contributions and possible future work respectively.

6.1 Contributions

This dissertation provides the following contributions:

- Specifying an initial alternative conceptual workflow for 3D digitalized models integration.

- Provide an indication, via the prototype's usability tests, that the proposed workflow is not unusable and, until proven otherwise, has the potential to be a complementary or alternative approach to more traditional workflows.

6.2 Future Work

6.2.1 Workflow and Prototype

The implementation of interaction related behaviors, or integration of already existing abstractions such as VRTK, is paramount in completing the workflow. The current validation only tackles the prototype and workflow usability. Only when the interactivity of the model is tested can the quality of completely integrated models be added to the equation in order to understand if the prototype and workflow are viable solutions.

Regarding the implemented prototype, its layout and usability can be heavily improved without changing the implemented workflow. The most relevant improvements were already mentioned in 5.1.6.

The implemented segmentation process can be tweaked and improved in a multitude of ways. Excluding the improvements to the prototype already mentioned, some of the possible major changes that could improve the implemented workflow are:

- **Enclosing volumes Segmentation**.

  This was the modelers preferable segmentation approach and it can be used to complement the current paint approach. The specific way through which they do this can vary.

  I conjecture the most useful implementation is one that provides immediate feedback, so after defining the volume the selected geometry should be painted immediately. This would also solve the problem of having to deal with the priority of operations of multiple enclosing volumes that intersect each other. However, calculating the geometry of interest later on can also be advantageous in some cases, for example, the user could configure how the enclosing volume uses to segment the geometry, this can be as simple as indicating that the normals directions of the model should be taken into account or as complicated as configuring the parameters of an automatic segmentation algorithm that runs locally.

- **Areas of interest for local automatic segmentation**.

  This would help to: avoid unwanted segments; speedup considerably the automatic segmentation process; improve results quality due to a less agnostic segmentation and by possibly removing the need to decimate the model; allow the algorithm to be used in models whose global geometry doesn't allow its usage.

  The way through which the user defines these areas can vary. They can be painted, defined by enclosing volumes, defined by already existing segmentations, and so on.

- **GrabCut like painting segmentation**.

  Something that was totally unexplored was the possibility of using a GrabCut like approach to the painting segmentation. Such approach, as illustrated in figure 3.3 in section 3.2, requires that the user loosely indicates the relevant geometry via drawing.

This approach has the potential to make the manual segmentation process more user friendly and faster. I conjecture that this might be doable by adapting the used algorithm. However this adaptation needs to be different than the adaption presented in 4.3.3. Unlike the color distance, the user painted areas are labels, which without any modification are useless as a feature. They can be blurred, using fuzzy labels instead of the originals, to be used as a feature. Alternatively, labels could be used to distort the geometry in order to reduce other features' distances locally, this requires a "background" label which would not apply any transformations to the mesh.

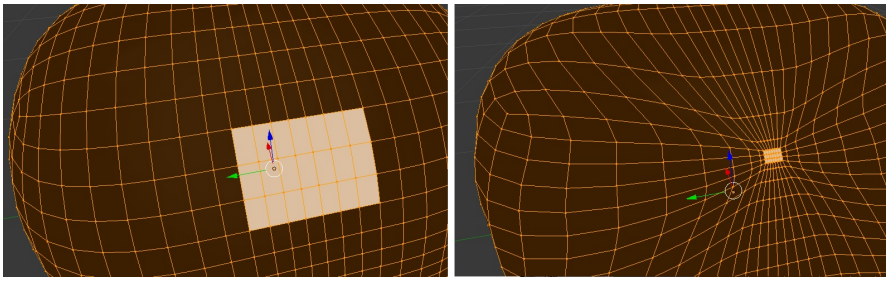


Figure 6.1: Example of geometry distortion. The area surrounding the painted area is shrinked reducing the geodesic distance of near faces.

Other possible improvements to the implemented workflow are in some form already mentioned in the chapter 3. Some of these are:

- **Material related procedures.**

  This was left out the conceptual workflow and is an important part in making the models look realistic.

- **Non disjoint groupings.**

  Despite the questionnaires results not indicating the need for such a feature. Its usefulness in more complete, and therefore complex, workflow implementations remains untested.

- **Automatic LODs.**

  A simple feature for automatic creation of LODs would be trivial to implement in the current prototype.

  The flow of operations is not yet conceptualized for more complicated cases, such as an element that disappears at some far away distance that requires not only that its geometry is not drawn but also that certain behaviors stop, or continue, to occur. E.g., a component might emit light, but at a certain distance the light is baked and replaced with a texture.

- **Automatic Retopology.**

  An important step in order to uniform same type components geometry and produce uniform quality results. Limitations in this automation process may possibly be major problems

in the proposed workflow because the integrated model quality may be compromised, thus requiring the usage of external software and manual labor. If this happens frequently and if the workflow is not flexible enough to accommodate this kind of updates then the proposed workflow creates an unnecessary overhead in the integration process, being its usage less justifiable over more traditional approach. For example, if by manually segmenting a model's components, using the proposed workflow, the results are visually unsatisfactory then all the segmentation work is wasted labor that wouldn't be present in a traditional approach.

- **Automatic property configuration.**

  Not imperative but allows the user to skip some procedures, thus making the workflow more quick and pleasant to use.

- **Documentation via annotations.**

  A basic annotation system would be trivial to implement over the current prototype. To conceptualize a more complete annotation system, which can organize annotations and has its own visual language regarding their priority and nature an extensive knowledge and awareness of digitalized models integration processes is needed. Questions such as: what symbols, colors, or labels do users associate with certain tasks? These would need to be object of study.

6.2.2 Future Validation Attempts

The presented validation in not very informative, and the keylogger logs ended up not being used. I would suggest that in future similar projects with similar validation objectives, to avoid the problems mentioned in 5.1.2, either use a large number user samples to make up for unusable samples or define a very rigorous test protocol or run a preparation test with the users before the actual test so that the users can get the gist of the workflow and tools. The hardware used for the test should also be taken into consideration to avoid situations were PC or Mac users execute additional interactions as a consequence of being unfamiliar with the keyboard layout or shortcuts.

