TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

PETRI AHO

VISUALIZING MINING DATA

Master of Science Thesis

Examiner:
Professor Tommi Mikkonen
Examiner and topic approved by the Council of the Faculty of Computing and Electrical Engineering on 03.06.2013

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master’s Degree Programme in Information Technology
AHO, PETRI: Visualizing Mining Data
Master of Science Thesis, 44 pages, 0 Appendix pages
August, 2015
Major: Software Engineering
Examiner: Professor Tommi Mikkonen
Keywords: Mining, visualization, Python, Qt

Interpreting data collected from the mining rigs provides challenge which is alleviated with help of visualization techniques. A good visualization shows all the relevant information at a glance and helps make decisions. In mining the information can be used for example to follow the concentration of the mined minerals, adjust drill settings according the rock properties, track wear of drill bits, make drill plans, detonation plans and bolting plans in order to ensure efficient and safe work.

Visualizing methods include graphs such as line or bar diagrams, interpolation and extrapolation algorithms and mapping data spatially or temporally. When observing data within single hole or comparing few holes it is best to use line graph and plot in respect of depth or time. When trying to visualize the whole mining pattern, such as the tunnel face or drill field, the best way is to use 3D and draw the holes in the scene as cylinders or lines and color them along the depth with different colors representing different values of the visualized quantity. Interpolations and extrapolations can be used to spread the data in between and around the holes in the 3D view. Interpolations can be visualized by using planes with either color coded height maps representing the data that can be moved around the space or by using isosurfaces. Isosurface is visible where the data matches the examined value and by adjusting the value isosurface changes accordingly.

This thesis concentrates on finding the best ways to present the data from drilling machines. It has three main points: 3D visualization for analyzing large batches of collected drill data, real time drill visualization for use in the drill equipment during and immediately after the drilling and interpolation to find ways to interpret and expand from the data available. All the features in this thesis are built into a DrillGraph software, to be used as a basis for actual products.

Powerful computers available these days make it possible to visualize the 3D scenes in real time. It is easier to get the grasp of the drill field when the visualization can be panned, rotated, zoomed and moved around freely. In this thesis it was found that even a laptop can handle drawing real time visualizations of moderate size drill fields when the heavy drawing operations were done using OpenGL.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
AHO, PETRI: Kaivostoiminnan visualisointi
Diplomityö, 44 sivua, 0 liitesivua
Elokuu 2015
Pääaine: Ohjelmistotuotanto
Tarkastaja: Professori Tommi Mikkonen
Avainsanat: Kaivos, Visualisointi, Python, Qt

Porauslaitteilta kerätyn porausdatan tulkinnan apuna on hyvä käyttää visualisointitekniikoita. Hyvän visualisoinnin avulla voidaan nähdä oleellinen tieto yhdellä vilkaisulla ja toimia sen pohjalta. Kaivostoiminnassa visualisoinnin tarjoamaa tietoa voidaan hyödyntää muun muassa mineraalisuonien vahvuuden seurannassa, poran asetusten säätämisessä porattavan kiven ominaisuuksien mukaan, poranterän kuluman seuraamiseen, poraussuunnitelmien, räjäytyssuunnitelmien ja pultitussuunnitelmien tekemiseen. Nopea ja oikea päätöksenteko mahdollistaa tehokkaan ja turvallisen työskentelyn.

Visualisointitekniikoita on monia, kuten viiva- ja pylväsdiagrammit, interpolaatio- ja ekstrapolaatioalgoritmit, ja datan kartoitus ajan tai paikan suhteen. Kun tutkitaan vain yhtä tai muutamaa reikää, ne kannattaa sijoittaa viivadiagrammiin ajan tai poraussyvyyden suhteen. Kun taas halutaan tutkia koko porausaluetta, kuten tunnelinperää tai porauskenttää, kannattaa reiät visualisoida 3D sylintereinä tai viivoina avaruuteen kuten ne on porattu toistensa suhteen. Data voidaan tässä tapauksessa esittää väreinä jolloin eri värit kuvaavat eri arvoja. Interpolaatiota ja ekstrapolaatiota voidaan hyödyntää arviomaan visualisoitavan datan arvot reikien välillä ja ympäristössä. Interpolaatiot voidaan visualisoita 3D maailmaan käyttämällä 2-ulotteisia tasoja joihin on värein kuvattu data ja jota voidaan liikuttaa 3D porauskentässä tai siinä voidaan hyödyntää tasaarvopintoja. Tasa-arvopinta on kuin massaa, joka näkyy vain niillä kohdin visualisoitavassa mallissa, missä arvo vastaa tutkittavaa arvoa. Kun arvoa muutetaan, myös tasaarvopinta muuttuu vastaavasti.

