VR viewer

Tobias Eilertsen

Master's thesis in Software Engineering at

Department of Computer science, Electrical engineering and Mathematical sciences,
Western Norway University of Applied Sciences

Department of Informatics,
University of Bergen

February 2022

Western Norway
University of
Applied Sciences

Abstract

Acknowledgements

First and foremost, I would like to thank

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is written as part of the University of Bergen and Western University of Applied Sciences master programme in software engineering.

1.1 Context and Approach

A CT scan is often performed when a hospital receives an injured patient considered for orthopedic surgery. The CT scan is displayed as 2D images or a 3D model on a computer. The model helps the medical personnel plan the surgery by understanding the anatomy of the fracture. If a surgeon has a good understanding of the anatomy related to the fracture, the surgery has less risk of complications or could require fewer resources.

1.2 Problem Description

Visualizing the model in 2D limits the understanding of the fracture because of the lack of scale and depth. The problem applies to both 2D images and 3D models rendered in 3D. A considered solution is visualizing the model in Augmented Reality or Virtual Reality to give medical personnel a good understanding of the problem area. VR has multiple potential benefits, and the surgery plannkkjing process can use some of these. As the users are already looking at 3D models on 2D screens, visualizing in VR could improve the surgeon's overview and patient safety. The entire planning process could also be more effective by removing or reducing the need for 3D printed models, especially in cases with limited time. Therefore the possible research questions are as follows:

How Can VR technology improve surgery planning by making the process safer or more effective? How Can VR technology give some of the same benefits as 3D printing gives today at a lower cost?

1.3 Methodology

The project should include a VR prototype of a standard where it is user friendly enough to test with non-technical subjects and with functionality that is comparable to the use cases of a printed model. The prototype is tested on medical personnel to measure any benefits on anatomical understanding and how it affects surgery planning and effectiveness. The application should be available for further development and study.

Firstly, a prototype VR viewer will be created with the help of related open-source frameworks/software and guidance from both orthopedic surgeons and developers with experience in medical technology.

In order to answer the research question, it is necessary to measure the performance of the final application. This thesis will use qualitative methods by interviewing related personnel to investigate the performance, including anatomical understanding, cooperation, and the effectiveness of the planning process. The performance will also be put in context to the existing solutions, possibly by making a direct comparison by using a 3D model, printed model, and VR viewer on the same case.

1.4 Contribution

1.5 Outline

Chapter 2

Background

This chapter will present some of the knowledge that our research is built upon.

2.1 orthopedic surgery

Orthopedic surgery is surgery involving the musculoskeletal system. Cases range from trauma surgery, where high impact forces cause injuries to infections and tumors[**manual ortho**].

Orthopedic surgeons do both elective (planned) and urgent surgery. In elective surgery, the surgeons will have days or weeks to plan out the surgery. A team of usually two surgeons will plan the surgery together. A surgeon will diagnose the patient from the following features: history, clinical examination, medical imaging, and any special investigations. History includes the patient's complaint and any previous injuries. Clinical investigation means examining the sources of the symptoms and the body as a whole. Medical imaging, including ultrasound, CT, and MRI, gives the surgeons a detailed insight into bones or soft tissue structures.

This research is based on elective surgeries on trauma or fractures where the surgeons plan out the surgery with different tools.

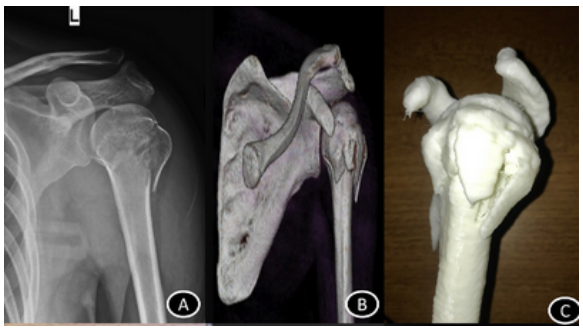
2.1.1 Visualisation of medical imaging

The output of both CT and MRI scans is a three-dimensional scan. The output is typically represented as slices, where a slice is a 2D picture repeating along an angle. The pixel at coordinate *(x, y)* at slice number z represent the absorption at the point *(x, y, z)*[2]. Each pixel in a slice represents a voxel in a volume, and all the slices combined make up volumetric data or a three-dimensional point cloud[2]. The slice thickness of a slice can be below 1 mm, giving a high-resolution scan where a single point is less than one cubic millimeter[6].