Additionally, some feedback received by the modelers foreshadows that one possible advantage of the proposed approach might not be that is more efficient, fast or intuitive but that because it allows a different approach to the modeling process it can potentially lead to a different modeling experience and, consequently, have an impact on the creative process. However, at the project current state, there is nothing that can measure it's impact on the creative process. The most concrete feedback obtained from one of the modelers which seems to be linked to this quality is regarding the grouping system, which could, according to the user, allow the creation of variety easily. If such conjecture is true, then, by allowing a better exploration of the many acceptable solutions, the proposed workflow could indeed have an impact on the creative process. Although everything explained in these paragraph remain very esoteric and in the realm of conjectures, such

feedback suggests that the workflow and its concise implementation may require an additional type of evaluation that was not taken into account in the presented validation.

Bibliography

ADAM Technology. 2008. “Laser Scanning vs Digital Photogrammetry Introduction”: 1–8.

“Substance B2M - IMAGE-TO-MATERIAL GENERATOR.” 2018. allegorithmic. Accessed February 6. https://www.allegorithmic.com/products/bitmap2material.

Attene, Marco, Francesco Robbiano, Michela Spagnuolo, and Bianca Falcidieno. 2009. “Characterization of 3D shape parts for semantic annotation.” *CAD Computer Aided Design* 41 (10): 756–763. ISSN: 0010-4485. doi:10.1016/j.cad.2009.01.003.

Azzam, Joseph. 2016. “The Workflows of Creating Game Ready Textures and Assets using Photogrammetry.” Gamasutra. Accessed February 6, 2018. https://www.gamasutra.com/blogs/JosephAzzam/20160824/278719/The%7B%5C_%7DWorkflows%7B%5C_%7Dof%7B%5C_%7DCreating%7B%5C_%7DGame_Ready%7B%5C_%7DTextures%7B%5C_%7Dand%7B%5C_%7DAssets%7B%5C_%7Dusing%7B%5C_%7DPhotogrammetry.php.

Bangor, Aaron, Philip Kortum, and James Miller. 2009. “Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale.” *J. Usability Studies* (Bloomingdale, IL) 4, no. 3 (May): 114–123. ISSN: 1931-3357. http://dl.acm.org/citation.cfm?id=2835587.2835589.

Bhatia, Sanjiv. 2014. “Creating Large Isotropic Textures Using Image Quilting,” no. January 2004.

“Blender.” 2018. Blender. Accessed February 6. https://www.blender.org/.

“Blender - About.” 2018. Accessed July 1. https://www.blender.org/about/.

Bowman, Doug A., and Larry F. Hodges. 1997. “An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments.” In *Proceedings of the 1997 Symposium on Interactive 3D Graphics,* 35–38. I3D ’97. Providence, Rhode Island, USA: ACM. ISBN: 0-89791-884-3. doi:10.1145/253284.253301.

Brooke, John, et al. 1996. “SUS-A quick and dirty usability scale.” *Usability evaluation in industry* 189 (194): 4–7.

Brooke, John. 2013. “SUS: a retrospective.” *Journal of usability studies* 8 (2): 29–40.

BIBLIOGRAPHY

Brown, Steven W. 2008. *INTERACTIVE PART SELECTION FOR MESH AND POINT MODELS USING HIERARCHICAL GRAPH-CUT PARTITIONING.* PhD diss., Brigham Young University.

Celaya-Padilla, Jose M., Carlos E.T. Galvan, J. Ruben C. Delgado, Issac Galvan-Tejada, and Ernesto Ivan Sandoval. 2012. "Multi-seed texture synthesis to fast image patching." *Procedia Engineering* 35 (August 2014): 210–216. ISSN: 1877-7058. doi:10.1016/j.proeng.2012.04.182.

Cho, I., and Z. Wartell. 2015. "Evaluation of a bimanual simultaneous 7DOF interaction technique in virtual environments," 133–136. Piscataway, NJ, USA. doi:10.1109/3DUI.2015.7131738.

Denning, Peter J., and Ted G. Lewis. 2017. "Exponential Laws of Computing Growth." *Communications of the ACM* 60. ISSN: 0001-0782. doi:10.1145/2976758.

DICE. 2016. "Photogrammetry and 'Star Wars Battlefront'." GDC. Accessed February 1, 2018. https://www.gdcvault.com/play/1023272/Photogrammetry-and-Star-Wars-Battlefront.

Efros, Alexei A., and William T. Freeman. 2001. "Image quilting for texture synthesis and transfer." *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01:* 341–346. ISSN: 0013-4694. doi:10.1145/383259.383296.

Ermi, Laura, and Frans Mäyrä. 2005. "Fundamental components of the gameplay experience: Analysing immersion." *Worlds in play: International perspectives on digital games research* 37 (2): 37–53.

Fernández, Rodrigo. 2018. "Codeartist.mx - TEXTURE PAINTING." Accessed June 21. http://codeartist.mx/tutorials/dynamic-texture-painting/.

Flotyński, Jakub, and Krzysztof Walczak. 2017. "Ontology-Based Representation and Modelling of Synthetic 3D Content: A State-of-the-Art Review." *Computer Graphics Forum* 00 (00): 1–23. ISSN: 1467-8659. doi:10.1111/cgf.13083.

Foster, Shaun, and David Halbstein. 2014. *Integrating 3D Modeling, Photogrammetry and Design.* ISBN: 978-1-4471-6329-9. doi:10.1007/978-1-4471-6329-9.

"OBJ Files - A 3D Object Format." 2018. FSU - Department of Scientific Computing. Accessed February 8. http://people.sc.fsu.edu/~jburkardt/data/obj/.

Furukawa, Y., B. Curless, S. M. Seitz, and R. Szeliski. 2009. "Manhattan-world stereo." In *2009 IEEE Conference on Computer Vision and Pattern Recognition,* 1422–1429. June. doi:10.1109/CVPR.2009.5206867.

Furukawa, Yasutaka, and Carlos Hernández. 2015. "Multi-View Stereo: A Tutorial." *Foundations and Trends® in Computer Graphics and Vision* 9 (1-2): 1–148. ISSN: 1572-2740. doi:10.1561/0600000052.

