Michael Kampl, BSc.

ViveDTI
A program to visualise diffusion tensor imaging data on the HTC Vive virtual reality system

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Biomedical Engineering

eingereicht an der

Technischen Universität Graz

Betreuer
Assoc. Prof. Dipl.-Ing. Dr.techn. Reinhold Scherer

Institut für Neurotechnologie

Graz, Juni 2018

EIDESSTATTLICHE ERKLÄRUNG

AFFIDAVIT

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

26.06.2018

Datum / Date

Unterschrift / Signature

Abstract

Diffusion tensor imaging (DTI) is a non-invasive procedure to visualise nerve fibres of the brain. Due to the wide distribution of virtual reality (VR) systems they also get into focus of science to use for visualising data.

Subject of this work is the development of a software environment to visualise nerve fibres of the brain on a HTC Vive VR- System. The fibres are calculated from a functional magnetic resonance imaging (fMRI) record with existing software, afterwards the data is processed with a 3D- Graphicsuite. So, the focus of this work is the visualisation of the data using the game development platform Unreal Engine 4 and the VR- System.
The goal is an easy to use program which profits from the possibilities of VR in presentation as in operation. The program is controlled using the controllers of the HTC Vive respectively the user's movements.

Furthermore, the pros and cons of this program and the performance are evaluated.

Keywords: HTC Vive, Diffusion- Tensor- Imaging, Virtual Reality, Unreal Engine 4, Tractography

Kurzfassung

Diffusions- Tensor- Bildgebung (DTI) ist ein nicht invasives Verfahren um Nervenfasern des Gehirns darzustellen. Durch die weite Verbreitung von Virtual Reality (VR) Systemen rücken diese auch in den Focus der Wissenschaft um sie für die Visualisierung von Daten zu nutzen.

Gegenstand dieser Arbeit ist die Entwicklung einer Softwareumgebung zur Darstellung von Nervenfasern des Gehirns auf einem HTC Vive VR- System. Die Fasern werden mit bestehender Software aus einer Aufnahme einer funktionellen Magnetresonanztomographie (fMRI) errechnet, anschließend werden die Daten mittels einer 3D- Grafiksuite weiterverarbeitet. Der Fokus dieser Arbeit liegt also auf der Visualisierung der Daten mithilfe der Spielentwicklungsplattform Unreal Engine 4 und dem VR- System.
Ziel ist ein einfach zu benutzendes Programm, welches von den Möglichkeiten der VR sowohl in der Darstellung als auch der Bedienung profitiert. Die Steuerung des Programms erfolgt durch die der HTC Vive zugehörigen Controller bzw. durch die Bewegungen des Anwenders.

Außerdem werden die Vor- und Nachteile dieses Programms und die Performance beurteilt.

Schlüsselwörter: HTC Vive, Diffusions- Tensor- Bildgebung, Virtuelle Realität, Unreal Engine 4, Traktografie

Inhaltsverzeichnis

1. Introduction and problem

1.1. Motivation

Diffusion MRI is a technique that came into existence in the mid-1980s [1, 2] and diffusion tensor imaging (DTI) was formally introduced in 1994 [3, 4]. Since then the scope of application broadened. DTI is used for investigating diseases of the nervous system such as Alzheimer's disease [32, 33] or multiple sclerosis [34, 35]. Also, it is used to follow the healing process after injuries of the corticospinal tract [3] and to get a better understanding of the development of the brain [2].

DTI data can be visualised with tools like DSI Studio [6] or TrackVis [7] (Wang, Wedeen; Harvard, MIT; USA). They generate 3D models which can be exported or investigated directly in those programs. Furthermore, several projects exist which aim to view the 3D fibre models in virtual reality. In [8] a web-based 3D visual analytics tool was created to work with the Oculus Rift (Facebook Inc., Menlo Park, USA) while [5] uses a program, called realDTI, which was developed with the Unreal Engine to visualise the data on an Oculus Rift.

The program described in [5] gives the user, as the author states it, a good feeling for the fibres but also has its flaws. The author also writes that realDTI is computationally intensive and only up to 100 fibres can be displayed without major side-effects like lag and artefacts. Also, an outlook for further research is given in which an orientation system, implementing atlases which connect the fibres with cortex areas and improving the performance of the program is mentioned.

In the course of this thesis a similar program is being developed. The idea is to take the preliminary work and the findings from [5] to build an application without the flaws of the previous program and add functionalities like an orientation system and atlases. Furthermore, instead of the Oculus Rift virtual reality (VR) Headset a Vive VR system will be used.

This way a program should emerge which gives scientists and doctors a new opportunity for analysing DTI data and provides a way to give lay people a better understanding of the brain.

1.2. Background

Whereas conventional magnetic resonance imaging (cMRI) provides methods to map the anatomy or tissue volume, diffusion-weighted imaging (DWI) of random translational water molecules offers quantitative anisotropy and orientation information that has been utilized to map the integrity or architecture of the soft tissue in the central nervous system. [9]

Conventional magnetic resonance images of the brain are generated by applying a radio frequency pulse in a homogeneous magnetic field and measuring the resulting magnetisation of the hydrogen atoms in the water molecules. Diffusion weighted magnetic resonance imaging measures the diffusion of water in tissues by applying a gradient that increases the strength of the magnetic field evenly in one direction. Based on the physics of magnetic resonance imaging, the emitted signal decreases as a function of the movement of water molecules in the direction of this gradient; however, it is unaffected by the movement perpendicular to the direction of the gradient. The rate of water diffusion in a given direction is calculated by comparing the signal when diffusion weighting is applied to the signal at the same location when no diffusion weighting is applied. The rate of water diffusion in other directions can be measured by changing the direction of the gradient. [10]

By its nature, the axonal architecture in the white matter of the central nervous system promotes diffusion of water molecules in a direction predominantly parallel, rather than perpendicular, to axon fibres. [3] This means the diffusion in white matter tracts of the brain is anisotropic. In regions with few or no constraints imposed by physical boundaries, such as cerebral spinal fluid (CSF) in the ventricles, water movement is random and uniform in every direction and is therefore isotropic. The degree to which diffusion is anisotropic is represented by a fractional anisotropy (FA), which is a nondirectional proportion from 0.0 to 1.0. [10] Where a FA of 1.0 represents anisotropic and 0.0 isotropic diffusion. This allows to differentiate between various tissues.

1.3. Goal

As already mentioned in 1.1. the program realDTI [5] serves as a template for this thesis. This thesis should bring forth a program which is less demanding on the computer hardware so that more fibres can be displayed. For input and output a Vive virtual reality system should be used. Furthermore, the program should give the possibility to display fibres connected to

functional cortex regions and fibres originating from specific brain areas. An orientation system is to be implemented so that the user can determine his current position in the brain. Without such a system, it is easy to get lost when the point of view is deep inside the fibre structure.

To sum up, the new program should:

- Be less demanding on the computer hardware
- Be able to display at least 500 fibres lag free to get a broader scope
- Offer the possibility to view fibres connected to specific brain regions
- Use the Vive VR system for input and output
- Have an orientation system built in

2. Methods

The plan was to modify respectively further develop realDTI to meet the goals mentioned above. After analysing the program architecture and the way it generates the 3D models it was clear that the best approach to achieve those goals is to use pre-rendered models instead of calculating and rendering the models during runtime. Furthermore, the control system built for the Oculus Rift is not compatible with the Vive. realDTI consists mainly of three parts: the control system, input/output for the Oculus Rift and the algorithms for calculating and rendering the models. Unfortunately, all three of those main parts could not be used so a whole new program is being developed from ground up. That program is called ViveDTI.

2.1. General procedure

Starting from DICOM (Digital Imaging and Communications in Medicine) files [11] DSI Studio generates a 3D model of the fibres and the brain volume. DSI Studio comes with an atlas library which is used to generate models of fibres connected to functional cortex regions and of fibres originating from the frontal-, parietal-, temporal-, occipital lobe and the cerebellum. These models are saved as *.wrl files for further processing in Blender to get 3D models which can be used in the Unreal Engine 4. The game engine is then used to build the main program which involves: creating different levels[1] which contain the fibre models, setting up the support for the HTC Vive VR- system, developing a control system and an orientation system, building a menu level where the user gets informed about how to control the application and where it is possible to choose from the different levels with the 3D models. Figure 1 shows a visualisation of this procedure.

[1] A level, map, area, stage, world, track, board, floor, zone, or phase in a video game is the total space available to the player during the course of completing a discrete objective. [36]

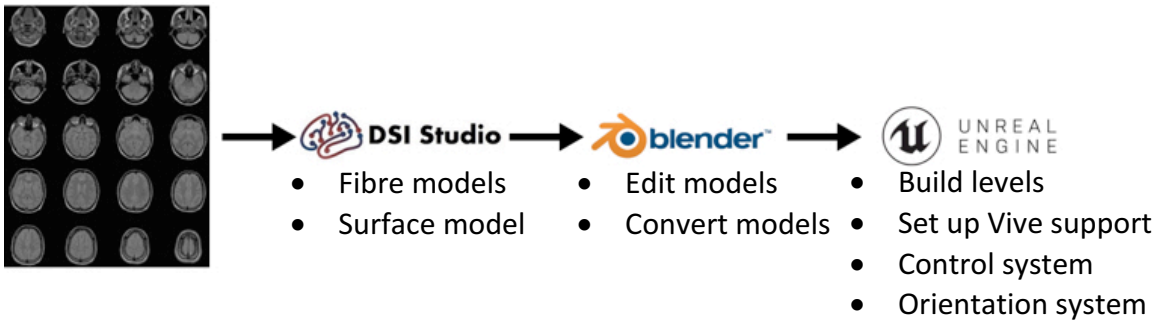

Figure 1: The general procedure of creating the program: Starting with DICOM files, generating the 3D models for brain and fibres in DSI Studio, editing and converting the models in Blender and finally import them in the Unreal Engine to build the levels and the whole program environment

2.2. Reconstruction software - DSI Studio

The fibres and the brain Volume are generated with DSI Studio [6] (Yeh; University of Pittsburgh; USA) (June 15, 2017 build). It calculates the fibres with a deterministic diffusion fibre tracking algorithm [14] from the provided DICOM files. The tracking algorithm will place starting points within seeding "voxels" at subvoxel resolution. The actual seeding points are determined randomly and uniformly within the seeding regions. For example, a seed voxel placed at (53,87,68) will have subvoxel seeding placed within (52.5~53.5, 86.5~87.5, 67.5~68.5). DSI Studio uses a deterministic random generator to place the seeds, and thus the seeding sequence is both deterministic and random. This ensures that the tracking result is reproducible using the same tracking parameters. Precisely, DSI Studio randomly selects a seed voxel (e.g. 53, 87, 68). Then, within the voxel (52.5~53.5, 86.5~87.5, 67.5~68.5), DSI Studio draws a point within the voxel range using a uniform distribution. The point is then used as the starting point within the selected voxel. [38] The software has many adjustable parameters like the number, colour and thickness of the displayed fibres. It is possible to analyse the whole brain data or specific regions via addable atlases. The generated 3D models of the brain volume and the fibres are then saved as *.wrl files which are further processed with a 3D graphicsuite.

There are different settings for how the tracks should be coloured like “assigned”, “directional” and “index based”. For this thesis, the setting “directional” was chosen. This means the colour of a specific point on a fibre depends on whether the direction of diffusion is in x, y or z direction. Figure 2 shows the colour coding on one slice of the MR data. The exported models have the colour information for every point of the model, the vertex colours, included.

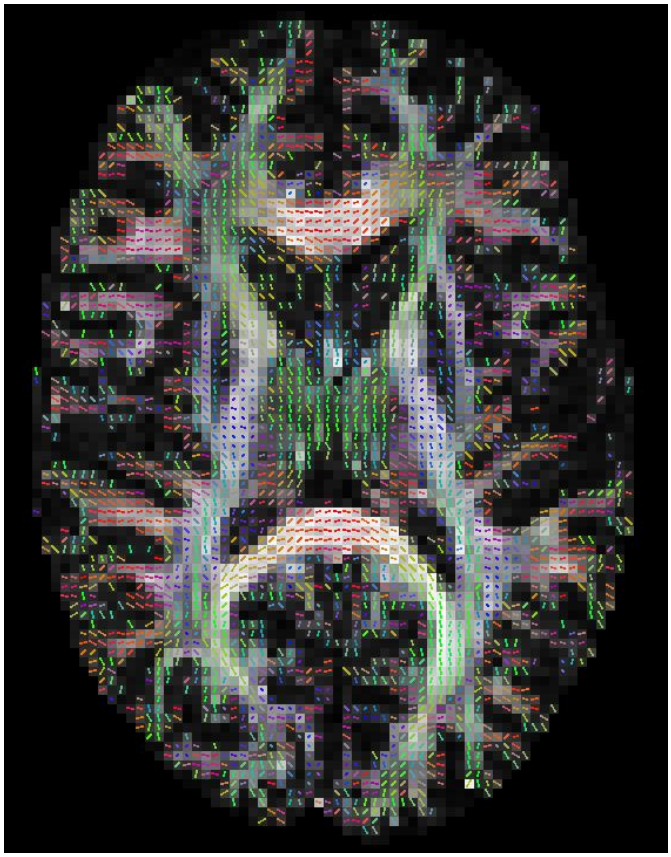

Figure 2: Colour code of diffusion direction applied on one MR slice

2.3. 3D graphicsuite – Blender

Blender [12] (Blender Foundation, Amsterdam, Netherlands) is a free and open source 3D creation suite. The used version is 2.78. It supports the entirety of the 3D pipeline—modelling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation [12]. Blender is cross-platform and runs on Linux, Windows, and Macintosh computers. Here it is used to make the 3D models generated from DSI Studio usable for the Unreal Engine 4 which involves editing the models and exporting them as *.fbx files.

The editing involves making the normals consistent. In geometry, a normal is a direction or line that is perpendicular to typically a triangle or surface. Normals of an object should face the same direction otherwise presentation disorders, like dents in a surface which should appear flat, are possible; this is why the normals have to be made consistent. For this the model is selected and in edit mode if Ctrl + N is pressed on the lower left-hand side appears a checkbox which has to be checked. Also, the export settings have to be adjusted like in figure 3. In 3D computer graphics, a smoothing group is a group of polygons in a polygon mesh which should appear to form a smooth surface. If no smoothing group is exported the Unreal Engine prompts import warnings respectively errors.

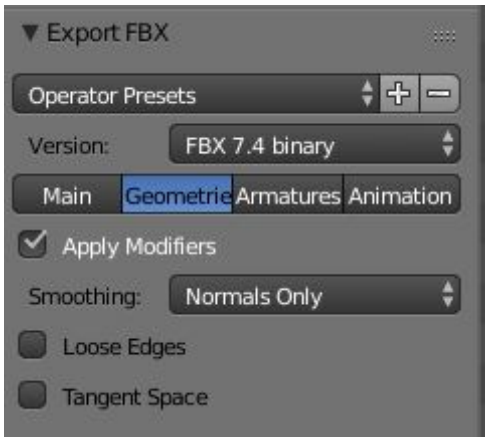

Figure 3: Blender export settings

2.4. Game development platform – Unreal Engine 4

The Unreal Engine 4 [13] is a game development platform developed by Epic Games (Cary, USA). The engine comes with a SteamVR plug-in and supports the HTC Vive out of the box. VR requires complex scenes being rendered at very high framerates. Because the Unreal Engine is designed for demanding applications such as AAA games[2], filmmaking and photo real visualisation, it meets these requirements and provides a solid foundation to build content on all VR platforms. [13] The platform supports C++ code and has its own visual coding language called blueprint. To create Code with blueprint different blocks, which are actually C++ code packages, are connected with each other. This allows fast development

[2] In the video game industry, **AAA** (pronounced "triple A") or **Triple-A** is a classification term used for games with the highest development budgets and levels of promotion. A title considered to be AAA is therefore expected to be a high quality game or to be among the year's bestsellers. [35]

without having to learn the syntax of a coding language. An example is pictured in figure 4. For this thesis, the Unreal Engine version 4.16.3 is used.