When rendering a 2D image, the volume must be projected to two-ddimensional space. The values in a single slice can be used to view the intersection at a specific slice number. A line of values along all the slices can be used to view

the intersection at another plane. All slices can be combined to show either the average or the maximum intensity[5]. The values are converted to greyscale by mapping ranges of values to greyscale pixels. This mapping function determines the brightness and contrast of the final image.

3D printing



(a)

Figure 2.1: Humeral (upper arm) fracture
A fracture from initial scan to finished print. (A) preoperative X-ray; (B) 3D model; (C) 3D printed model. [7]

An alternative to digital representation is to print the 3D model to inspect it physically[7]. 3D printing has many advantages, such as the surgeon physically holding the model, measuring the model, trying out equipment, and practicing with the model.

2.1.2 Unity

The Unity game engine is used for developing the application[**unity**]. Using a game engine speeds up the development as it includes systems needed, like rendering, animations, and physics. Unity is widely used for game development[**gameenginesonsteam**] and is well documented online with a lot of community-supported plugins.

Unity has good support for different VR and XR platforms[**unityxr**] including the OpenXR standard widely used by VR headsets[**openxr**]. This framework allows creating VR games with Unity handling most VR logic.

Part of the Unity VR support is the XR interaction Toolkit (XRTK)[**xrinteractiontoolkit**]. It is a high-level framework for using XR interactions with unity events. The toolkit also includes components for selecting/grabbing, haptic feedback, and UI interaction.

To develop for several XR platforms, the different controllers have a mutual interface for setting keybindings, called an XR controller. Almost all commercial VR controllers support an index finger trigger, a joystick/trackpad with 2D directional input, a grip button, and at least one extra button[8].

Chapter 3

Design and Implementation

The application was developed by me in about 7 months. I have no medical experience and was supported by staff at Helse vest with guidance and expertise on orthopedics, medical imaging and Unity, and VR development.

3.1 Demonstration

gameplay

3.2 Development methods

3.2.1 Iterative Development

The application was developed using iterative development and elements from design thinking and scrum. The team had biweekly meetings with a demonstration, sprint retrospect, and sprint planning. Sprint retrospect was not as relevant as only one person worked full time on the project. However, sprint planning was a valuable way of sharing interdisciplinary information and getting feedback from medical experts. The biweekly demo also served as smaller user testing sessions. The medical personnel in the team would try out the application and any new features and give continuous feedback during testing. Optimally the test subjects should have been personnel outside the project, but busy schedules and COVID restrictions limited this. After the biweekly demo, I updated the sprint backlog with updated tasks prioritized by value.

3.3 Project overview

3.4 Design

3.4.1 Design goals

The goal of the application design is as follows: give a good understanding of what the fracture looks like (bruddlinje og mindre biter) Ease of use for non

technical personnel Be able to cooperate in smaller teams - multiplayer

3.4.2 VR interface

To make the VR application easy to use, I focused primarily on familiarity with current tools and a simple design.

Familiarity with current tools means taking interactions from programs such as CT viewers and adopting the same interactions to VR to make it recognizable to the user. An example of this is navigating menus the same way one would navigate a popup menu with a pc mouse. All unnecessary interactions and information is removed to allow new users with little VR experience to start using the program quickly. The goal is to make the experience less overwhelming and lower the skill required to use the viewer effectively.

Learning process

A video game style tutorial was considered, where the user would go through tasks with hints or descriptions of the features in the application. However, this was not implemented as early testing revealed users were learning the controls quickly.

Every button has a description displayed as text above the button to remind the user of the keybindings. An image can optionally be displayed, showing the user the keybindings. The hints were implemented as users repeatedly forgot button controls in testing.

Grab interaction

The most used part of the interface is moving the fracture parts to inspect them. The grab should mimic how users move objects in real life to make this easy to use. One issue is that the user might want to move either the entire model or move a part of the model. Including both could introduce mode issues[**nngroup**], so the grab interaction always picks up a model part. A joystick input moves the entire model.