Gaurav, Sharma, Wu Wencheng, and Dalal Edul N. n.d. "The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations." *Color Research & Application* 30 (1): 21–30. doi:10.1002/col.20070.

Gordon, Calleja. 2007. "Revising Immersion: A Conceptual Model for the Analysis of Digital Game Involvement." In *DiGRA '07 - Proceedings of the 2007 DiGRA International Conference: Situated Play*. The University of Tokyo, September. http://www.digra.org/wp-content/uploads/digital-library/07312.10496.pdf.

Huiberts, Sander. 2010. "Captivating Sound: The Role of Audio for Immersion in Computer Games," no. November: 1–200.

Jacobs, J., and B. Froehlich. 2011. "A soft hand model for physically-based manipulation of virtual objects," 11–18. Piscataway, NJ, USA. doi:10.1109/VR.2011.5759430.

Jankowski, Jacek, and Martin Hachet. 2013. "A Survey of Interaction Techniques for Interactive 3D Environments." *Eurographics2*. doi:10.2312/conf/EG2013/stars/065-093.

Jerald, Jason, Joseph J. LaViola Jr., and Richard Marks. 2017. "VR Interactions." In *ACM SIGGRAPH 2017 Courses*, 19:1–19:105. SIGGRAPH '17. Los Angeles, California: ACM. ISBN: 978-1-4503-5014-3. doi:10.1145/3084873.3084900.

Jover, Cyril. 2017. "Unity - Photogrammetry Workflow."

Kolecka, N. 2011. "Photo-Based 3D Scanning Vs . Laser Scanning – Competitive Data Acquisition Methods for Digital Terrain Modelling of Steep." *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XXXVIII (June): 203–208. ISSN: 1682-1777. doi:10.5194/isprsarchives-XXXVIII-4-W19-203-2011.

Kry, Paul, Adeline Pihuit, Adrien Bernhardt, Marie-paule Cani, Paul Kry, Adeline Pihuit, Adrien Bernhardt, et al. 2008. "HandNavigator : Hands-on Interaction for Desktop Virtual Reality." doi:10.1145/1450579.1450591.

kugelrund. 2013. "Mesh Segmentation Blender Addon." December 21. Accessed June 13, 2018. https://github.com/kugelrund/mesh_segmentation.

Lane, Jeff, Bob Magedson, and Mike Rarick. 1984. "An efficient point in polyhedron algorithm." *Computer Vision, Graphics, and Image Processing* 26 (1): 118–125. ISSN: 0734-189X. doi:10.1016/0734-189X(84)90133-6.

Li, Jing, and Wencheng Wang. 2017. "Fast and robust GPU-based point-in-polyhedron determination." *Computer-Aided Design* 87:20–28. ISSN: 0010-4485. doi:10.1016/j.cad.2017.02.001.

Liu, Jianfei, Y.Q. Chen, José M. Maisog, and George Luta. 2010. "A new point containment test algorithm based on preprocessing and determining triangles." *Computer-Aided Design* 42 (12): 1143–1150. ISSN: 0010-4485. doi:10.1016/j.cad.2010.08.002.

Liu, Rong, and Hao Zhang. 2004. "Segmentation of 3D meshes through spectral clustering." In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.* 298–305. October. doi:10.1109/PCCGA.2004.1348360.

Martín-Gutiérrez, Jorge, Carlos Efrén Mora, Beatriz Añorbe-Díaz, and Antonio González-Marrero. 2017. "Virtual technologies trends in education." *Eurasia Journal of Mathematics, Science and Technology Education* 13 (2): 469–486. ISSN: 1305-8223. doi:10.12973/eurasia.2017.00626a.

"MeshLab - Features." 2018. MeshLab. Accessed February 6. http://www.meshlab.net/#features.

Milgram, Paul, and Fumio Kishino. 1994. "A taxonomy of mixed reality visual displays." *IEICE TRANSACTIONS on Information and Systems* 77 (12): 1321–1329.

"Mira Tools." 2018. Accessed February 6. https://github.com/mifth/mifthtools/wiki/Mira-Tools.

Mokrzycki, Wojciech, and Maciej Tatol. 2011. "Color difference Delta E - A survey," 20 (April): 383–411.

Nickolls, John, and William J. Dally. 2010. "The GPU computing era." *IEEE Micro* 30 (2): 56–69. ISSN: 0272-1732. doi:10.1109/MM.2010.41.

Oh, Min. 2015. "Imperfection for Perfection." Unreal. Accessed February 6, 2018. https://www.unrealengine.com/en-US/blog/imperfection-for-perfection.

"PhotoModeler How it works." 2017. Accessed January 27. http://www.photomodeler.com/products/how-it-works.html.

Pierce, Jeffrey S., Brian C. Stearns, and Randy Pausch. 1999. "Voodoo Dolls: Seamless Interaction at Multiple Scales in Virtual Environments." In *Proceedings of the 1999 Symposium on Interactive 3D Graphics,* 141–145. I3D '99. Atlanta, Georgia, USA: ACM. ISBN: 1-58113-082-1. doi:10.1145/300523.300540.

Pierce, J.S., A. Forsberg, M.J. Conway, Seung Hong, R. Zeleznik, and M.R. Mine. 1997. "Image plane interaction techniques in 3D immersive environments," 39–43. New York, NY, USA. doi:10.1145/253284.253303.

Pietroni, Nico, Marco Tarini, and Paolo Cignoni. 2010. "Almost isometric mesh parameterization through abstract domains." *IEEE Transactions on Visualization and Computer Graphics* 16 (4): 621–635.

Qin, Feiwei, Shuming Gao, Xiaoling Yang, Ming Li, and Jing Bai. 2016. "An ontology-based semantic retrieval approach for heterogeneous 3D CAD models." *Advanced Engineering Informatics* 30 (4): 751–768. ISSN: 1474-0346. doi:10.1016/j.aei.2016.10.001.

Raad, Lara, and Bruno Galerne. 2017. "Efros and Freeman Image Quilting Algorithm for Texture Synthesis." *Image Processing On Line* 7:1–22. ISSN: 2105-1232. doi:10.5201/ipol.2017.171.

Radosevic, Goran. 2010. "Laser Scanning Versus Photogrammetry Combined with Manual Post-modeling in Stecak Digitization." *14th Central European Seminar on Computer Graphics (CESCG 2010):* 167–174.