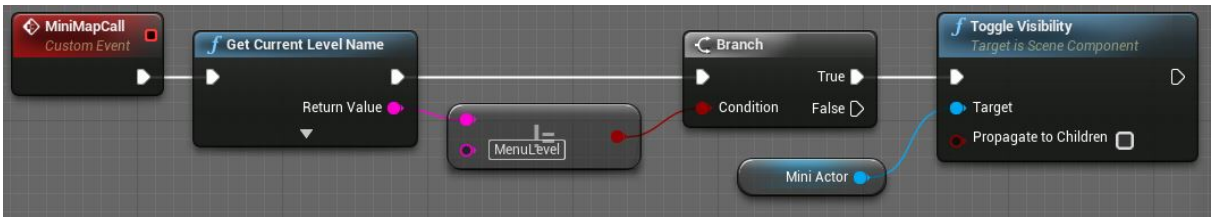

Figure 4: Example of blueprint code. If the button to call the minimap gets pushed and the current level is another than the menu level the visibility of the minimap is toggled.

2.5. Virtual reality suite – SteamVR

SteamVR [15] is developed by the Valve Corporation (Bellevue, USA) and is the link between the VR hardware and the computer program. Its task is to track the Head Mounted Display (HMD) and the controllers. Also, it prevents the user from colliding with real world objects by blending in a grid if the user gets too close to an obstacle called the chaperone. SteamVR is used as version 1510619446.

2.6. Virtual reality system – HTC Vive

The Vive [16] is developed by the HTC Corporation (New Taipei, Taiwan) and Valve and was released in 2016. It has one display for each eye with a resolution of 1080x1200 pixels and a refresh rate of 90 Hz. The controllers and the headset are covered with several infrared sensors and internal gyroscopes and accelerometers which allow precise tracking in combination with the two base stations which emit structured infrared light.

2.7. Main program – ViveDTI

The main program is launched with a typical Windows (Microsoft Corporation, Redmond, USA) *.exe file. Starting the program directly starts SteamVR if it is not running already. The main program's purpose is to visualise the data generated by the other programs and handling the user input. There are several levels for the menu and different brain regions. The user is able to move in 3D space in every level and can resize the brain and fibres in case

he wants to zoom in some area. Also, an orientation system based on a minimap[3] was implemented.

2.7.1. User Interface (UI)

Creating an UI in virtual reality leads to problems which don't arise in traditional computer programs. Usually the UI is head-locked (attached to the virtual head of the user) and orthographically projected. This causes two main issues when ported to VR. First, due to the orthographical projection the user's eyes would treat the UI as if it were infinitely far away. This leads to a mismatch between the user's eyes and every object in the virtual world that is not at infinity which can cause discomfort for the user. Another reason to avoid traditional UIs is that the lenses that are currently needed in HMDs have the limitation that the clearest image is in the centre of the user's vision. Therefore, pushing UI elements to the edges of the user's vision like traditional programs do lower the readability. Furthermore because of the differences in eye relief among users, what may be in the periphery for some may be completely hidden for others. [17] To avoid those problems a whole Level was created for the main menu and a suitable orientation system was implemented. The menu level contains nothing except a 3D- widget which can be operated by a virtual laser pointer as seen in figure 5.

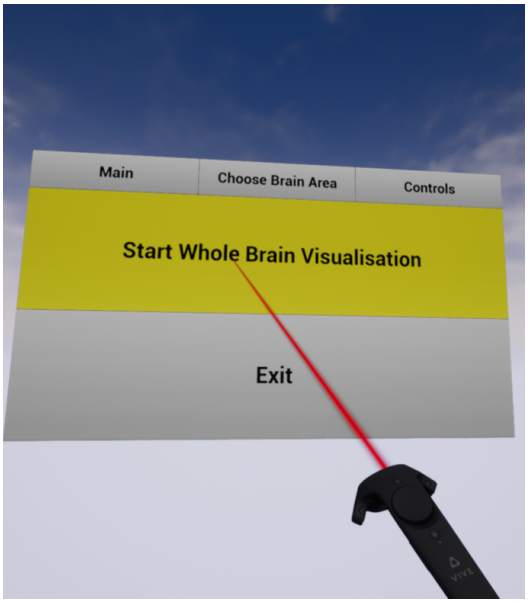

Figure 5: The main menu. The user selects a button by pointing at it and then presses the back trigger of the controller to actually "press" the button

[3] A minimap is a small live picture of the current level which indicates the user's position.

The 3D widget consists of three independent widgets which are brought together with a so-called widget switcher. A widget switcher can contain several widgets and works like an array in C++. Every widget gets an index through which a certain widget can be addressed. So, this menu consists of two parts, the widget switcher which occupies the lower area and the three buttons “Main”, “Choose Brain Area” and “Controls”. The three buttons change the active index of the widget switcher and thus the displayed widget.

The orientation system consists of two minimaps. One shows the top view of the Level and the other one shows a side view. In each of those maps is a throbbing red dot which marks the user’s position. This way it is possible to find the user’s position in 3D space in the current level, respectively the brain. Because of the reasons mentioned before the maps could not be realised like traditional minimap- based orientation systems by fixing them to the edge of the field of view. So, the maps were attached to the virtual image of the left controller, which gives the user the possibility to position them where he can read them properly. Furthermore, when the minimap is called the opacity of the brain surface is increased so that the outline of the brain is better visible on the minimap and the fibres are made invisible to ensure an unobstructed view on the minimap. This can be seen in figure 6.

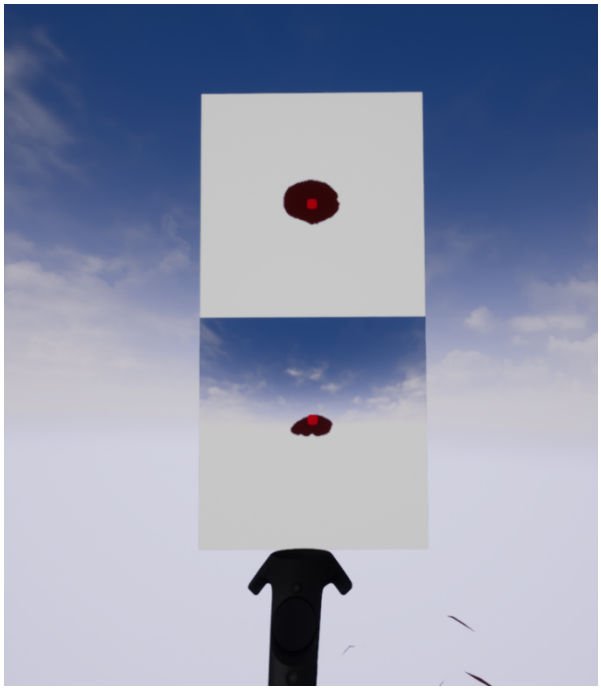

Figure 6: The minimaps called from inside the brain. The opacity of the brain surface is increased for a better visible outline but is transparent from inside the brain. The fibres are made invisible to ensure an unobstructed view of the minimaps. The red dot shows the user's location.

Each of the minimaps consists of five parts: a camera, a texture, a material, a widget and an actor. First, the cameras are put into the level. It is important, that the cameras are aligned with the point where the player spawns when entering a level because otherwise the dot indicating the player's position would have an offset. The distance of the cameras to the object of interest, in this case the brain, determines the scale of the object in the minimap. The position of the cameras can be seen in figure 7.

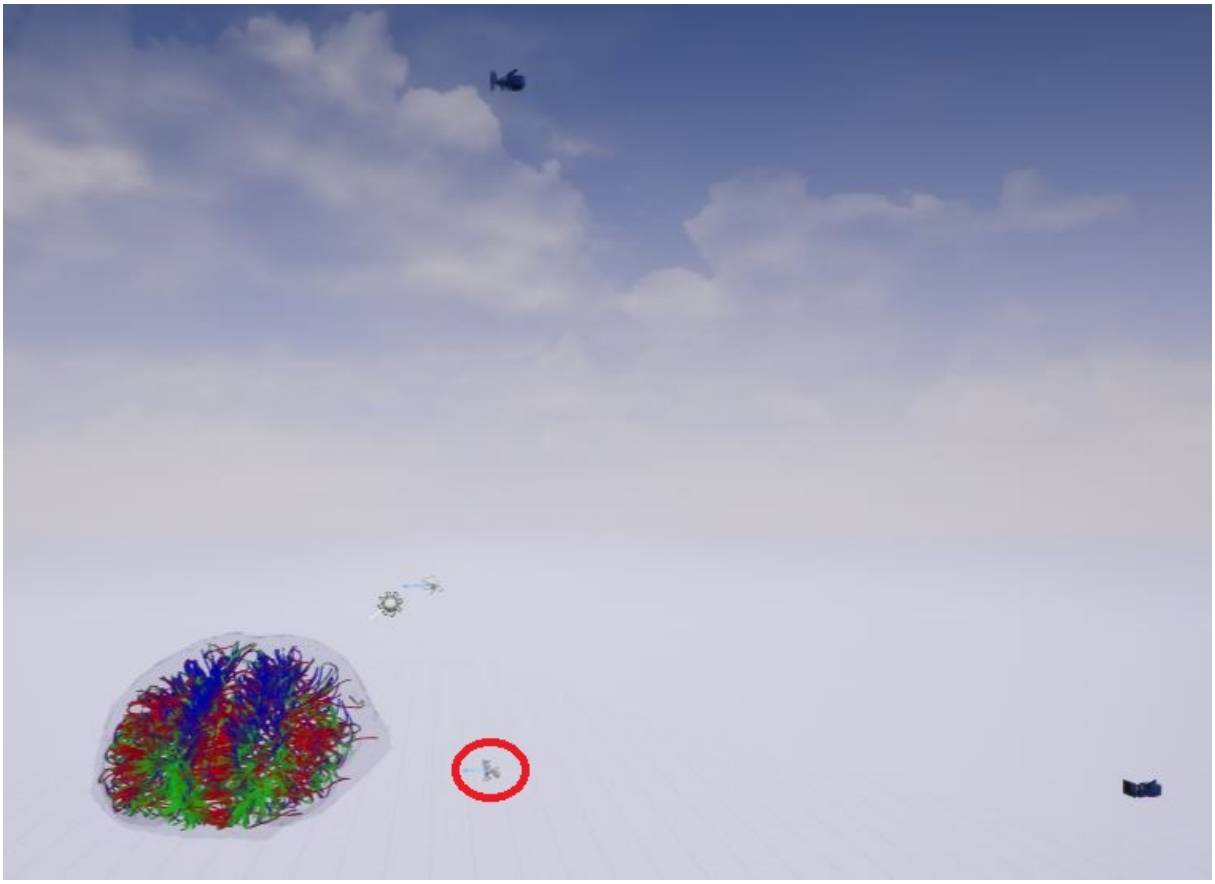

Figure 7: Position of the cameras for the minimaps in the frontal lobe level. The red circle marks the players spawning point.

Furthermore, the cameras still have to be aligned with the player when he moves. For this purpose, custom events called “Outside Tick” for the top view camera and “Side Tick” for the side view camera have been set up. Every level has its own blueprint in which an “Event Tick” is preconfigured to fire an event on every frame. This event is linked to the custom events in order to move the cameras with the player. Figure 8 shows the setup for the custom events and figure 9 shows how they are used for the cameras using the top view camera blueprint as an example.

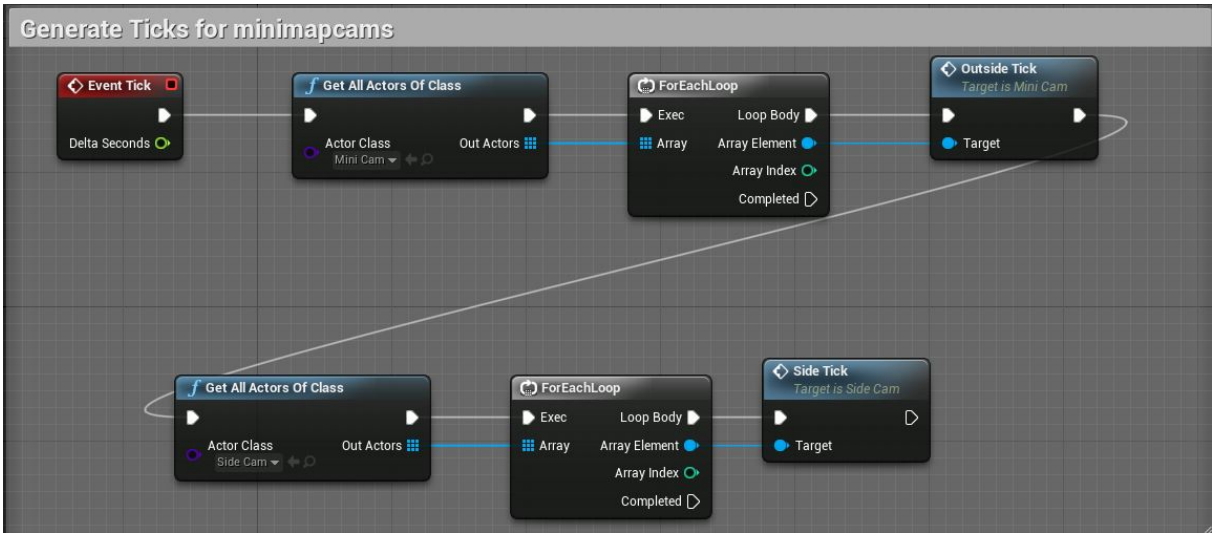

Figure 8: Creation of a custom event

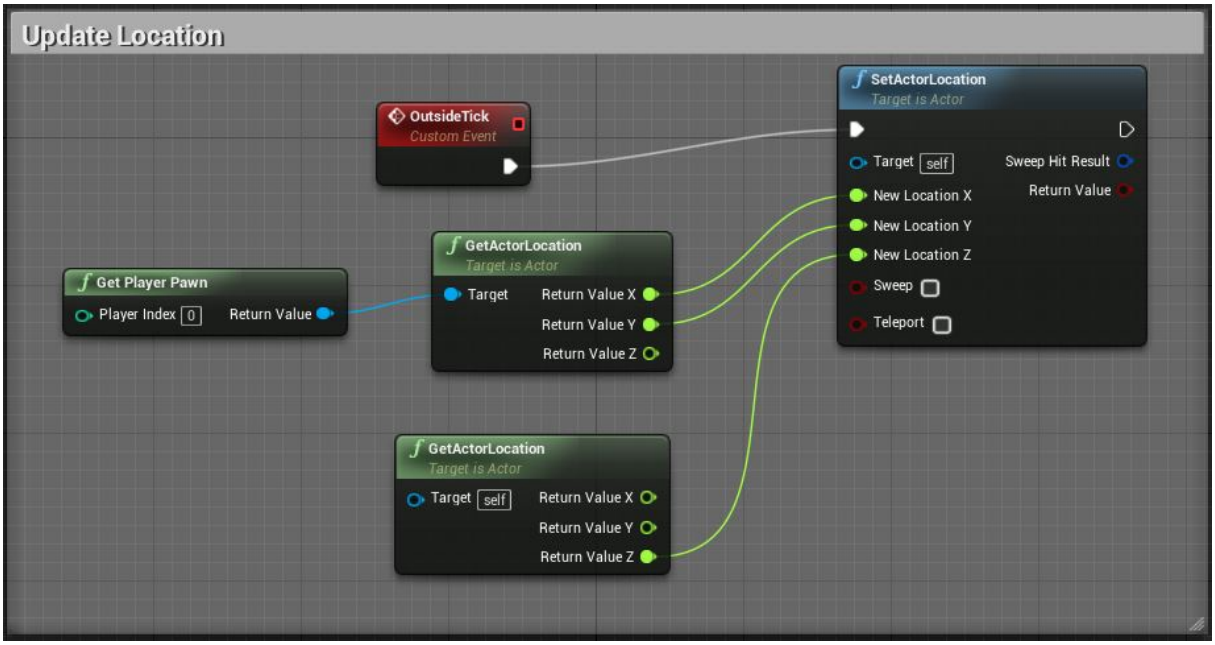

Figure 9: Use of custom event so that the camera follows the user

Second, a TextureRenderTarget2D is added. This is a texture which can be used as a render target for a camera which is exactly what has been done. Everything the camera sees gets transferred to the texture. The texture can be applied to any material which leads to the next step, creating a material. A basic material blueprint is added and as a texture the TextureRenderTarget2D is used. This material is then used as an image in a widget blueprint and because this is a VR application the widget has to be a 3D widget. This is achieved by putting the widget into an actor and placing that actor into the level.