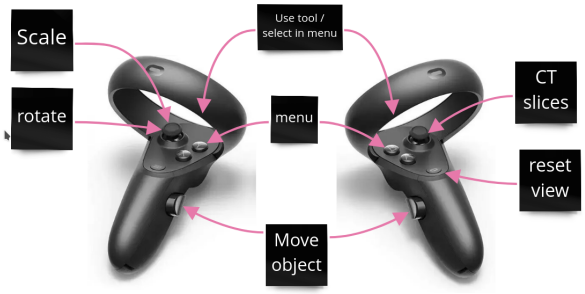
The grab functionality is implemented using Unity's built-in *Interactable* system for XR[9] development. This allows to set up grabbing, throwing, and more. By default, the grab will move the center of the interactable to the user's hand. The built-in grab action is cumbersome when precisely adjusting larger objects, as moving the interactable a small distance requires repositioning it from wherever the center is. To solve this, a script extending the interactable changes the position vectors

Separation mode

The grabbing functionality introduces a problem: If the user moves a model part, the CT images will no longer be similar to the model displayed. Also, if the user is measuring the distance between two model parts, it gives an invalid result if the parts are moved. The solution to this is a separation mode. Moving any model part enters separation mode and reverting to the original position exits separation mode. In separation mode, the CT images are hidden, and

measurements are hidden. A large button appears to display the current mode, prompting the user to click the button to reset the model and once again view CT images.

Keybindings



(a)

Figure 3.1: Keybindings on Quest controllers
This illustration is also used within the application.

The most common operations are bound to trigger and grip button, which is easily reachable on most controllers. The controller trigger mimics a mouse click on a pc, and so it is used for clicking buttons and using selected tools. The grip button mimics a real-world grabbing gesture and is used for grabbing a model part.

Menu system

A menu system is implemented to allow the user to navigate less common operations while not distracting from inspecting the model. When the menu button is pressed, the menu screen appears in front of the user. It is shown in world space, which means the text appears as a physical screen. The model is temporarily hidden to ensure the menu is always in the user's field of view. Placing content at the center of the screen avoids a typical VR UX problem where some information is not visible to the user.

Motion sickness

The most relevant motion sickness factor for this project is the use of motion and static content. The only moving content is the fracture model, which only moves because of player input. The environment is designed like an office room and is always static. All HUD elements are also static, with few exceptions. During multiplayer, the model may move caused by another player's input, which might cause some motion sickness if the model is viewed by the user. This could be solved by hiding the model when it moves, but it could interfere with the multiplayer experience if the model suddenly disappears.

3.4.3 Multiplayer design

The goal of the multiplayer feature is to allow two or more surgeons to cooperate effectively. Having one surgeon using a VR headset and another using a screen is likely interfering because the surgeons are seeing different models. The multiplayer component allows the surgeons to use the same tools while seeing the same model. Multiplayer should not make the application more difficult to use. All interactions should be similar to the usual experience, and cooperation should not require any extra steps, except connecting to the multiplayer server. Any important information should be available to all players, and irrelevant information (another player opening a menu) should not be disturbing.

3.5 DICOM data visualisation

I received several anonymized CT scan files from Helse Vest to use as input data. The CT scans are sent as DICOM packages[3], where the scan is stored as several files, each representing a slice. The model is also split up into bone fragments. The files include 3D surface models as STL files, where each file contains one bone. The STL files are created by (radiograf ???) at Haukeland University Hospital.

3.5.1 Rendering CT image

To render CT images from the DICOM data, I used the .NET library *Fellow Oak DICOM*[4]. It is an open-source library for parsing DICOM data and image rendering. I use the data the DICOM files to create a *Slice* object for each of the files. The slice object implements reading the density value at position (x, y) at that slice. a Unity Texture is created for each slice to render the image in Unity.

Every pixel on the texture is set to the greyscale color correlating to the density value. To find the color in a position, calculate $(density - minDensity)/(maxDensity - minDensity)$. If result is 1, it has maximum density and is set to white color.

3.5.2 Rendering bone fragments

In STL files all vertices are stored coordinates relative to the center, so by inserting all STL files at the same world space coordinate, the bone fragments will be placed correctly relative to each other. The STL files are exported to .obj files using blender and rendered with Unity.

Separating bone fragments

Adding some form of manual splitting of meshes was considered. This would allow the user to split the model pieces into separate pieces further.

An example open-source framework called Ezy-slice[**aryan**] was tested.