Rautaray, Siddharth S., and Anupam Agrawal. 2012. "Vision based hand gesture recognition for human computer interaction: a survey." *Artificial Intelligence Review* 43 (1): 1–54. ISSN: 1573-7462. doi:10.1007/s10462-012-9356-9.

Simeone, A.L. 2016. "Indirect touch manipulation for interaction with stereoscopic displays," 13–22. Piscataway, NJ, USA. doi:10.1109/3DUI.2016.7460025.

Singh, R. Vikram Pratap, and Anoop M. Namboodiri. 2012. "Efficient texture mapping by homogeneous patch discovery." *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing - ICVGIP '12:* 1–8. doi:10.1145/2425333.2425370.

Skarlatos, D, and S Kiparissi. 2012. "Comparison of Laser Scanning, Photogrammetry and Sfm-Mvs Pipeline Applied in Structures and Artificial Surfaces." *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* I-3 (September): 299–304. ISSN: 2194-9050. doi:10.5194/isprsannals-I-3-299-2012.

Stavrou, Pavlos, Pavlos Mavridis, Georgios Papaioannou, Georgios Passalis, and Theoharis Theoharis. 2006. "3D Object Repair Using 2D Algorithms." In *Computational Science – ICCS 2006,* edited by Vassil N. Alexandrov, Geert Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, 271–278. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-34382-0. doi:10.1007/11758525_36.

Talvas, A., M. Marchal, and A. Lecuyer. 2013. "The god-finger method for improving 3D interaction with virtual objects through simulation of contact area," 111–114. Piscataway, NJ, USA. doi:10.1109/3DUI.2013.6550206.

Tullis, Thomas S, and Jacqueline N Stetson. 2004. "A comparison of questionnaires for assessing website usability." In *Usability professional association conference,* 1–12.

Turk, Greg. 2001. "Texture synthesis on surfaces." *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01,* no. August: 347–354. ISSN: 1686-4360. doi:10.1145/383259.383297.

"Unity - Features." 2018. Accessed July 1. https://unity3d.com/unity.

Von Luxburg, Ulrike. 2007. "A tutorial on spectral clustering." *Statistics and computing* 17 (4): 395–416.

"VRTK - Virtual Reality Toolkit." 2018. Accessed July 1. https://assetstore.unity.com/packages/tools/vrtk-virtual-reality-toolkit-vr-toolkit-64131.

Wei, Li-Yi, Jianwei Han, Kun Zhou, Hujun Bao, Baining Guo, and Heung-Yeung Shum. 2008. "Inverse Texture Synthesis." *ACM Trans. Graph.* (New York, NY, USA) 27, no. 3 (August): 52:1–52:9. ISSN: 0730-0301. doi:10.1145/1360612.1360651.

Wei, Li-Yi, and Marc Levoy. 2001. "Texture synthesis over arbitrary manifold surfaces." *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01,* no. August: 355–360. doi:10.1145/383259.383298.

Zalesny, Alexey, Dominik Auf der Maur, and Luc Van Gool. 2001. "Composite textures - emulating building materials and vegetation for 3D models." *Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage - VAST '01:* 179–186. doi:10.1145/584993.585022.

Zeltzer, David. 1992. "Autonomy, Interaction, and Presence." *Presence: Teleoperators and Virtual Environments* 1 (1): 127–132. doi:10.1162/pres.1992.1.1.127.

Zhang, Jingdan, Kun Zhou, Luiz Velho, Baining Guo, and Heung-Yeung Shum. 2003. "Synthesis of progressively-variant textures on arbitrary surfaces." *ACM Transactions on Graphics* 22 (3): 295–302. ISSN: 0730-0301. doi:10.1145/882262.882266.