Tässä diplomityössä pyritään löytämään parhaat mahdolliset tavat esittää dataa porauslaitteista. Datan esityksessä on kolme pääosiota: 3D visualisointi, millä pystytään esittämään isoja määriä kerättyä porausdataa, Reaaliaikanäkymät, joita hyödynnetään porauksen aikana sekä interpolointi, joka selvittää mahdollisuuksia tulkita ja laajentaa käytettävissä olevaa dataa. Kaikki ominaisuudet tässä diplomityössä ohjelmoidaan DrillGraph-sovellukseen, jota käytetään myöhemmin pohjana varsinaisille visualisointituotteille.

Nykyisillä tehokkailla tietokoneilla 3D visualisointi voidaan helposti tehdä reaaliajassa. Kun 3D mallia porauskentästä voidaan liikuttaa, pyöritellä ja zoomailla vapaasti, sen hahmottaminen helpottuu huomattavasti. Tässä diplomityössä havaittiin että jopa nykyaikainen kannettava tietokone pystyy suoriutumaan kohtuullisen kokoisen porauskentän 3D-visualisoinnista, kun raskaat piirtokäskyt suoritetaan OpenGL:llä.

PREFACE

Writing this thesis was an interesting project and quite a learning experience. In the course of the writing many parts of the software got refactored several times because the specifications kept changing and each time I feel I managed to improve on the previous iteration so it was visible during the project how my skills as a programmer were improving. The actual writing spanned lot longer than the actual software project it was made to describe. Somehow documenting the results is not as much fun as making them and when I did not finish the thesis immediately after ending the project, it was hard to get started again.

I would like to thank my excellent examiner Professor Tommi Mikkola for all the support and pushing he gave me to get this thesis done. Also integral in the making was Miika Huikkola who was leading the project and gave me this possibility. He had a vital role in supplying background information and specifications for the visualization needs and confirming the results.

Jani Savuoja and Markku Pusenius gave invaluable technical assistance with many aspects of the programming, taught me a lot and all in all were very supporting throughout the process.

Tampere 29.7.2015
PETRI AHO

TABLE OF CONTENTS

1. INTRODUCTION

Mining is the process of extracting minerals from the ground. It has traditionally been seen relatively expensive, slow and even dangerous while at the same time generating lot of waste. However modern mining technology has made big strides in making it much easier, faster and cost effective. Even small improvements at the early stages of the mining process can make big difference in productivity and profitability when it carries over all the steps. Waste management and other environmental side-effects have become more problematic as legislation regarding those have gotten stricter and yields smaller. Waste production can be minimized if the drill operator could minimize unnecessary drilling by detecting early if the drilling is not hitting the vein.

Other ways of increasing productivity come from data analysis. Proper analyzing tools can help increase the output and validate the results fast and reliably by making decision making easier. Knowing the characteristics of the rock and the behavior of the veins makes planning on site easier by directing the work more efficiently and avoiding unnecessary work. Researchers can make the work methods and tools more suitable for certain rock types or scenarios improving safety, productivity and predictability.

This thesis is about visualizing data collected by mining drills in order to increase efficiency and to understand the mining process better. Thesis emphasizes on the research needs, with some aspects of specific visualizations for drill operators, chargers, foremen and drill planners. The overall structure of the thesis is listed below.

Chapter 1 gives a brief description of the mining industry and explains the mining process. There is also some explanation of the different mining types and drilling machines used in those.

Data gathering and usage are aspects that are needed by the visualizing so the chapter goes through what kind of data is available, how it is used now and what it could be used for.

Chapter 2 gives an introduction to different visualization methods and gives examples how they may be used in the mining context. Interpolation and extrapolation methods and data types are explained and explored how they are used in the visualization.

Chapter 3 is about the design of the DrillGraph software which is used to prototype and demonstrate the different visualizations. The architecture and module structure is included here. The development tools are also listed here along with all the libraries and their versions.

Chapter 4 goes through the implementation details of the DrillGraph software. Data handling is explained including how it is loaded, used and stored. Implementation details of the visualizations and interpolations are explained, how they were made and what problems do they solve. Some discarded methods are also described.

Chapter 5 sums up the results of the visualizations. It goes through the development process and what was learned from it. Interpolation methods and their benefits are listed and brief mention is given of the new visualization methods that could be helpful.

Chapter 6 wraps up the process used in researching the drilling visualizations and sums up the results of the thesis. It also briefly explores what was good and bad in the methods used.