2.7.2. Levels

As mentioned in 2.7.1. the menu was realised in its own level as were the brain regions frontal lobe, temporal lobe, occipital lobe, parietal lobe and cerebellum. They can be accessed from the menu. Another approach was taken at the fibres connected to the functional regions motoric cortex, auditory cortex, visual cortex and somatosensory cortex. They share a level with the visualisation of fibres of the whole brain and can be switched through. To let the user know what he is looking at a label floats above the brain to indicate what fibres are visible at the moment. The label is programmed to always face the user when he moves around the brain. The models of the brain areas were obtained using the MNI atlas and the models of the functional regions were obtained by using the Brodmann atlas of DSI Studio.

2.7.3. Input

The program gets user input primarily from the two controllers of the Vive but also by movement of the user.

Controller input Via buttons on the controllers the user can move along the x-, y- and z-axis in and around the brain. They are also used to call the minimap, select items in the menu, switch through the functional regions in the whole brain visualisation and change the size of the brain and the fibres. The Key assignment is pictured in figure 11 in section 3.2.

Movement input Because of the tracking system of the Vive it is possible to move in the program by moving in the real world. Movement in the real world in direction of the x- and y- axis will be transferred one to one to the virtual world. This is just an additional input method because it requires at least 2x1.5 meters free space and the vertical movement and the functions are still operated with the controllers.

2.7.4. Output

The output is principally on the Vive. For spectators is one of the Vive displays mirrored to the computer display. This is automatically handled by SteamVR.

3. Results

This part deals only with the main software created with the Unreal Engine 4. DSI Studio, Blender and SteamVR are independent software and therefore not a part of the results of this thesis. How to create the models with DSI Studio and Blender and how to import them to the Unreal Engine is described in the manual as appendix.

3.1. Menu

The main menu is a whole autonomous level with a 3D- widget for the user to interact with. As pictured in figure 5 in 2.7.1. the user can start the visualisation of fibres running through the whole brain or exit the application. As already mentioned in 2.7.2. the user can switch through different functional regions in the whole brain visualisation. Those are pictured in figure 13 in 3.3. Brain and fibres. Figure 10 shows the selection screen to start the visualisation of the different brain areas mentioned in 2.7.2. and pictured in figure 14. In figure 11 the key assignment for the controllers is pictured.

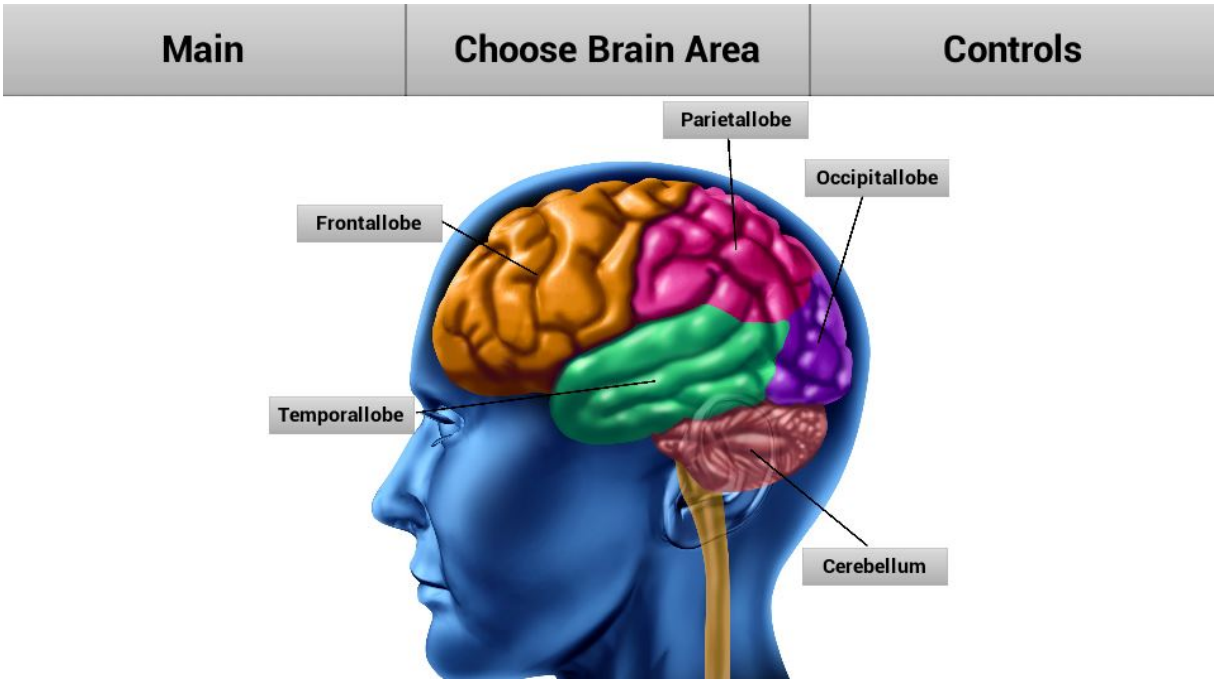

Figure 10: Menu screen for brain area selection

3.2. Controls

As already described in 2.7.3. input can be given by using the Vive controllers and moving around. The key assignment of the controllers can be seen in figure 11.

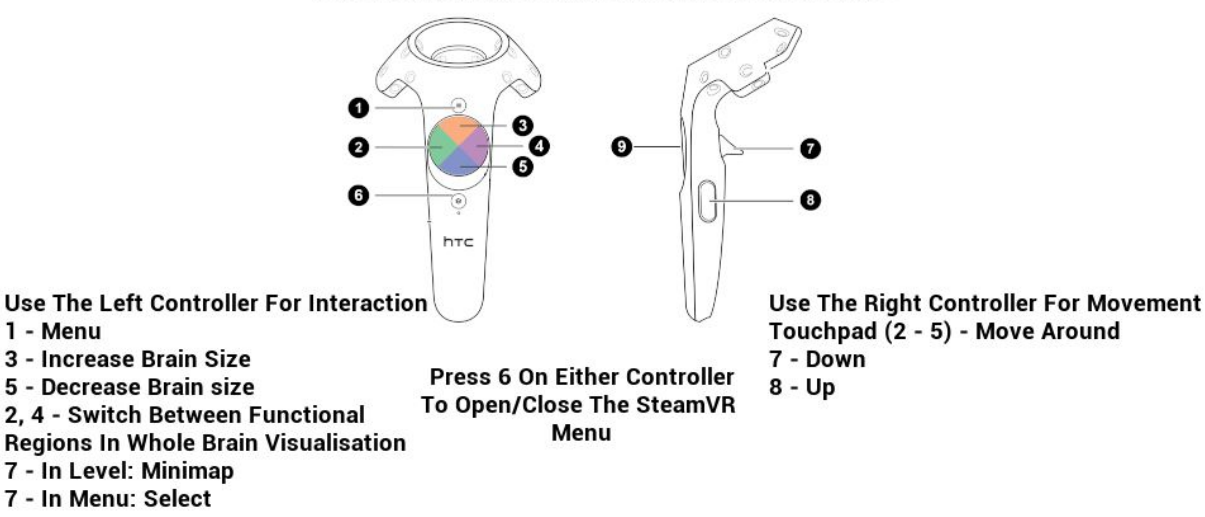

Figure 11: The key assignment of the left and right controller

3.3. Brain and fibres

The colorization of the fibres is predefined by DSI Studio and depends on the direction of the diffusion. The colour of the surface is chosen by the developer and is not changeable for the user of ViveDTI. The user moves around a static brain to look at the area of interest. Figures 12 to 14 are screenshots of all different 3D models of the brain surface and the fibres in the application. All models are generated with the same dataset provided by the Graz University of Technology and each model contains 1000 fibres.

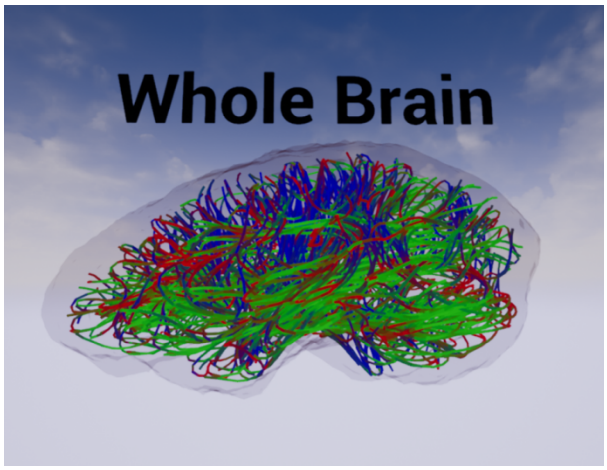

Figure 12: Visualisation of 1000 fibres through the brain

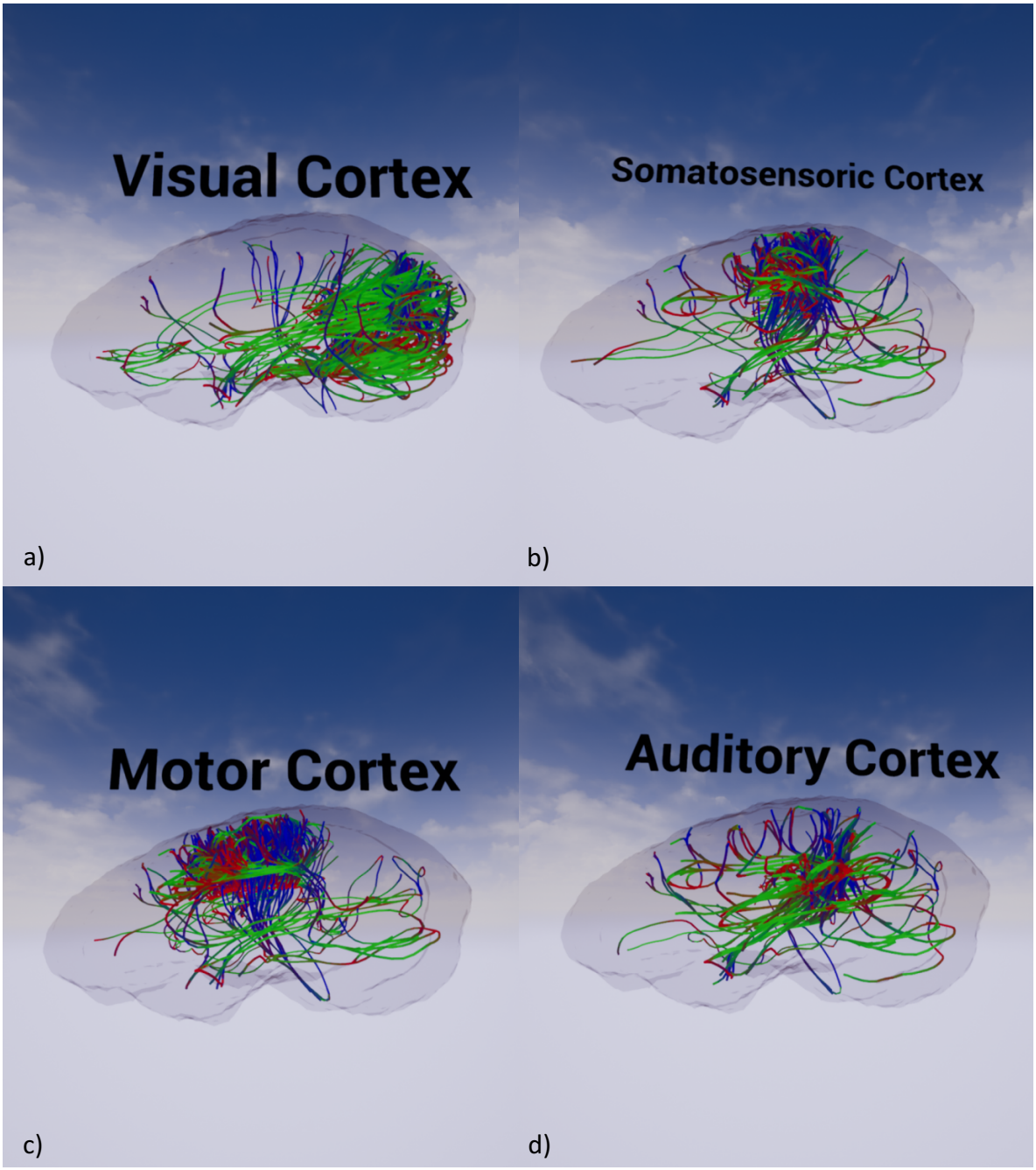

Figure 13: Visualisation of 1000 fibres connected to a) the visual cortex b) the somatosensory cortex c) the motor cortex d) the auditory cortex