Chapter 7

Appendix

7.1 External Scripts XSD

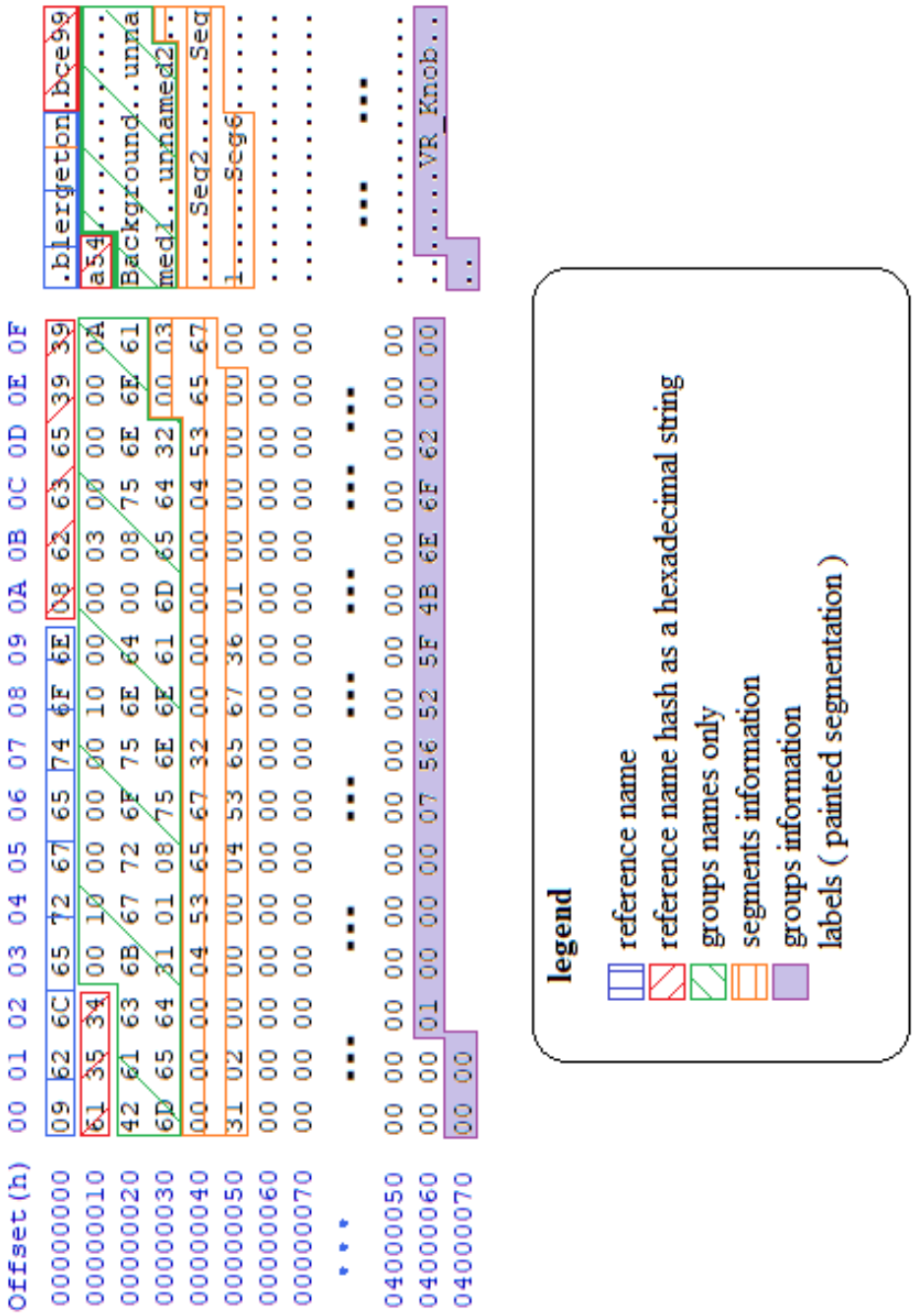
```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- Arg Complex Types -->
  <xs:complexType name="Empty">
    <xs:sequence/>
  </xs:complexType>

  <xs:complexType name="DecimalBounds">
    <xs:sequence>
      <xs:element name="lowerBound" type="xs:decimal" />
      <xs:element name="upperBound" type="xs:decimal" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="IntegerBounds">
    <xs:sequence>
      <xs:element name="lowerBound" type="xs:int" />
      <xs:element name="upperBound" type="xs:int" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="File">
    <xs:sequence>
      <xs:element name="extension" type="xs:string" maxOccurs="unbounded"
      minOccurs="0"/>
```

```
  </xs:sequence>
</xs:complexType>

<!-- Main Complex Types -->
<xs:complexType name="arg">
  <xs:sequence>
    <!-- TODO: create regex to validate argument name -->
    <xs:element type="xs:string" name="name"/>
    <xs:element name="type" >
      <xs:complexType>
        <xs:choice>
          <xs:element minOccurs="0" name="Boolean" type="Empty" />
          <xs:element minOccurs="0" name="UnboundedFloat" type="Empty" />
          <xs:element minOccurs="0" name="BoundedFloat" type="DecimalBounds" />
          <xs:element minOccurs="0" name="UnboundedInteger" type="Empty" />
          <xs:element minOccurs="0" name="BoundedInteger" type="IntegerBounds" />
          <xs:element minOccurs="0" name=" Vector2" type="Empty" />
          <xs:element minOccurs="0" name=" Vector3" type="Empty" />
          <xs:element minOccurs="0" name=" Vector4" type="Empty" />
          <xs:element minOccurs="0" name=" String" type="Empty" />
          <xs:element minOccurs="0" name=" Folder" type="Empty" />
          <xs:element minOccurs="0" name=" File" type="File" />
        </xs:choice>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" type="xs:string" name="default"/>
    <xs:element minOccurs="0" type="xs:string" name="description"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="script">
  <xs:sequence>
    <xs:element type="xs:string" name="name"/>
    <xs:element type="xs:string" name="tool"/>
    <xs:element type="xs:string" name="dependencies" minOccurs="0" maxOccurs="1" />
    <xs:element name="args">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="arg" maxOccurs="unbounded" minOccurs="0" type="arg"/>
        </xs:sequence>
```

```
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <!-- Tree -->
  <xs:element name="root">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="blender"/>
        <xs:element type="xs:string" name="meshlab"/>
        <xs:element type="xs:string" name="autodeskmaya"/>
        <xs:element name="operations">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="operation" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:string" name="name"/>
                    <xs:element type="xs:string" name="description"/>
                    <xs:element name="scripts" >
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="script"
                          type="script" maxOccurs="unbounded" minOccurs="0"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Appendix

7.2 Segmentation Binary File Structure