2. MINING OPERATIONS, NEEDS AND REQUIREMENTS

Mining is defined according to Castree as the activities associated with the extraction of natural resources from the ground. Extraction is either by subsurface tunneling or drilling, or by removal of the surface materials through quarrying, open-pit mining, or strip-mining, i.e. the systematic removal of large amounts of surface material (Castree et al. 2013).

Mining has played a vital role in the development of civilization and even with effective recycling of the materials the demand of the minerals has increased steadily as industry and consumption has expanded. Today's mining operations may require whole mining towns with associated infrastructure, including hospitals, air and seaports, power plants, landfill facilities, and roads (Spitz & Trudinger 2008). The mining equipment has improved considerably in recent years; the drill rigs have been outfitted with increasing number of computers and sensors. As a result a large amount of data is available from the mining operations which can be used immediately to monitor the progress and help operator in adjusting the drilling parameters. This can affect the immediate aspects of the mining, such as the efficiency and drill rig reliability, but also the following phases such as blasting, hauling and crushing when more is known about the structure of the rock and it can be handled in the most efficient way for the equipment available.

Furthermore the data can be used in for research and development of the mining methods and finding new ways to improve productivity in machine and tool design, workflow planning, safety and other decision making.

2.1 Mining Process

Mining is done by fracturing the rock into manageable pieces that can then be processed as the minerals are separated from the ore. Depending on the characteristics of the rock material, different approaches can be used. For softer materials such as sandstone it may be enough to rip it with mechanical cutting devices or breaking it with rock hammers. In most cases the most effective way of fragmenting the rock is blasting. To direct the explosive force into the rock, holes are drilled into it and the explosive charges inserted into the holes. This is called drill and blast method, which is illustrated in Figure 1.

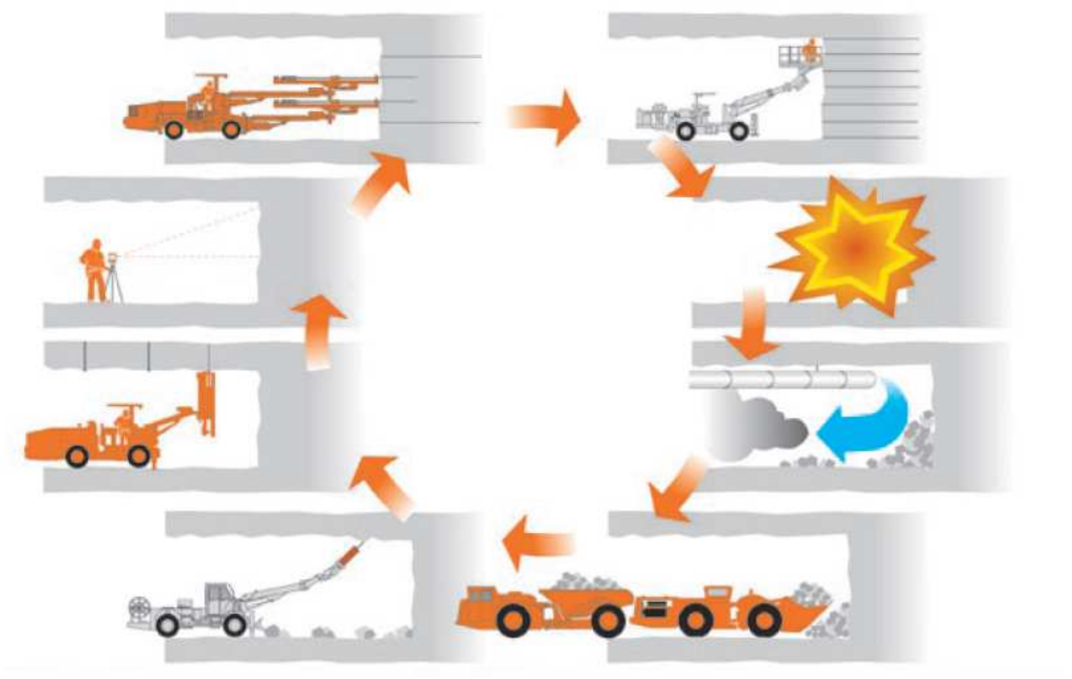


Figure 1 Drill and blast cycle in tunneling . (Heiniö 1999)

In the drill and blast method, blastholes are typically drilled using either rotary or percussive drilling methods. In rotary drilling, the rock is crushed and chipped by the teeth of the rotating drill bits. Rotary drill head contains three separate rotating parts which are pressed against the rock at constant pressure to crush it. It is necessary to keep the bearings clean from the dust and particles with air and water flow. Percussion drilling hits the drill bit against the rock causing minor fractures which then cause the rock to chip. The bit is rotated so that buttons hit slightly different positions during each strike. Water on air is used to clean the borehole and cool the drill. (Tatiya 2005)

In blasting phase the drilled blastholes are filled with explosives and in some applications sealed to concentrate the explosive force is to the rock. Blasting is a crucial phase to assure safety, generate suitable fragment size for the loaders and crushers available, e.g. avoidance of toes and other unwanted rock formations that would make it difficult to load the rock material and to continue drilling the rock face.