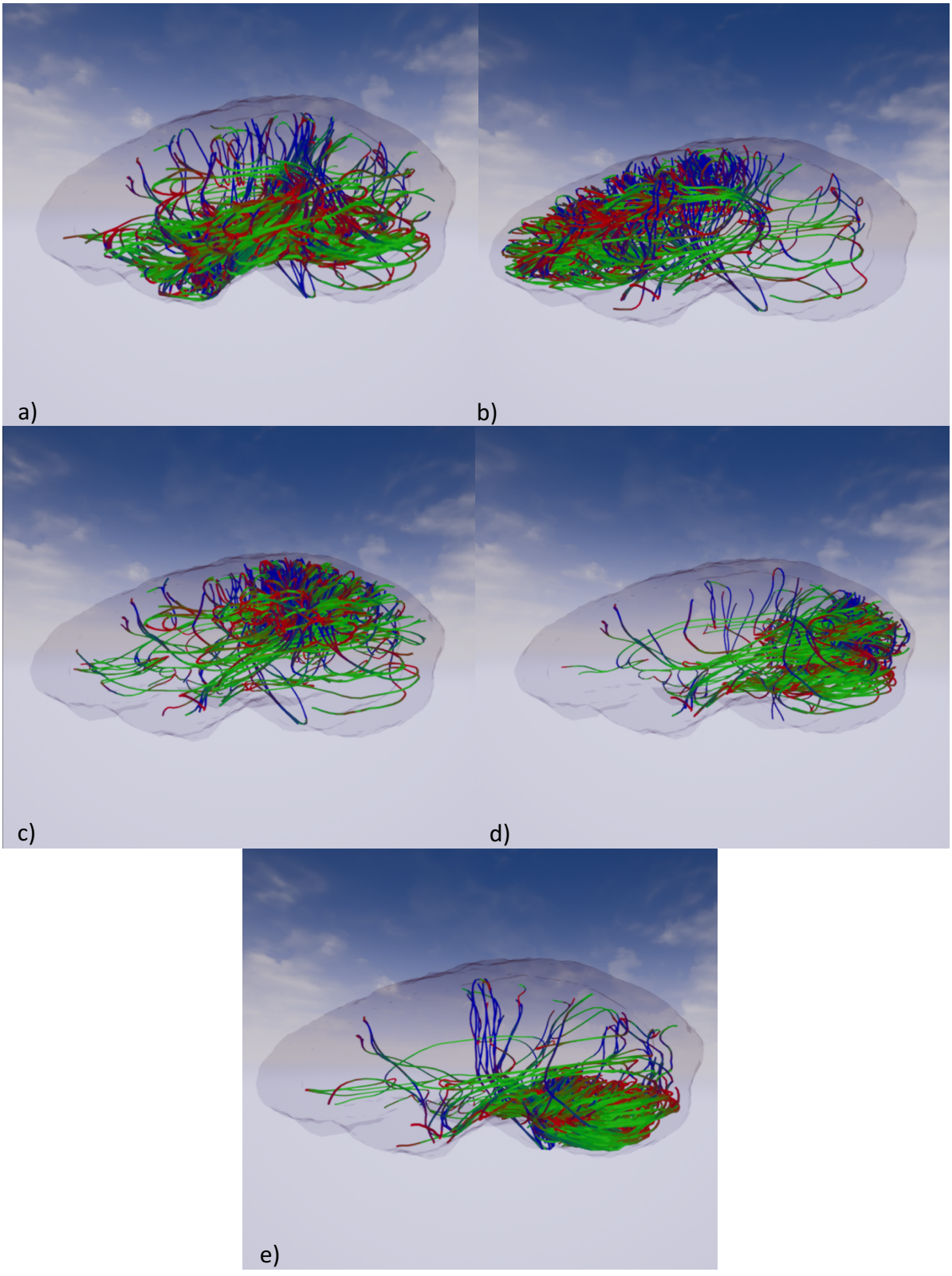

Figure 14: Visualisation of 1000 originating from a) the temporal lobe b) the frontal lobe c) the parietal lobe d) the occipital lobe e) the cerebellum

3.4. Orientation system

Figure 15 shows an example of the minimaps when the user is outside the brain. The brain surface is made opaque so the outline of the brain is better visible on the minimaps. The red dot indicates the user's position. In figure 6 in 2.7.1. the minimaps can be seen when called inside the brain.

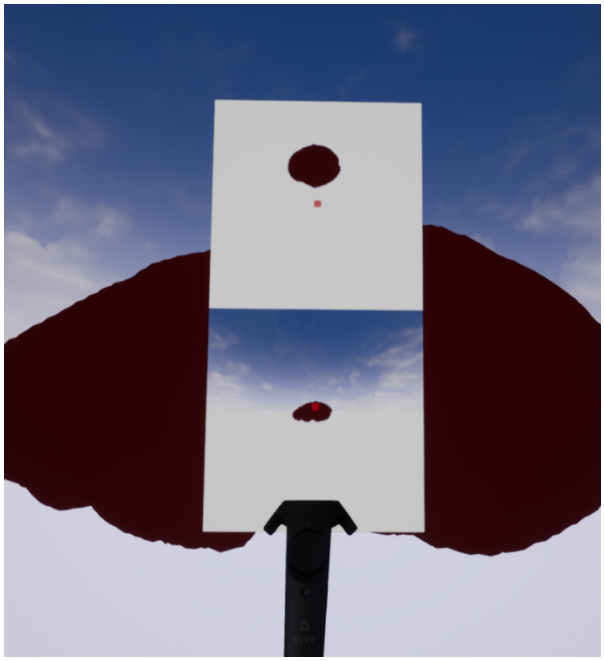

Figure 15: The minimaps called from outside the brain. The opacity of the brain surface is increased for better visibility on the maps

3.5. Performance

The graphs in figure 16 were measured on an Intel i7-7700k accompanied by 16 GB DDR4 RAM and an 11 GB Nvidia Geforce 1080 Ti. The measuring programs were MSI Afterbuner v4.3.0.9267 [24] (Micro-Star International, Zhonge, China) in combination with RivaTuner Statistics Server v7.0.0. Beta 30 [25]. No other programs were open at the time of monitoring so the hardware usage seen in figure 16 can be assumed as caused just by ViveDTI itself. As seen in figure 16 the GPU usage is at its maximum 79%, the frames per second (FPS) are mostly stable at 90, the CPU usage fluctuates between 30% and 10% and the RAM usage is below 800 MB.

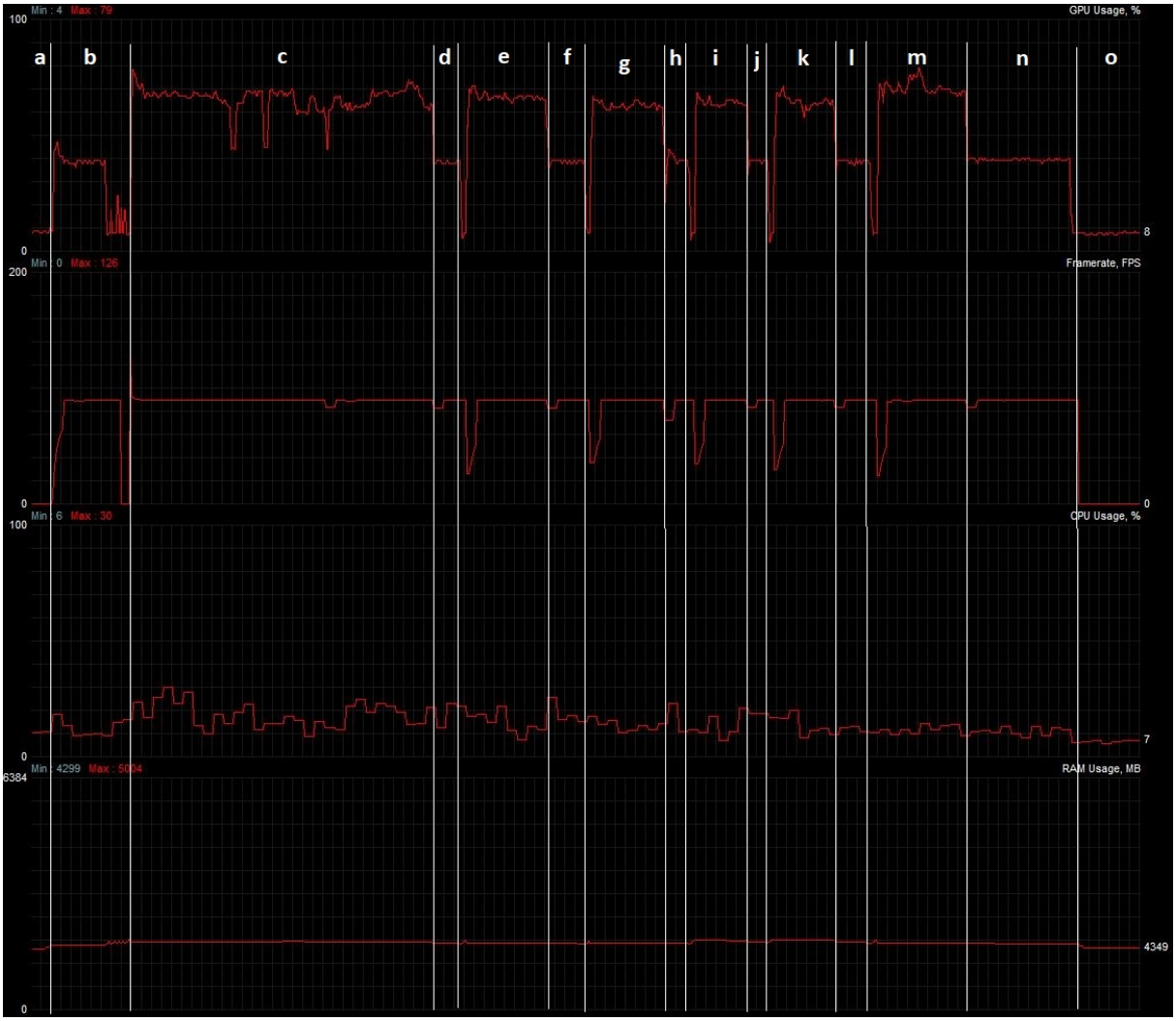

Figure 16: Framerate, GPU-, CPU- and RAM- usage during execution of ViveDTI. a) before program start, b,d,f,h,j,l,n) menu level is executed; at the framerate and GPU usage drops a new level is loaded, c) whole brain visualisation is executed; variation in GPU- and CPU- usage due to moving around in the level and switching through functional regions, e) temporal lobe visualisation, g) frontal lobe visualisation, i) parietal lobe visualisation, k) occipital lobe visualisation, m) cerebellum visualisation, o) program exit; hardware usage drops to operating system consumption

4. Discussion

4.1. Software

4.1.1. DSI Studio

This software was used because of the good feedback in [5]. The graphic user interface makes it easy to use and the manual explains the analysing steps well. Furthermore, the support for this application [18] is very good and fast. Creating and exporting the models calculated from the DICOM files was fast and convenient.

4.1.2. Blender

Blender was chosen for editing and converting the models so that they can be used in the Unreal Engine because it is a powerful open source program which is free to use. It can handle a lot of different file formats for import and export and offers a lot of tools for editing. This big scope of functions is also a disadvantage because it makes it hard to get started with the application. But the documentation [19] is helpful and there is also a support forum [20] to help with specific questions.

4.1.3. Unreal Engine

This software comes with a lot of plugins and offers an environment to create a game from the first level to the packaged program for shipping. The documentation [21] is very helpful and extensive. Also, there is a Youtube channel [22] of the developing studio and a community based support site [23]. Due to the graphical programming language blueprint it is easy to get started and for specific tasks which are not already implemented by the developers the user is able to create his own blueprint functions using C++. The HTC Vive and its controllers are well integrated as the engine supports the HMD out of the box and works flawlessly with SteamVR. Also, there are preprogramed functions to use the controller keys.

4.2. HTC Vive

The VR system works well with the Unreal Engine and setting it up for use in a game is easy using the official Unreal Engine documentation [21].

4.2.1. Motion tracking

On one hand the motion tracking is precise and no latency in the movements were recognisable but on the other hand it needs at least 2 x 1.5 m of space to work correctly. For this application if the user intends to use solely the movement input to navigate around the brain even more space is needed. Approximately 4.5 x 4.5 m is recommended which is the maximum area the Vive can cover.

4.2.2. Image quality

The two displays of the HMD have a resolution of 1080 x 1200 pixels and a refresh rate of 90 Hz. This leads to a good image quality and a field of view of 110 degrees. Although the resolution of the Vive is high there is still a screen door effect noticeable. This means individual sub-pixels are discernible when focusing on them which impacts negatively on the image quality and the VR experience.

4.3. Main program

4.3.1. Performance

The program performs well and without lag. The personal experience is corroborated by the monitored hardware usage pictured in figure 16 in the results. As there can be seen, the RAM usage of the program is only about 700 MB which is very low compared to the 4300 MB the computer uses already for the operating system and background processes before and after ViveDTI is executed. The CPU- usage is 30% at its highest point and 7% at the lowest point during execution which is as low as the CPU- usage after the program is closed. The framerate is zero before and after the program execution. This is because the monitoring software only measures the frames per second if a game or similar software is running. During execution of ViveDTI the framerate is mostly stable at the 90 Hz of the Vive. Only when switching from the menu to a level or vice versa the framerate drops significantly. As expected the biggest and longest framerate drop is at the beginning of section c in figure 16 because that is the point when the whole brain simulation was launched, which is the most complex level. During the loading time of approximately two seconds the displays of the vive remained black which explains why during this time the FPS count is zero and the GPU- usage is also at its lowest. The GPU is the most demanded part of the hardware with a usage of 79% at the highest point. This is because of the HTC Vive with its high resolution

and refresh rate. Just to visualise the menu level needs around 40% of the GPU's capacity and the levels with the fibres are even more demanding with a GPU- usage between 60% and 70%. Like the framerate the GPU- usage drops during loading of levels because nothing has to be rendered then.

4.3.2. Input

Using the controllers to move around in the levels works well and enables the user to work with the application either seated or standing. This way only very little space is needed, even just one of the base stations is sufficient though not recommended. On the other hand, moving in VR without moving in real life can cause kinetosis in some users. Kinetosis is also called motion sickness and arises if the movement of the body sensed by the vestibular system does not correlate with the movement perceived with the eyes.

Moving around the brain horizontally by using the tracking system to transfer the movement of the user into the virtual reality of the program works also, but as mentioned in 4.2.1. significantly more space is required. However, using this option for movement lowers the risk of kinetosis.

4.3.3 3D- Models

The way the models are implemented leads to a great performance as seen in 3.5. but also has its downsides which are discussed below.

Implementation of new Models Every set of fibres that should be visualised with this application has to go through the data acquisition process of analysing in DSI Studio, editing in Blender and also putting them in the levels with the editor of the Unreal Engine. After that the program has to be packaged again with the Unreal editor to work as a stand-alone application.

Artefacts In DSI Studio the fibres look smooth with a steady colour transition but when imported in Blender the transition between two fibre segments looks hard. This can be seen in figure 17.

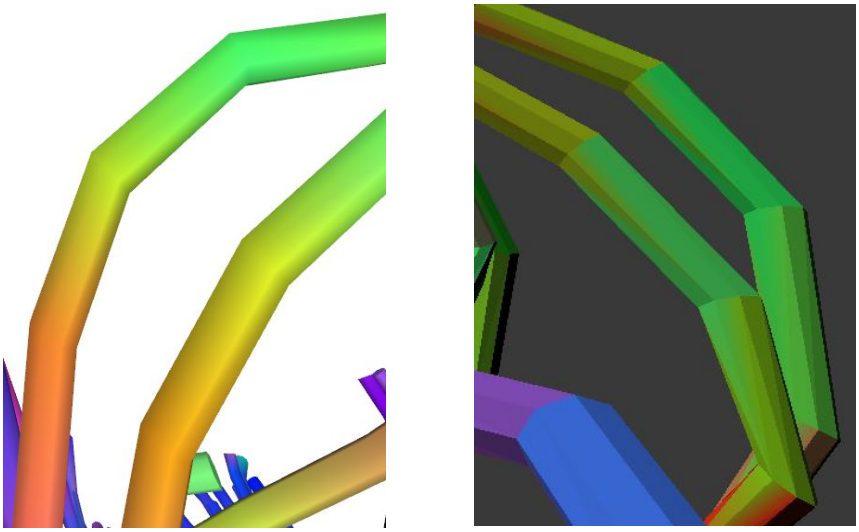

Figure 17: The same fibres from different angles. The left picture is from DSI Studio and the right is from Blender.

Also, the colour transition gives a torn look at some points as pictured in figure 18.