7.3 Questionnaire for any user profile

Questionnaire for any user profile

English adaptation of the original Portuguese online questionnaire

1. What problems do you think might occur if this workflow were applied in a professional production context?

2. Compared to a traditional workflow what do you consider to be advantageous in the proposed flow?

3. Complementing the previous question, what do you consider to be disadvantageous?

4. Rate the following features, from 1 to 5, according to how important you think they are. 1 being not important and 5 being of utmost importance.

(if you are a modeler you may leave points 4, 5 and 6 blank)

4.1 Automatic Segmentation

4.2 Segmentation by Painting

4.3 Segmentation by Enclosing Volumes

4.4 Organize components into disjoint groups

4.5 Organize components into non-disjoint groups

4.6 Visual auxiliary guides and tools to edit interactive components

4.7 Automatic components' topology optimizations

5. Would you like to suggest additional features?

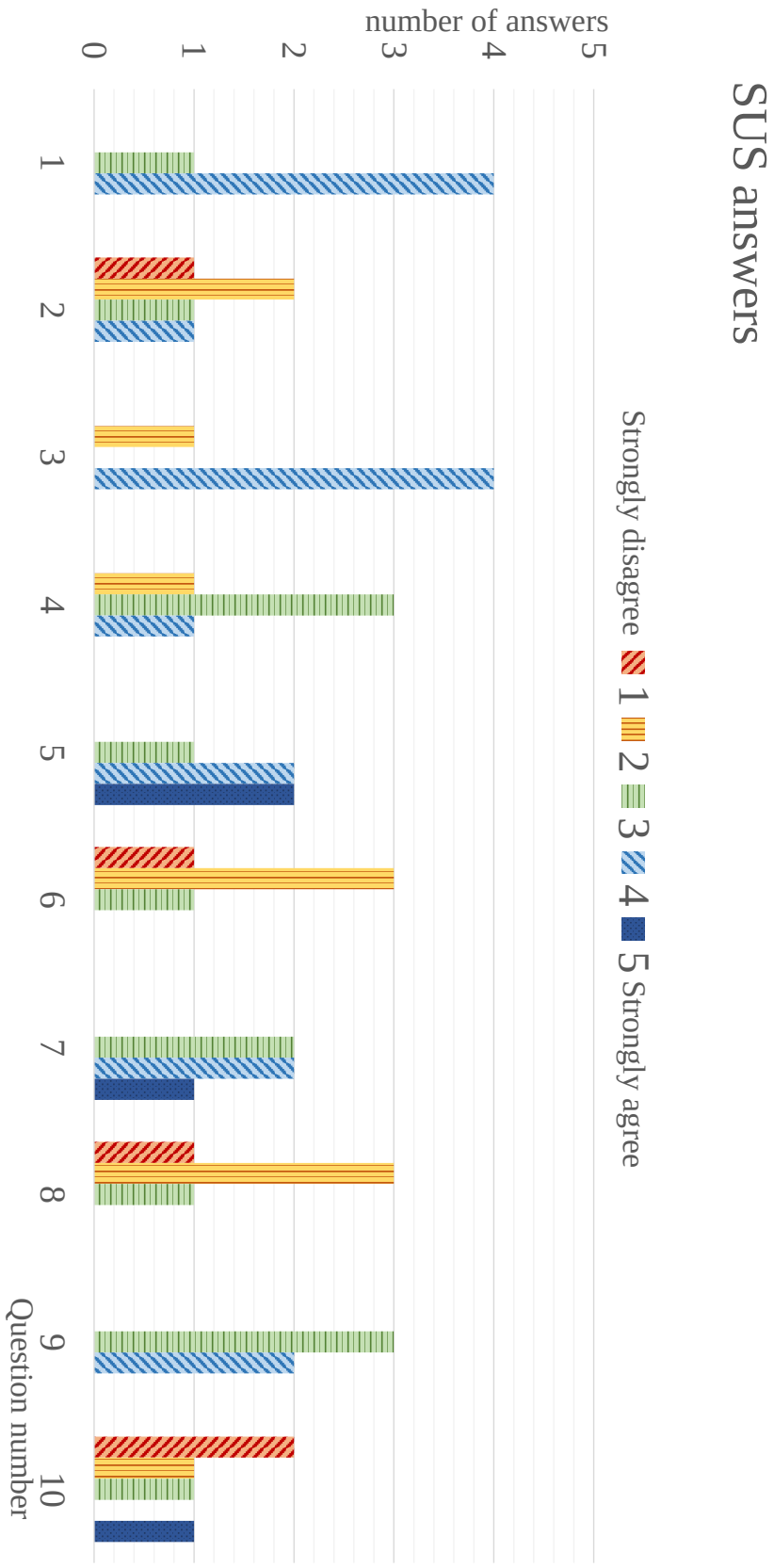
6. Do you think that the proposed workflow can complement or replace a more traditional approach?

No, anyway 1[ ] 2[ ] 3[ ] 4[ ] 5[ ] Yes, certainly

7.4 SUS Results

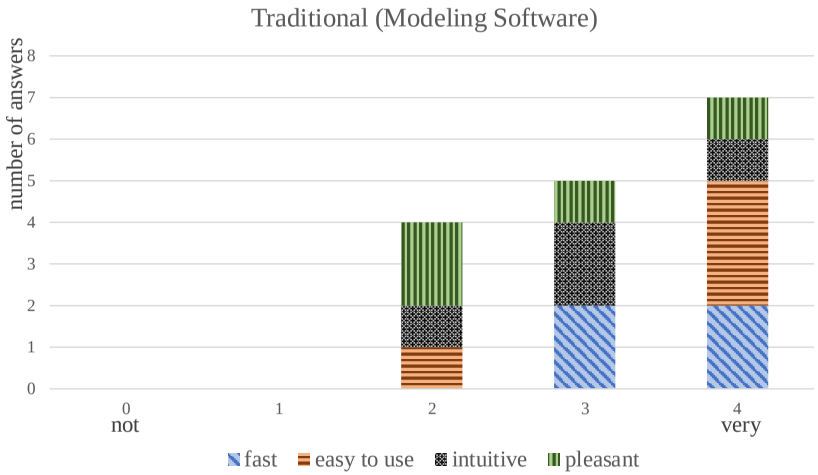
	Answers				
Questions	**1**	**2**	**3**	**4**	**5**
I think that I would like to use this system frequently	0	0	1	4	0
I found the system unnecessarily complex	1	2	1	1	0
I thought the system was easy to use	0	1	0	4	0
I think that I would need the support of a technical person to be able to use this system	0	1	3	1	0
I found the various functions in this system were well integrated	0	0	1	2	2
I thought there was too much inconsistency in this system	1	3	1	0	0
I would imagine that most people would learn to use this system very quickly	0	0	2	2	1
I found the system very cumbersome to use	1	3	1	0	0
I felt very confident using the system	0	0	3	2	0
I needed to learn a lot of things before I could get going with this system	2	1	1	0	1

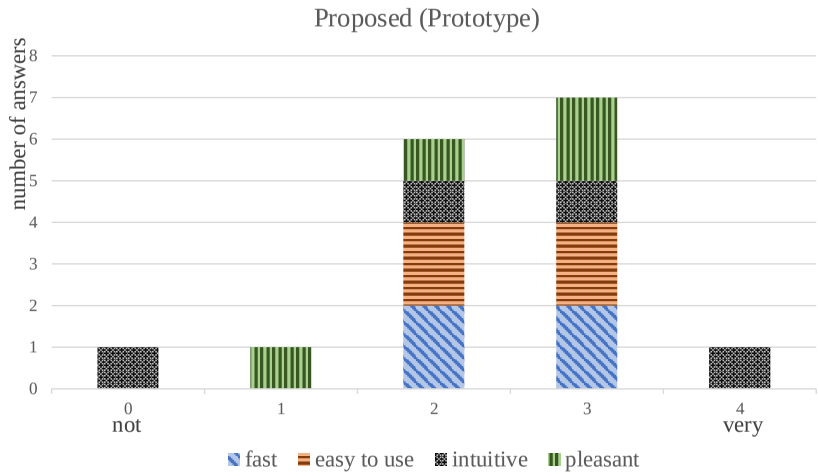
final usability score results:
mean 67,5 points (out of 100)
standard deviation:10,2 (rounded up to 1 decimal place)


SUS answers
number of answers
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
10
Question number
Strongly disagree
1
2
3
4
5
Strongly agree

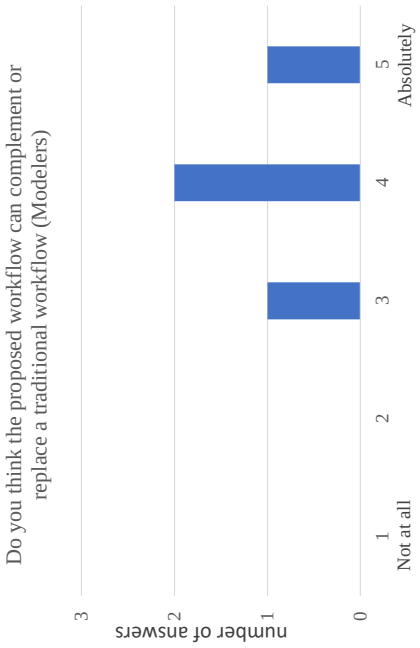
7.5 Questionnaire (Modelers Only) Results

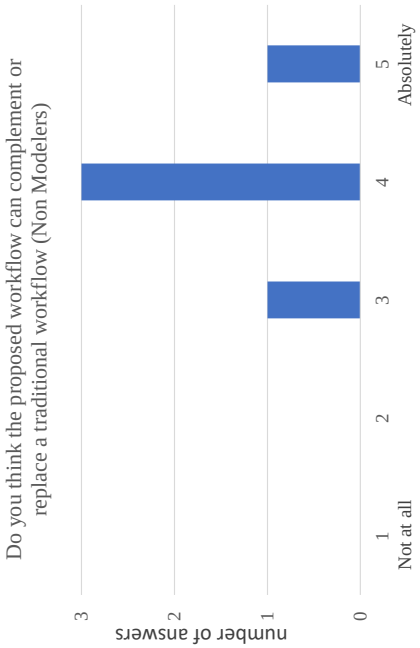
Users’ classification of both approaches regarding the given tasks (Modelers Only)

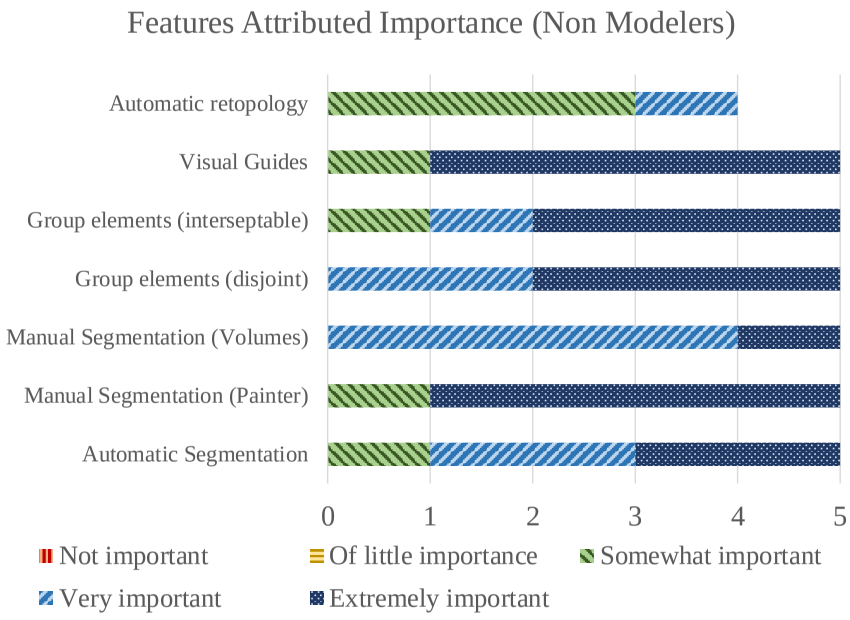


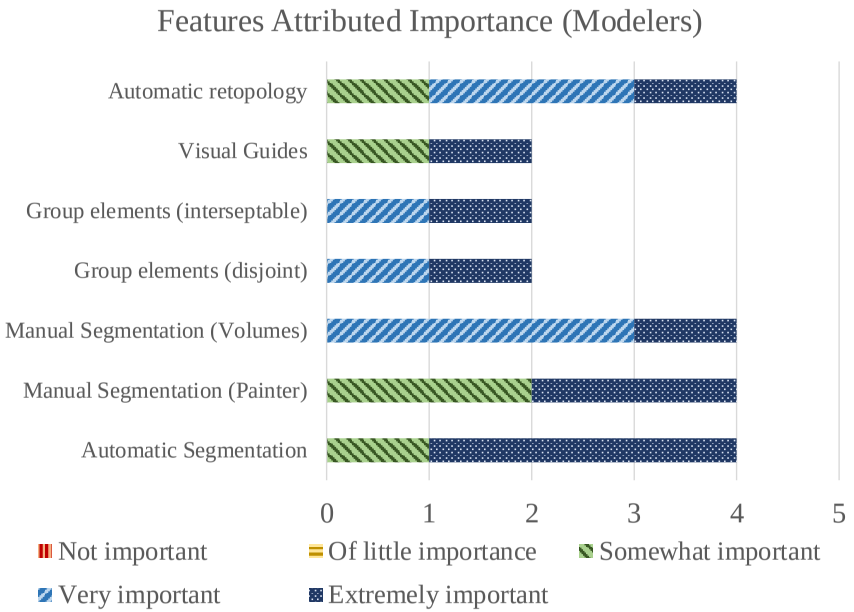


7.6 Questionnaire (Any User Profile) Questions 4 and 6 Results






Features Attributed Importance (Non Modelers)
Automatic retopology
Visual Guides
Group elements (interseptable)
Group elements (disjoint)
Manual Segmentation (Volumes)
Manual Segmentation (Painter)
Automatic Segmentation
0
1
2
3
4
5
Not important
Of little importance
Somewhat important
Very important
Extremely important


Features Attributed Importance (Modelers)
Automatic retopology
Visual Guides
Group elements (interseptable)
Group elements (disjoint)
Manual Segmentation (Volumes)
Manual Segmentation (Painter)
Automatic Segmentation
0
1
2
3
4
5
Not important
Of little importance
Somewhat important