3.5.3 Rendering intersection

While rendering the selected slice as an image gave some understanding, there was still a disconnect between the model and the CT image. To improve upon this, I rendered the CT image at the exact position where it would intersect with the model. This required removing the part of the model above/under the intersection plane, so it did not obscure the image. For this, I used a Unity Assetstore shader to view the cross-section of a mesh[1].

This effect can be disturbing as the entire model is not visible. To allow for rendering the entire model, this rendering is a made optional. The option defaults to showing the entire model because it is easier to understand. The toggle button is shown in the CT image options.

3.5.4 implants

The menu has a sub-menu for choosing what implant to add to the scene. It consists of a list with a description and a preview of the implant. Pressing a button will spawn the selected implant in front of the user, and can then be moved in the same way a model part can be moved. To improve the surgeon's ability to line up screws and find what parts of the fracture is penetrated by the screw/object an outline is shown. This allows the user to see where an object is compared to any other parts, even if it is entirely obscured by the model. This is achieved by a shader using a depth buffer to only draw the outline when obscured by other meshes[**shader depth**].

3.6 performance

Some performance guidelines are outlined by the Occulus developers for the Quest 1[**performance**].

The recommended triangle count is between 350k-500k triangles. The imported meshes can have a high vertex count and are reduced manually using Blender remeshing. This could be automated, but this limit would be irrelevant using a headset using a GPU for dedicated rendering.

Chapter 4

analysis and assessment

4.1 tests

4.1.1 test 1

date: 16 jan. 2022 This was the first test done with professional end users. The goal of the test was to figure out if any of the proposed usecases was relevant for the user. The test was performed by introducing the software, its potential usecases and a short introduction on how to use the VR controllers. As both surgeons are very experienced with reading 2D CT images, they agreed that using 3D and VR to improve the understanding of fractures were minimal.

Occupation	Experience	proficiency with VR	comments
hline Orthopedic surgeon	10+ years	None	The density of bone is important to know parts are solid enough for drilling. Also, some parts below the desnity threshold are missing.
Orthopedic surgeon	10+ years	None	Smaller parts and bone fragments should be moveable to "reponere" brud-det.

Bibliography

[1] Abdullah Aldandarawy. *Unity Cross Section Shader Using Shader Graph*. en. Dec. 2019. URL: https://codeburst.io/unity-cross-section-shader-using-shader-graph-31c3fed0fa4f (visited on Jan. 27, 2022).

[2] Chougule, Mulay, and N. Ahuja. "Conversions of CT Scan Images into 3 D Point Cloud Data for the Development of 3 D Solid Model using B-Rep Scheme." en. In: *undefined* (2013). URL: https://www.semanticscholar.org/paper/Conversions-of-CT-Scan-Images-into-3-D-Point-Cloud-Chougule-Mulay/d4875163b4162ba44843a326490f20d6c7fa33e7 (visited on Dec. 1, 2021).

[3] *DICOM*. URL: https://www.dicomstandard.org/ (visited on Jan. 21, 2022).

[4] *Fellow Oak DICOM*. original-date: 2015-05-09T13:35:00Z. Feb. 2022. URL: https://github.com/fo-dicom/fo-dicom (visited on Feb. 11, 2022).

[5] Elliot K. Fishman et al. "Volume Rendering versus Maximum Intensity Projection in CT Angiography: What Works Best, When, and Why." In: *RadioGraphics* 26.3 (May 2006). Publisher: Radiological Society of North America, pp. 905–922. ISSN: 0271-5333. DOI: 10.1148/rg.263055186. URL: https://pubs.rsna.org/doi/10.1148/rg.263055186 (visited on Dec. 6, 2021).

[6] David L. Hamblen and A Hamish R.W. Simpson. *outline of orthopaedics*. 14. edition. 2010.

[7] Abhishek Mishra et al. "Virtual preoperative planning and 3D printing are valuable for the management of complex orthopaedic trauma." eng. In: *Chinese Journal of Traumatology = Zhonghua Chuang Shang Za Zhi* 22.6 (Dec. 2019), pp. 350–355. ISSN: 1008-1275. DOI: 10.1016/j.cjtee.2019.07.006.

[8] Unity Technologies. *Unity - Manual: Unity XR Input*. en. URL: https://docs.unity3d.com/Manual/xr_input.html (visited on Feb. 2, 2022).

[9] *XR Interaction Toolkit — XR Interaction Toolkit — 2.0.0-pre.7*. URL: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/index.html (visited on Feb. 7, 2022).