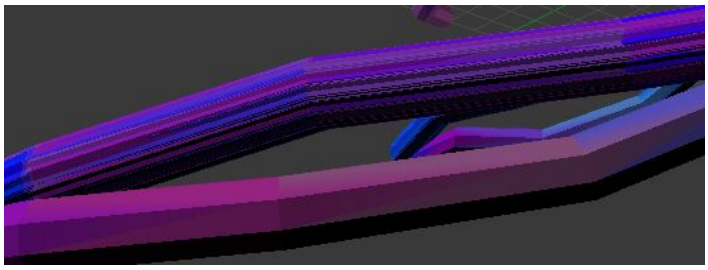

Figure 18: Fibres in Blender. Colour transition makes fibres look torn at some points.

This change in the look of the tracks most likely comes from the export process from DSI Studio because if the models are opened with another 3D suite like Microsoft 3D- Builder (Microsoft Corporation, Redmond, USA) these changes are also noticeable.

4.3.4. Orientation system

The two minimaps with different perspectives on the level enable the user to find his position in 3D space. However, the current implementation is not ideal because the brain on the maps looks small, making it difficult to determine the exact location within the brain. This issue could be solved by either enlarging the brain or moving the cameras which generate the minimap images closer. Also, it would be better if only the fibres between the minimaps and the user were hidden. Furthermore, it would benefit the user if the minimap would show in what direction he is looking. This way the user would be able to look at the

minimap to find his position and check if the fibre structure he is looking at is the one of interest.

4.4. Performance comparison between realDTI and ViveDTI

In table 1 the key numbers representing the performance of the two programs are compared. The measurements on realDTI were taken from [5] and are therefore limited to the maximal fibre count available in that thesis.

Table 1: Comparison between key performance values of realDTI and ViveDTI

	realDTI	ViveDTI
Fibres	500	1000
FPS	30	90
CPU usage maximum	13%	30%
RAM usage	2250 MB	700 MB

As it can be seen, although ViveDTI shows a fibrecount twice as high as realDTI the framerate and the RAM usage are better than the values realDTI delivers. The maximum CPU usage of ViveDTI is 17% higher than the CPU usage of realDTI which means ViveDTI is potentially more demanding on the CPU.

4.4. Conclusion

The software delivers a virtual reality experience which enables the user to move through the brain and to look at the fibres and the connections they make. The program developed meets the goals defined in 1.3. ViveDTI displays twice as much fibres than realDTI at a three times higher framerate while consuming less RAM. The different levels load on the test PC in under two seconds and are lag free. Except for the graphics card no hardware part is used even to half of their capacity. This means there is some potential for cost reduction by using less potent hardware. The different 3D models implemented in the created levels allow looking at fibres originating from different brain regions and fibres connected to functional cortex regions. The Vive VR system is fully integrated and used for input and output. An orientation system and menu suitable for virtual reality applications were developed and implemented. ViveDTI can be used for educational purposes or presentations but because of the work and time intensive exchange of 3D models an application in diagnostics is not beneficial.

4.5. Further research

The biggest limitation of the program is that the 3D models can only be changed if new models are generated with the process described in 1.1 and implemented in a new program using the Unreal Engine and the basic structure of ViveDTI. An investigation of the command line input for DSI Studio and Blender would be interesting to see if the needed process could be automated. Also, ViveDTI itself would need some restructuring. Integrating a file handling system to load specific models from the hard drive of the computer would pave the way for using such an application for diagnostic purposes. Furthermore, implementing a gesture control system with the Vive controllers would give the user a more intuitive way to interact with the program and thus a more immersive experience.

5. References

[1] Le Bihan D, Breton E.: Imagerie de diffusion in-vivo par résonance. C. R. Acad. Sci. (Paris). 301:1109–12 (1985)

[2] Qiu A, Mori S, Miller M. I.: Diffusion Tensor Imaging for Understanding Brain Development in Early Life. Annu Rev Psychol. 66: 853–876 (2015)

[3] Vedantam A, Jirjis M. B., Schmit B, Wang M. C., Ulmer J. L., Kurpad S. N.: Diffusion Tensor Imaging of the Spinal Cord: Insights From Animal and Human Studies. Neurosurgery. 74(1): 1–8 (2014)

[4] Basser P.J., Mattiello J, LeBihan D.: MR diffusion tensor spectroscopy and imaging. Biophys J. 66(1):259–267 (1994)

[5] Eixelberger T. realDTI – A software package to visualize DTI data in the Unreal Engine 4 using the Oculus Rift and Leap Motion. University of technology Graz, 2017

[6] Yeh F. DSI Studio website, 2017. http://dsi-studio.labsolver.org/

[7] Wang R, Wedeen V. J.. TrackVis website, 2017. http://trackvis.org/

[8] Conte G, Ye A. Q., Forbes A. G., Ajilore O, Leow A: BRAINtrinsic: A Virtual Reality-Compatible Tool for Exploring Intrinsic Topologies of the Human Brain Connectome. University of Illinois at Chicago. Lecture Notes in Computer Science 9250:67-76 (2015)

[9] Hasan K. M., Walimuni I. S., Abid H, Hahn K. R.: A Review of Diffusion Tensor Magnetic Resonance Imaging Computational Methods and Software Tools. Computers in Biology and Medicine. 41(12): 1062–1072 (2011)

[10] Feldmann H. M., Yeatman J. D., Lee E. S., Barde L. H. F., Gaman-Bean S: Diffusion Tensor Imaging: A Review for Pediatric Researchers and Clinicians. Journal of Developmental & Behavioral Pediatrics. 31(4):346-356 (2010)

[11] National Electrical Manufacturers Association (NEMA). Dicom, digital imaging and communications in medicine, 2016. http://medical.nema.org/standard.html

[12] Blender Foundation. Blender website, 2017. https://www.blender.org/

[13] Epic Games Inc. Unreal Engine website, 2017. https://www.unrealengine.com

[14] Yeh F, Verstynen T.D., Wang Y, Fernández-Miranda J. C., Isaac Tseng W: Deterministic diffusion fiber tracking improved by quantitative anisotropy. (2013)

[15] Valve Corporation. SteamVR website, 2017. http://store.steampowered.com/steamvr

[16] HTC Corporation. Vive website, 2017. https://www.vive.com

[17] McCaffrey M: Unreal Engine VR Cookbook. Boston – Massachusetts, Addison-Wesley (2017)

[18] Yeh F: Discussion group for DSI Studio, 2017. http://dsi-studio.labsolver.org/discussion-group

[19] Blender Foundation: Blender documentation, 2017. https://docs.blender.org/

[20] Stack Exchange Inc.: Blender Q&A subsite, 2017. https://blender.stackexchange.com/

[21] Epic Games Inc.: Unreal Engine documentation, 2017. https://docs.unrealengine.com/

[22] Epic Games Inc., Youtube LLC: Unreal Engine development channel, 2017. https://www.youtube.com/user/UnrealDevelopmentKit

[23] Epic Games Inc.: Unreal Enginen answerhub, 2017. https://answers.unrealengine.com

[24] Micro-Star International: MSI Afterburner website, 2017. https://www.msi.com/page/afterburner

[25] Guru3D: RivaTuner statistics server website, 2017. http://www.guru3d.com/files-details/rtss-rivatuner-statistics-server-download.html

[26] MRI Symbolpicture, 2017. https://www.mathworks.com/content/mathworks/www/en/company/newsletters/articles/accessing-data-in-dicom-files/_jcr_content/mainParsys/image_0.img.jpg/1469941449199.jpg

[27] DSI Studio Logo, 2017. http://dsi-studio.labsolver.org/_/rsrc/1468760876817/config/customLogo.gif?revision=17

[28] Blender Logo, 2017. https://download.blender.org/institute/logos/blender-plain.png

[29] Unreal Engine Logo, 2017. https://epicgames.app.box.com/s/dzi271i7lenfe0hr6t7ihgbs981lvypd/file/208465651423

[30] Vive controller button layout, 2017. Modified for this thesis. https://forums.unrealengine.com/filedata/fetch?id=1111783&d=1460020388

[31] Brain regions, 2017. http://smskouc.cz/wp-content/uploads/2015/03/bigstock-Human-brain-chart-with-colored-11863427.jpg

[32] Acosta- Cabronero J., Nestor P.J.: diffusion tensor imaging in Alzheimer's disease: insights into the limbic- diencephalic network and methodological considerations. Frontiers in aging neuroscience. 6:266 (2014)

[33] Oishi K., Mielke M.M., Albert M., Lyketsos C.G., Mori S.: DTI analyses and clinical applications in Alzheimer's disease. Journal of Alzheimer's disease. 26 Suppl 3:287-96 (2011)

[34] Commowick O., Fillard P., Clatz O., Warfield S.K.: Detection of DTI white matter abnormalities in multiple sclerosis patients. Medical image computing and computer-assisted intervention. 11(Pt1): 975-982 (2008)

[35] Sbardella E., Tona F., Petsas N., Pantano P.: DTI measurements in multiple sclerosis: Evaluation of brain damage and clinical implications. Multiple sclerosis international. 2013:671730 (2013)

[36] https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/AAA_%28video_game_industry%29.html (2017)

[37] https://en.wikipedia.org/wiki/Level_(video_gaming) (2017)

[38] http://dsi-studio.labsolver.org/Manual/Fiber-Tracking (2017)

A. **Manual**

A.1. DSI Studio

1. Open the desired source images (DICOM files). The B-table will open afterwards.

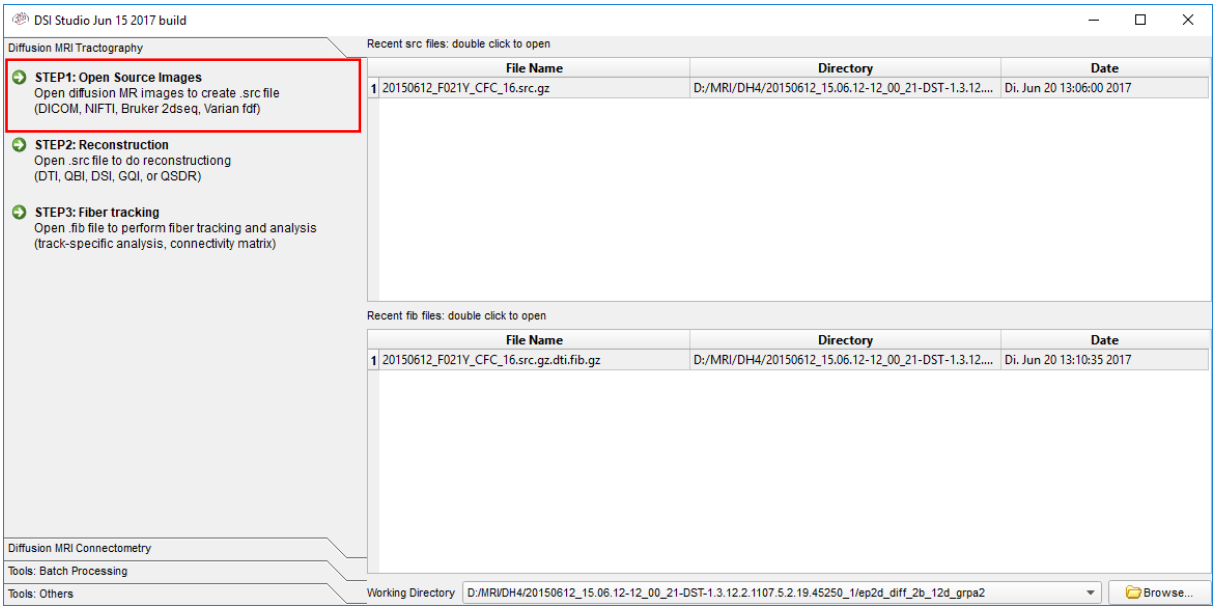

Figure 19: press the highlighted button to open the source images

2. Check the B-table data. By clicking on “OK” a DSI Studio source file will be generated.

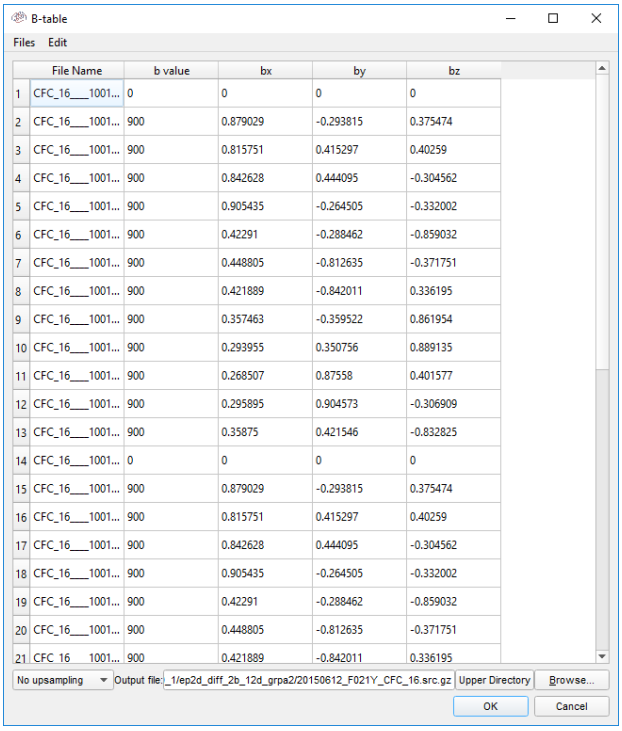

Figure 20: B-table of the used images

Optional: load an external B-table if necessary.