Very important
Extremely important

7.7 Questionnaire (Any User Profile) Remaining Questions Results

Long answers were translated from Portuguese.

7.7.1 Question 1 - In Production Problems

What problems do you think might occur if this workflow were applied in a professional production context?

7.7.1.1 Modelers Answers

1. Process itself is a bit slow. It has a few more steps and a bit of guesswork, it's not very explicit. (sequences and variables)
2. Initial learning curve can make it difficult to stick to the tool.
3. Lack of control over topology. Need to use complementary software in most cases. Not intuitive.

7.7.1.2 Non Modelers Answers

1. Certain functionalities useful to the modeler normally present in the modeling tools might not be present in this workflow. It would involve using the modeling tool aside.
2. If there is not good documentation / tutorials, it could be complicated to solve simple problems. Issues such as delta> 0 should be documented or have a warning in the interface itself.

7.7.2 Question 2 - Workflow Advantages

Compared to a traditional workflow what do you consider to be advantageous in the proposed flow?

7.7.2.1 Modelers Answers

1. This method with some interface improvements is potentially faster than doing retopology by hand
2. New work process. As of now it doesn't accelerates existing workflows, but may allow you to create or make new forms ...
3. The possible integration with the various game-engines can facilitate its use and sometimes even replace tools external to the ecosystem of the respective engine.
4. Speed - Get variety quickly

7.7.2.2 Non Modelers Answers

1. More immediate response time. The tool allows to revisit areas that had already been "completed", allowing iteratively to improve the final result
2. ease of doing tasks on multiple objects in a single iteration
3. I consider it advantageous because it allows a more visual and interactive approach, simplifying and abstracting some procedures
4. It is more advantageous in that it is possible to work with a single tool (it is not necessary to switch working environments)

7.7.3 Question 3 - Workflow Disadvantages

Complementing the previous question, what do you consider to be disadvantageous?

7.7.3.1 Modelers Answers

1. If the tool can not produce accurate results, a search for the traditional method will be normal considering that in production the quality standards require an increased control.

7.7.3.2 Non Modelers Answers

1. I do not see significant disadvantages
2. Increases the complexity of the tool used

7.7.4 Question 5 - Suggestions

Would you like to suggest additional features?

1. in a first situation, it is difficult to perceive which elements we are interacting, needing to go to analyze the color of the elements that we want to select
2. The Usage Interface needs to be more mouse dependent. The shortcut keys should be moved a bit further so that keys are not pressed accidentally.
3. the tool is a bit abstract, it is necessary to work the parametric character of the tool adapting it to the visuality of the users of this type of tools. Previews could be useful.
4. By hovering over an area, a square could appear with the color referring to it.
   More informative tags in the options, for example, in the element group selection.
   An assistant in situations of ambiguous selection (in a limiting area).
5. More intuitive color choice. Visualization of groups (visually identify the elements belonging to each group)

6. Save data from automatic segmentation and make it non-destructive.
7. Easier selection of triangles
8. Undo

7.8 Original, Untranslated User Tests' Consent Forms

Formulário de Consentimento para Realização do Teste

Integração de Modelos 3D Dinâmicos Digitalizados em Ambientes Virtuais

Bruno Madeira, aluno do Mestrado Integrado em Informática e Computação (MIEIC) da Faculdade de Engenharia da Universidade do Porto (FEUP)

Para o teste a realizar é necessário registar todas as interações realizadas no computador. Para tal, é necessário o uso de um **Keylogger** e uma **gravação vídeo do ecrã do computador**.

Todos os dados recolhidos, independentemente do seu formato, não serão associados de alguma forma ao seu nome ou outra informação que poderiam ser usadas para o identificar. Os dados serão apenas acessíveis à equipa de investigação e usados para fins estatísticos e de avaliação qualitativa da solução proposta.

Os dados obtidos das análises realizadas poderão ser usados em documentos e apresentações relacionadas com o presente estudo. Estes documentos ou apresentações nunca terão presente o seu nome ou alguma informação que permita identificá-lo.

Ao assinar esta secção do formulário, estou a permitir que seja usado um **Keylogger** e uma **gravação de ecrã** durante a realização do meu teste.

Assinatura do Participante: _____
Data: _____

Formulário de Consentimento para Gravação Áudio e Vídeo

Integração de Modelos 3D Dinâmicos Digitalizados em Ambientes Virtuais

Bruno Madeira, aluno do Mestrado Integrado em Informática e Computação (MIEIC) da Faculdade de Engenharia da Universidade do Porto (FEUP)

Com o fim de poder realizar uma análise da usabilidade da solução proposta rigorosa, pretende-se realizar uma **gravação de áudio e vídeo**, através do microfone e da câmara do computador onde o teste é realizado, do utilizador durante a realização do teste.

Todos os dados recolhidos, independentemente do seu formato, não serão associados de alguma forma ao seu nome ou outra informação que poderiam ser usadas para o identificar. Os dados serão apenas acessíveis à equipa de investigação e usados para fins estatísticos e de avaliação qualitativa da solução proposta.

Após a análise dos dados todas as gravações vídeo e áudio relativas aos testes serão eliminadas. (antes do dia 26 de julho de 2018)

Da análise qualitativa poderão ser transcritas ou citadas reações observadas nos testes a ser usadas em documentos e apresentações relacionadas com o presente estudo. Estes documentos ou apresentações nunca terão presente o seu nome ou alguma informação que permita identificá-lo (como uma imagem ou gravação de voz).

Ao assinar esta secção do formulário, estou a permitir que seja realizada uma **gravação áudio e vídeo de mim durante a realização do teste**. Estou também ciente que este consentimento é válido até à seguinte data: _____. Nesta data, ou antes, as **gravações de áudio/vídeo e qualquer duplicado destas** serão eliminados.

Assinatura do Participante: _____
Data: _____