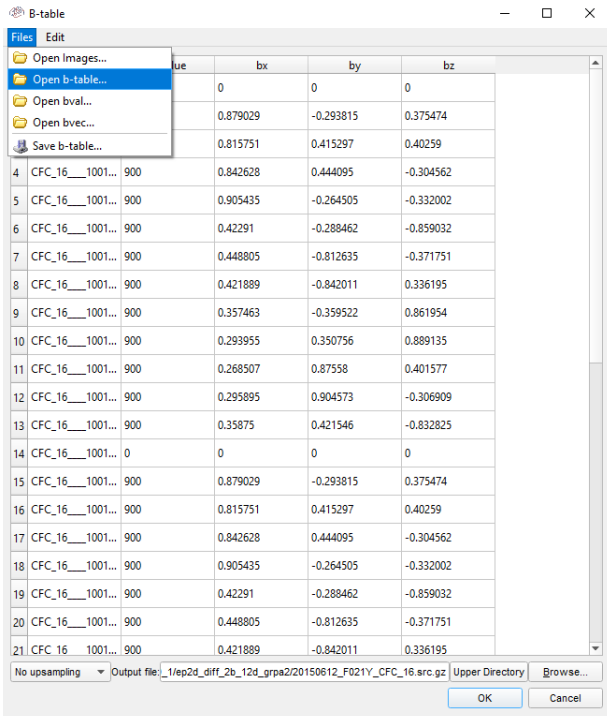

Figure 21: how to load an external B-table

3. Begin the reconstruction.

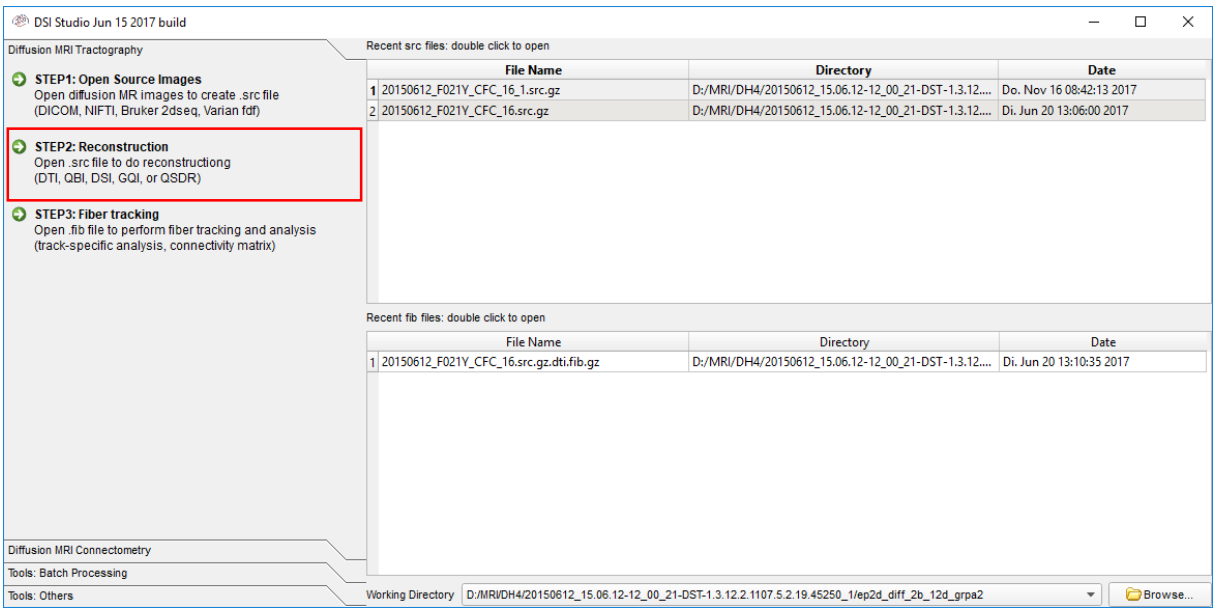

Figure 22: press the highlighted button to begin the reconstruction

4. Open the DSI Studio source file generated in step 2.

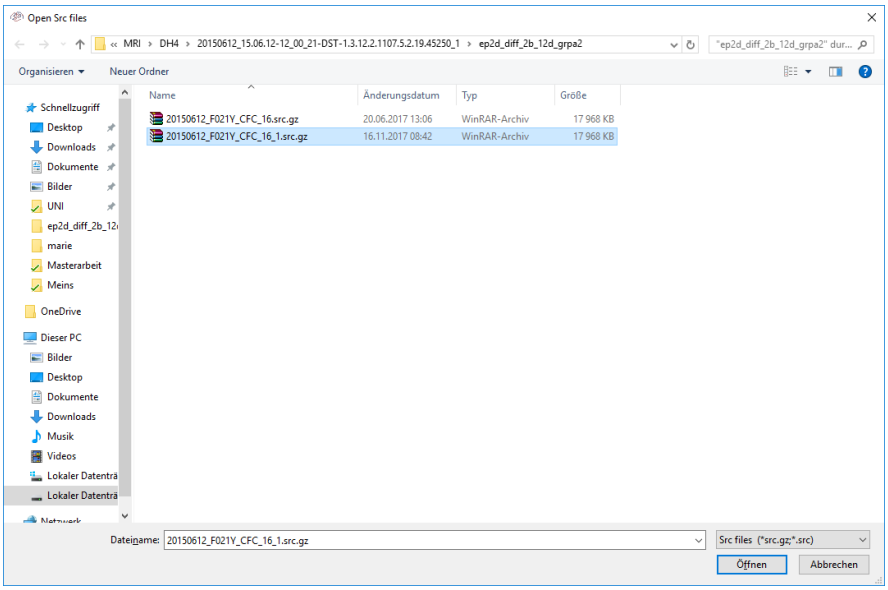

Figure 23: DSI Studio source file selection screen

5. Optimize the threshold, dilation and other parameters if necessary. Afterwards click on “select reconstruction method”.

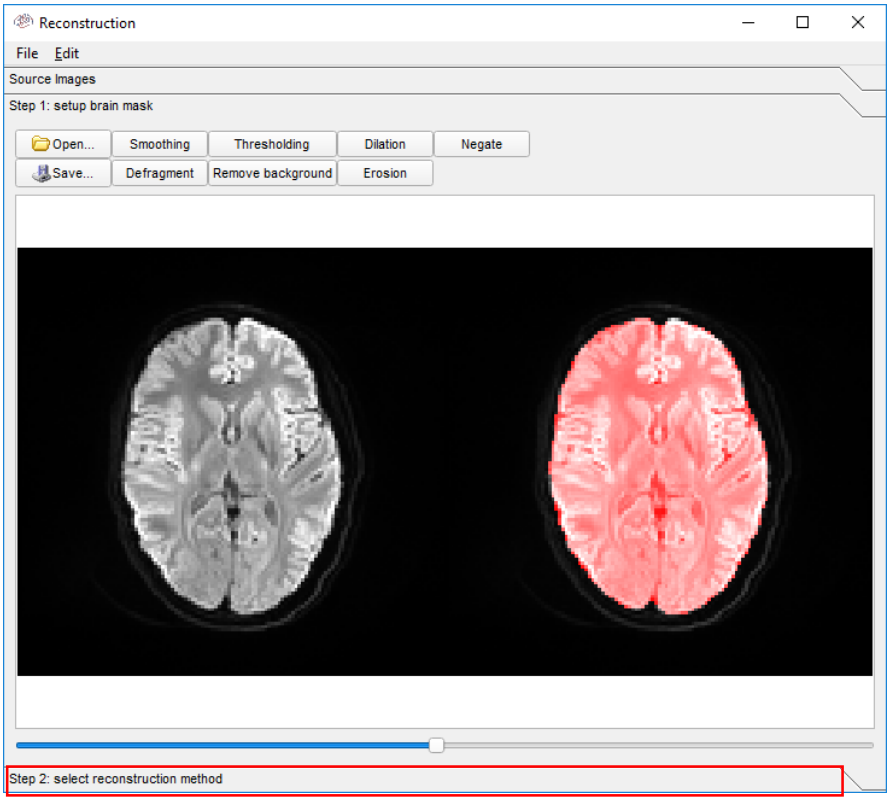

Figure 24: screen for parameter adjustment

6. Select “DTI” and run reconstruction. A *.fib file will be created.

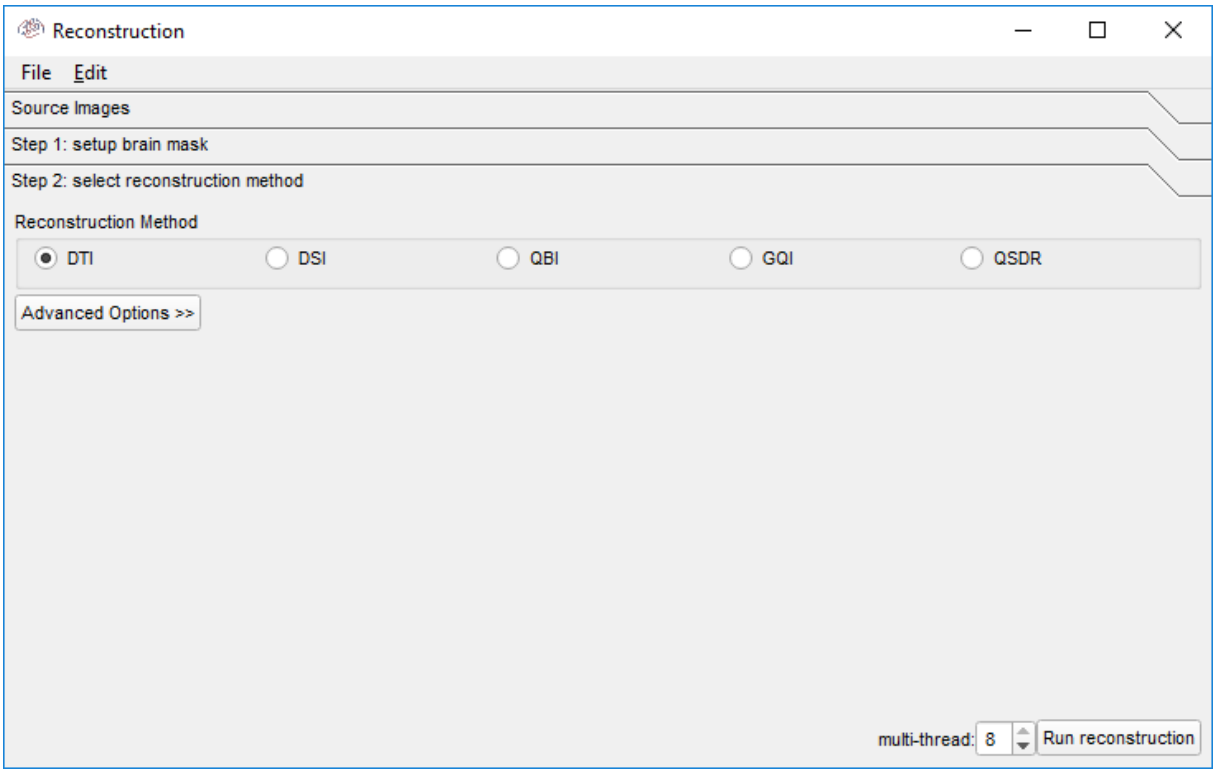

Figure 25: reconstruction method selection screen

7. Begin fibre tracking.

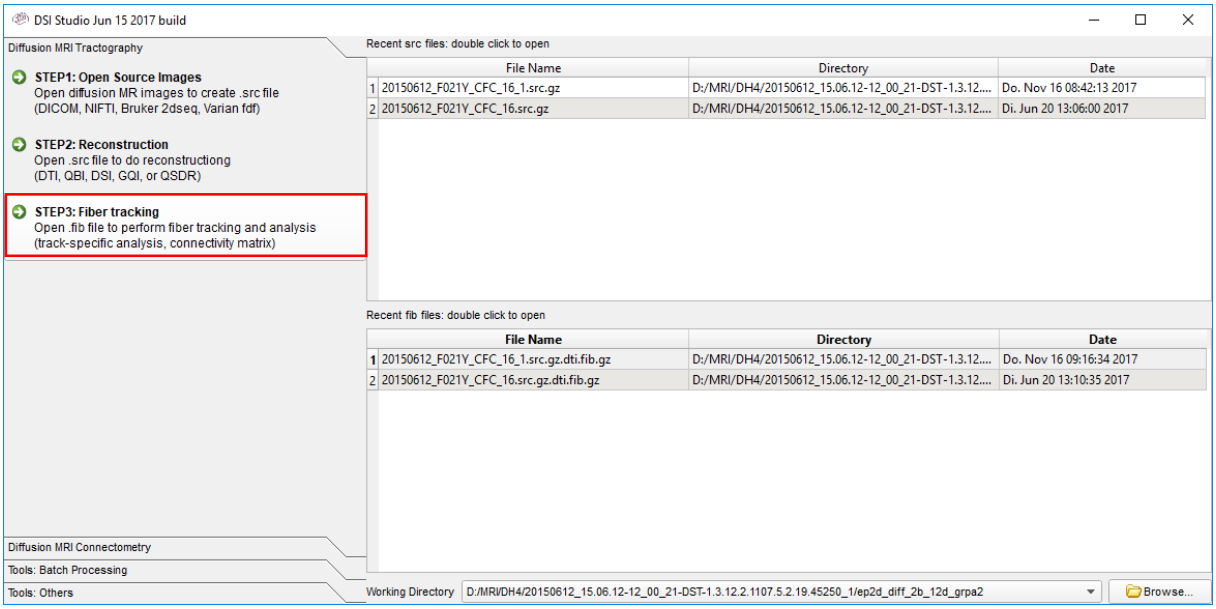

Figure 26: press the highlighted button to start fibre tracking

8. Open the *.fib file created in step 6.

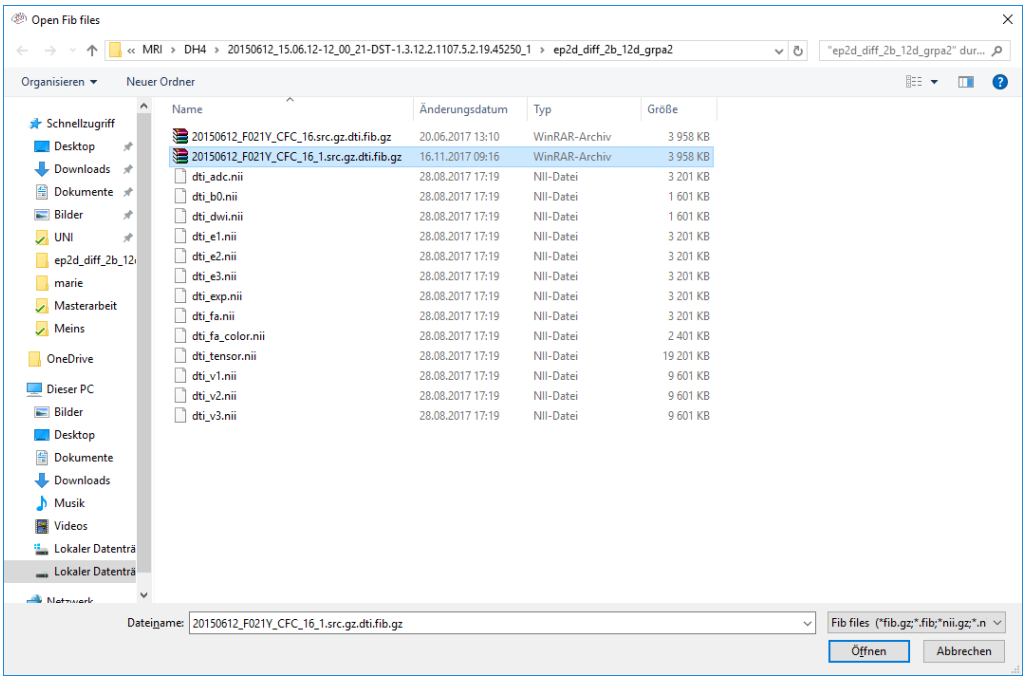

*Figure 27: *.fib file selection screen*

9. Adjust the tracking parameters like threshold, how many fibres should be tracked and so on in the options window

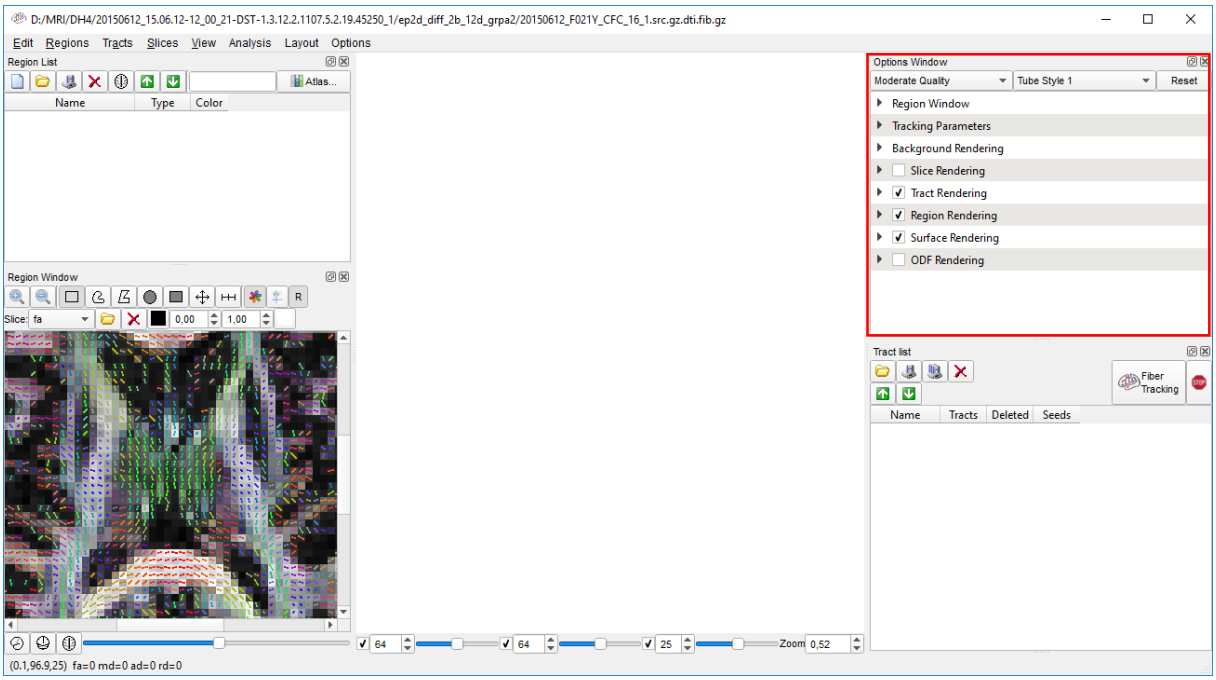

Figure 28: work screen of DSI Studio, options window highlighted

10. Click on “Fiber Tracking “. The fibre network will be built.

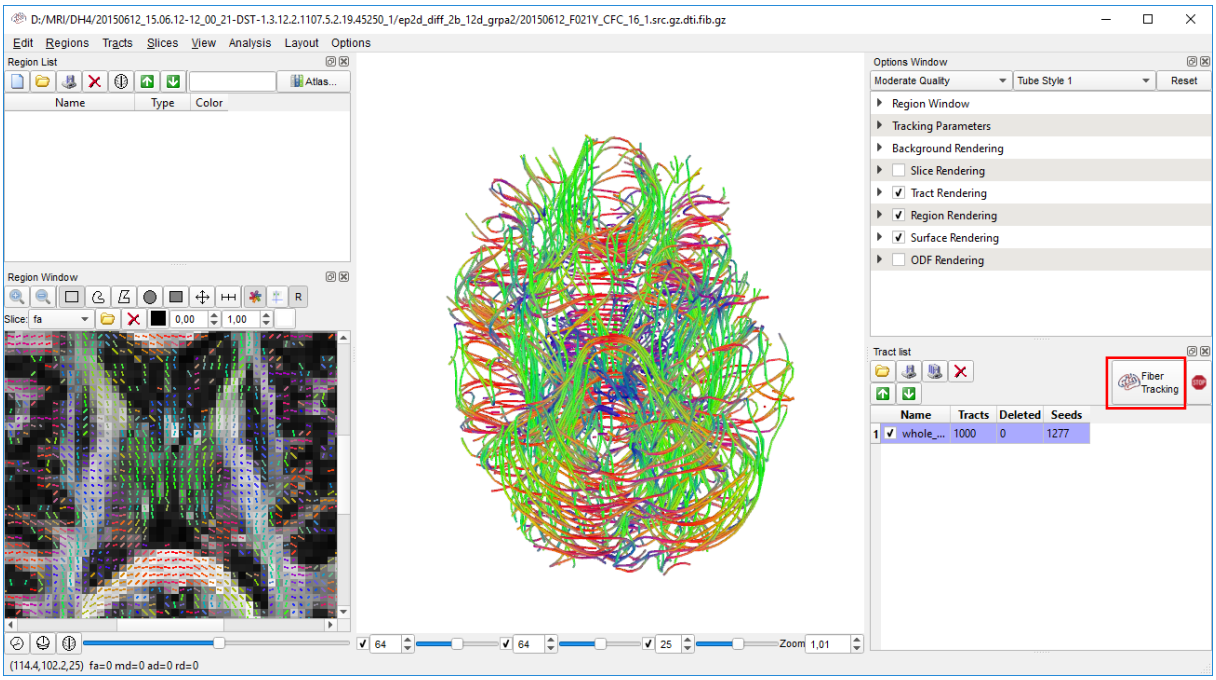

Figure 29: reconstructed fibre network

Optional: To get only fibres of specific brain areas choose an atlas before clicking on “Fiber Tracking”.

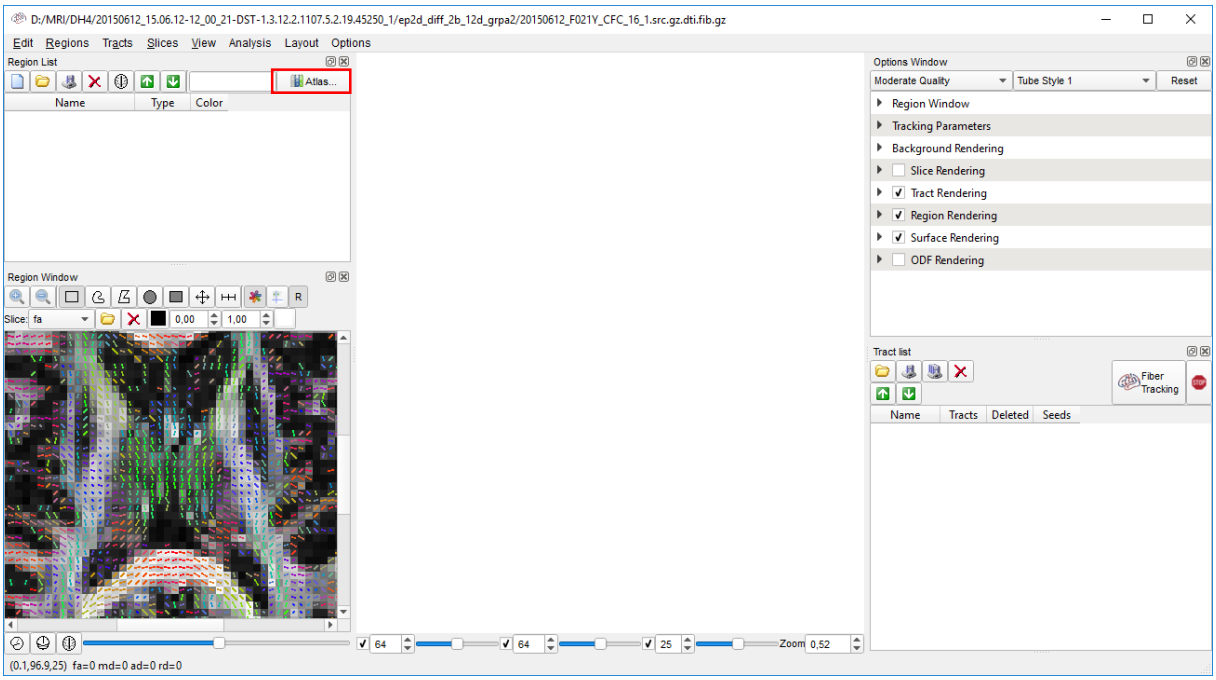

Figure 30: click on the highlighted button to choose an atlas

After choosing a region it will appear in the region list. You can choose between different types for the region. For definition of those types please look at the documentation on [18]. For this thesis the type “Seed” was chosen. After clicking on “Fiber Tracking” the network originating from the chosen region will be generated.

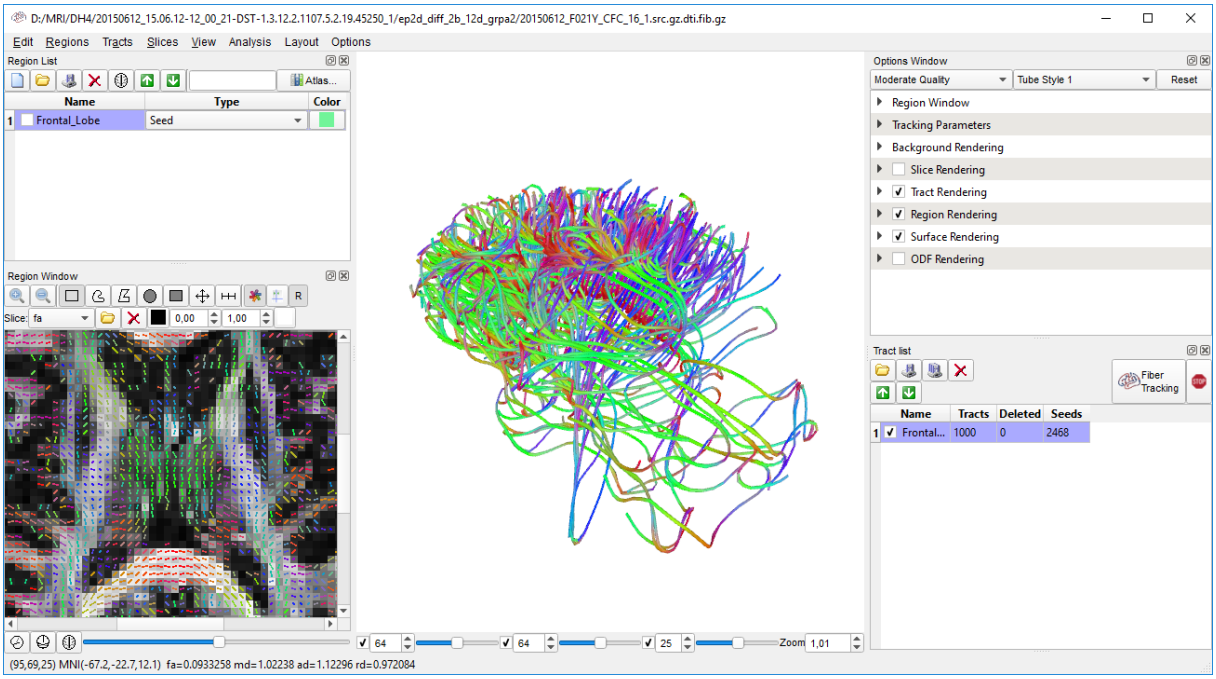

Figure 31: fibre network originating from the frontal lobe

11. Save the fibres as *.VRML file.

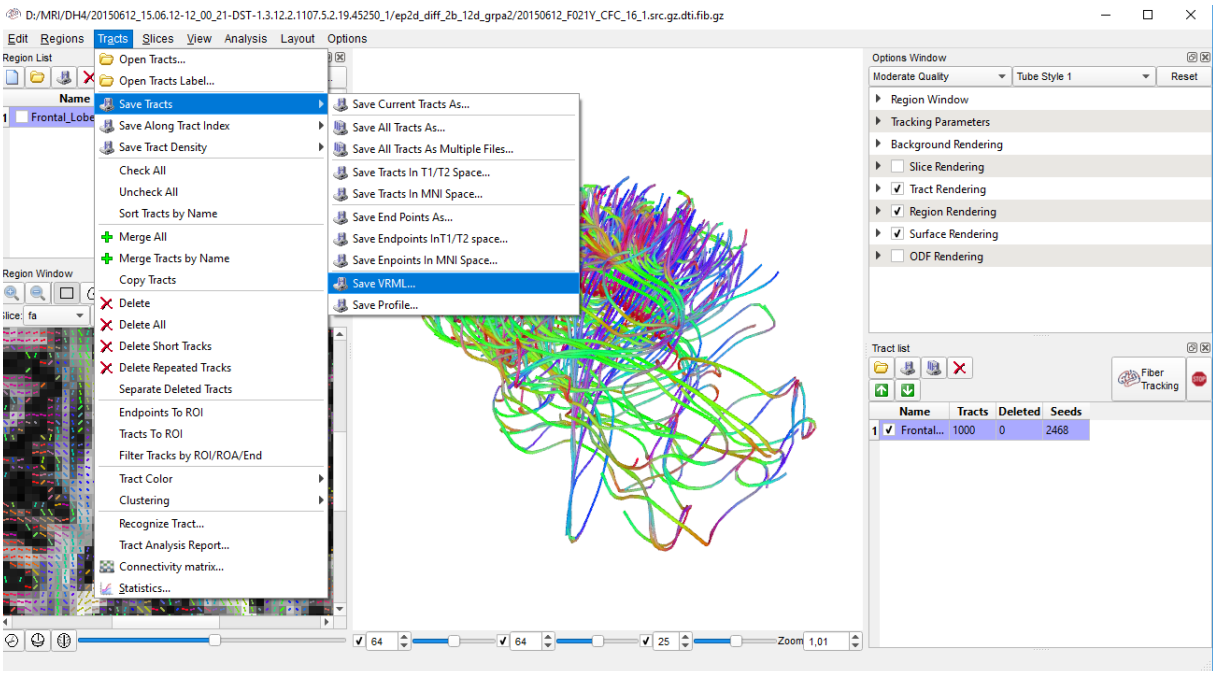

*Figure 32: saving the fibre network as *.VRML file*

12. Add the surface. You will be prompted a window to choose a threshold. For this thesis the threshold was 0,015.

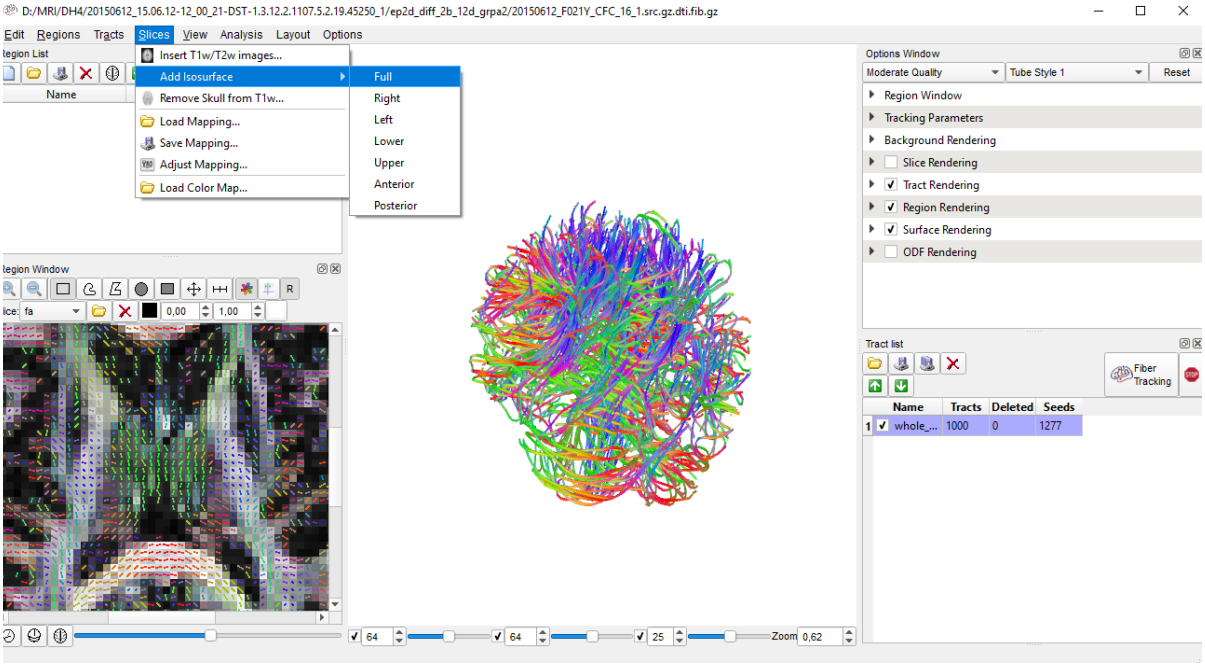

Figure 33: adding the surface

13. Save the model as in step 11 but be sure to give the file a different name.

A.2. Blender

1. Delete the default items camera, cube and lamp. To do so select them in the highlighted field and hit “delete” on your keyboard.

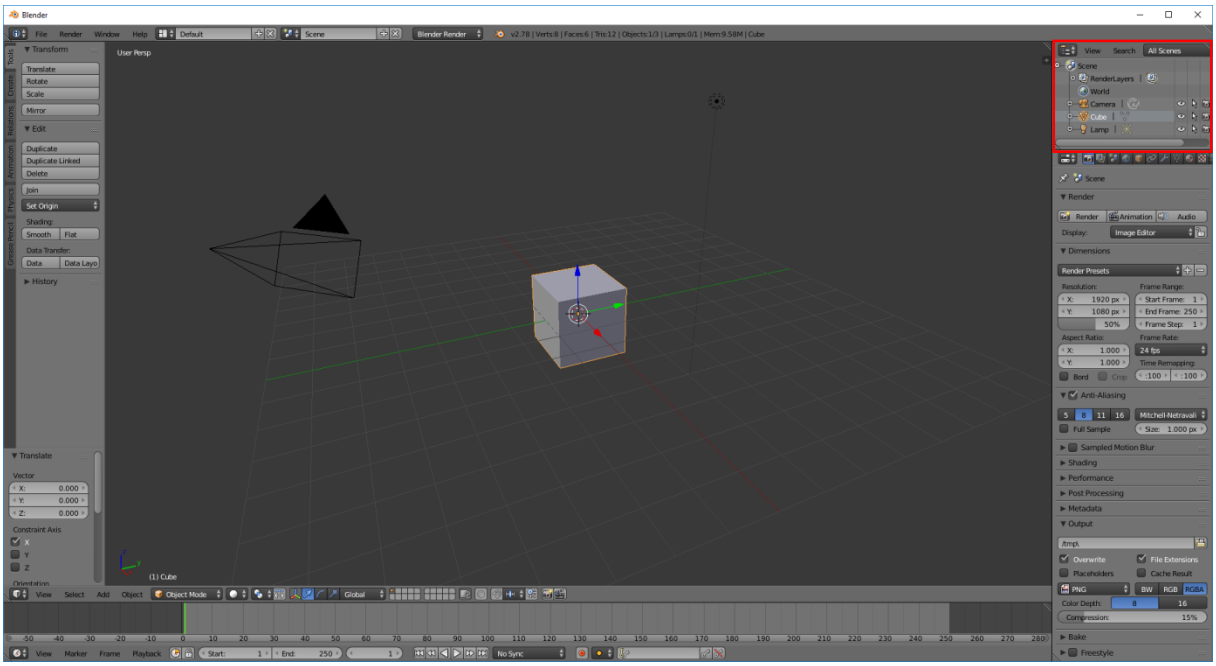

Figure 34: Default work screen of blender

2. Import the fibre model.

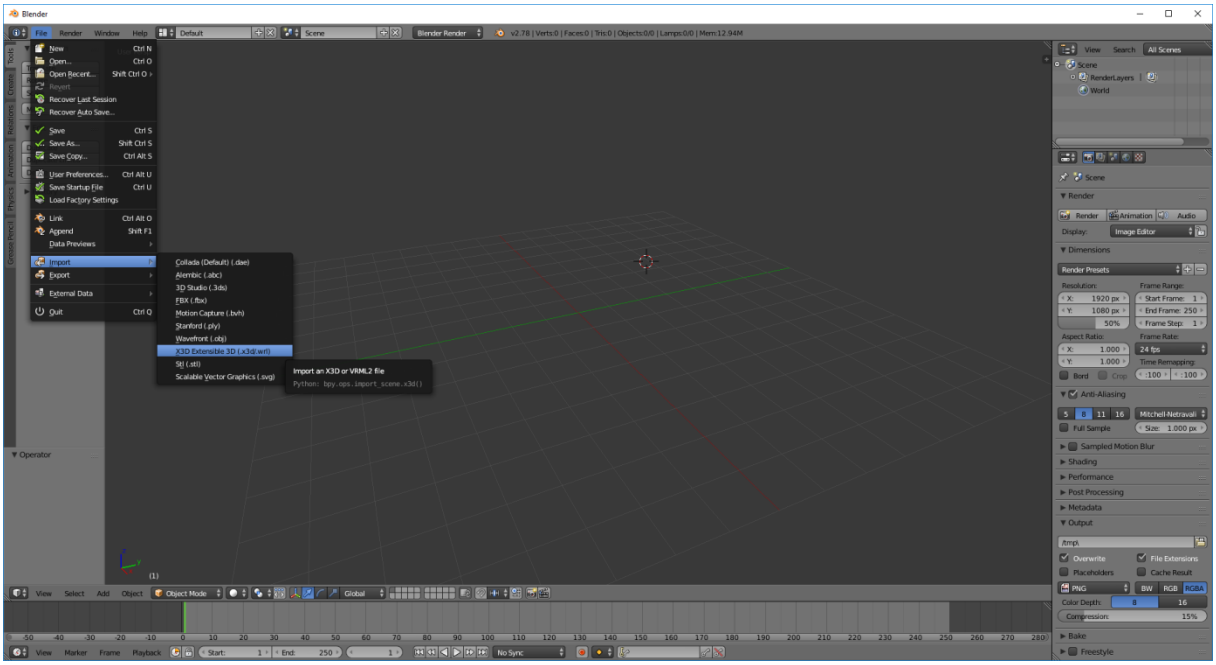

Figure 35: importing the fibre model

3. Right click on the model to select it and change to edit mode.

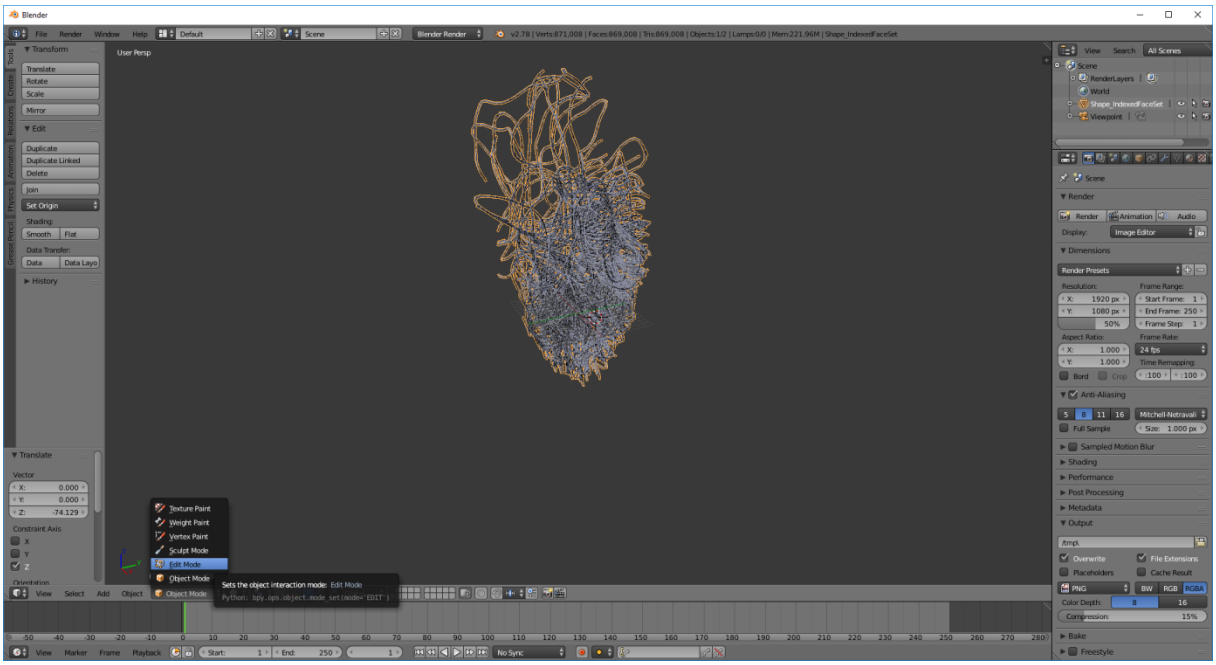

Figure 36: switching to edit mode with the model selected

4. In edit mode press “ctrl + n” and check the highlighted field of figure 37.

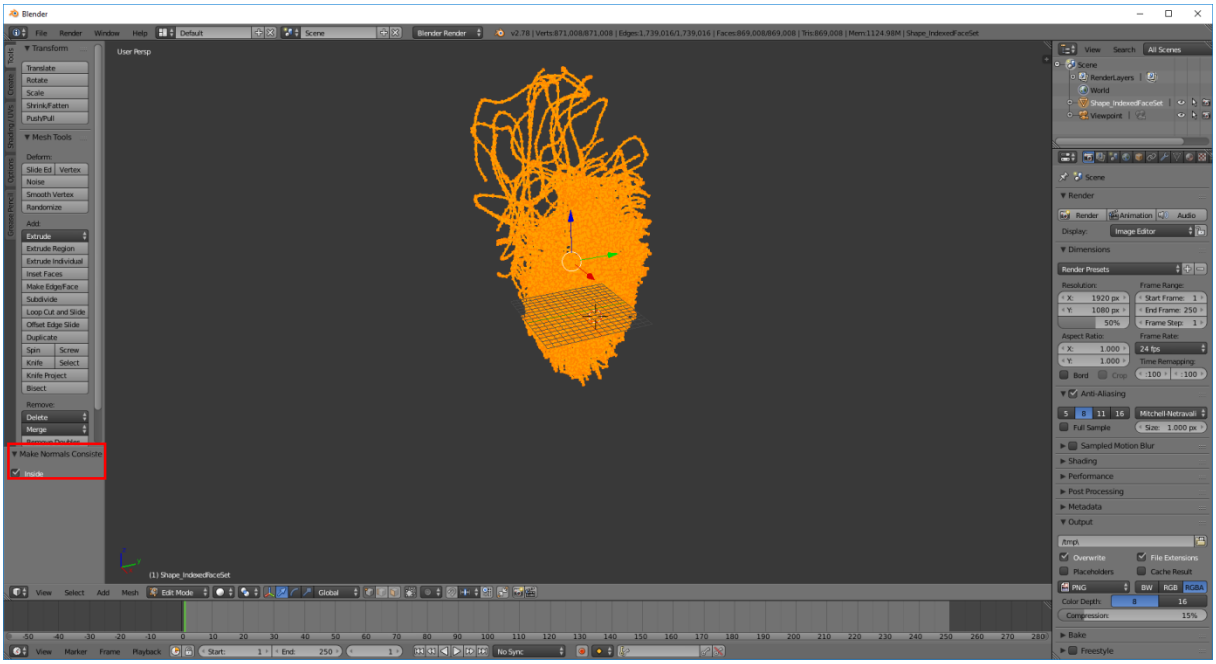

Figure 37: make the normals consistent by checking the box

5. Export the model as *.fbx so that the Unreal Engine can import it.

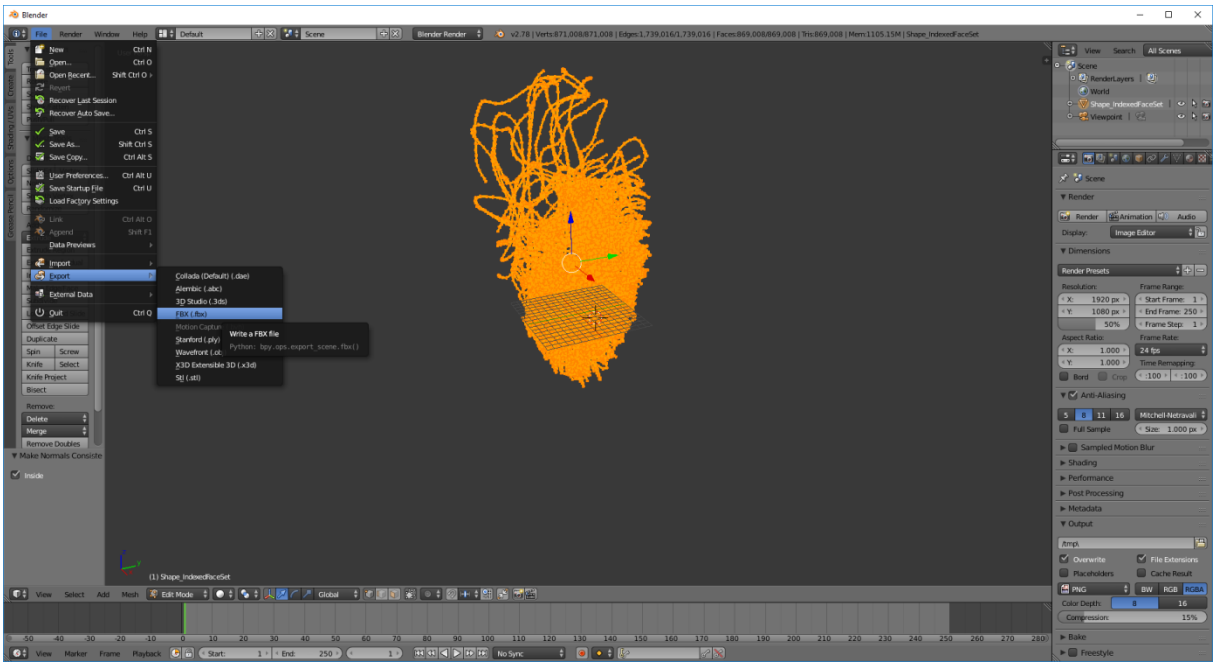

Figure 38: export the model

6. Make sure to set the export settings as pictured in figure 3.

7. Import the surface model as in step 2.

8. Delete the fibre network like the default items in step 1.

9. Save the surface model like the fibre model.

A.3. Unreal Engine

When importing the models make sure to have the import settings as in figure 39, especially the highlighted ones. Otherwise you get error messages and don’t have the fibre colourisation from DSI Studio available.

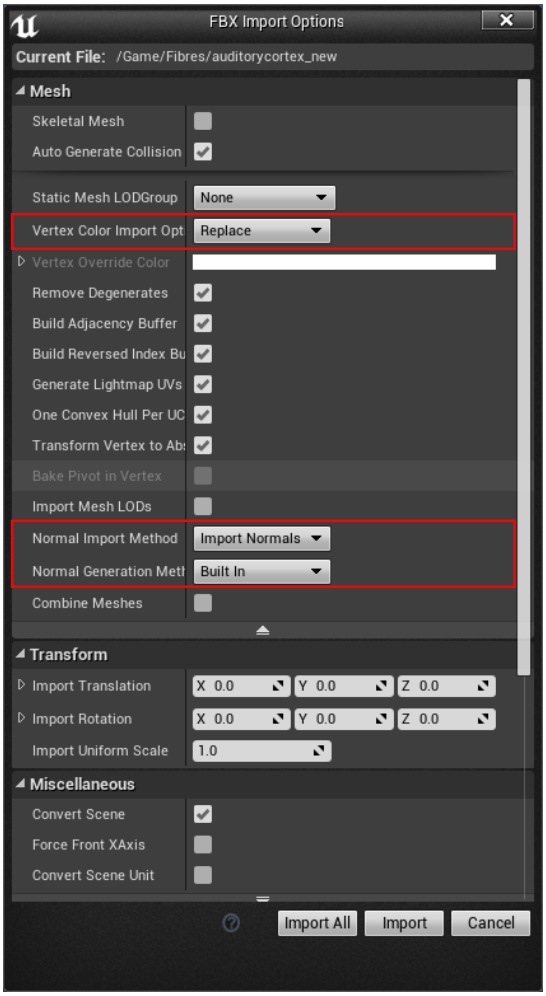

Figure 39: Import settings for the models in the Unreal Engine