Correct drilling and communication between the drill operator and the blaster is important in assuring correct blasting. The drill operator should inform the blaster of any fragmentation and changes in the rock quality within the holes and possible deviations from the drill plan.

Too big charge can cause vibrations in surrounding areas. This can damage structures, send flyrocks and create shock waves that can cause danger, collapses and general instability in the mine and could make it difficult to continue and might require extra support structures. Too small charge can leave humps and too big boulders that are difficult to handle and may not fit crushers. Humps prevent proper loading and execution of the further drill plans. Manual hammering with rock hammers would be required to remove humps and break too large boulders.

After blasting the fragmented pieces of rock are loaded and hauled to crushers for further comminution then processed by various methods to increase the concentration of the wanted minerals and to separate minerals from the gangue. Processes differ for various minerals, but they depend on the physical and surface chemical differences between the extracted mineral and the other materials in the ore. Methods such as gravity concentration, froth flotation, electrostatic separation and magnetic separation are commonly used. (SME Mining Engineering Handbook (3rd Edition))

2.1.1 Drilling Equipment

Drill rigs for mining can be categorized in three categories: Rotary, tophammer and down-the-hole (DTH) drill rigs. The different methods are shown in Figure 2. The type of rig is selected depending of the rock type and hardness, diameter and the length of the holes.

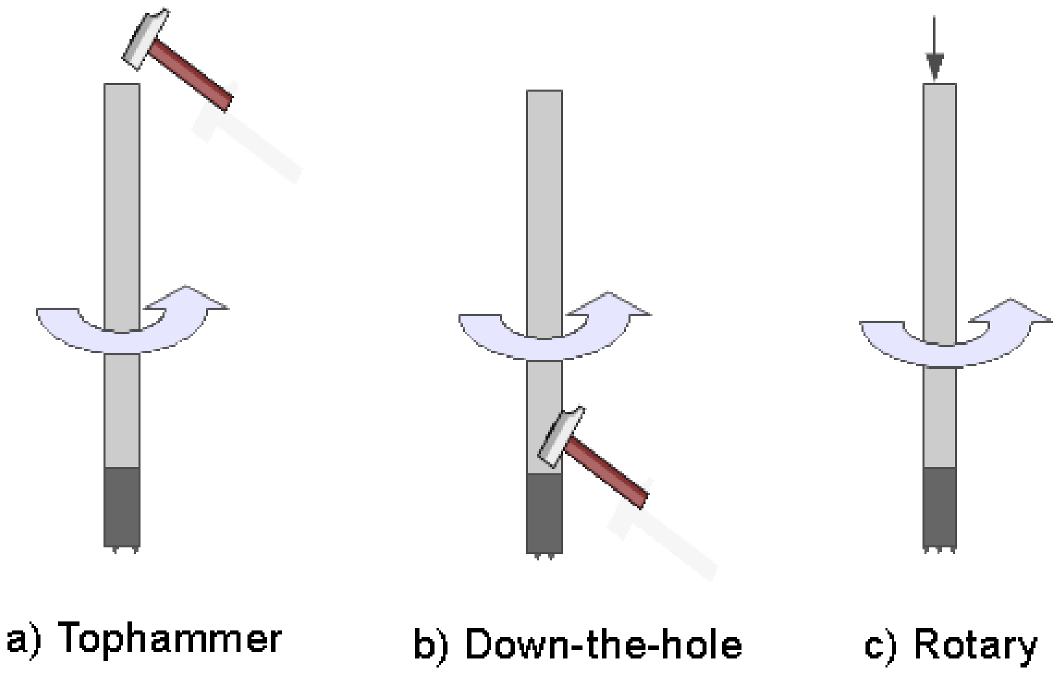


Figure 2 Drilling methods

Tophammer (Figure 2 a) has the piston at the end of the drill, where it hits the shank and the shockwave is transferred through the string to the drill bit and the buttons in the bit make the rock crack and crush. The feeding mechanism makes sure the bit is in contact of the rock all the time and a slight rotation is done between the hits so the whole hole area is worked. Percussive drill bits are shown in Figure 3a.

DTH (Figure 2 b) machines lower the percussion mechanism in the hole where it is in direct contact of the drill bit. This method can be used for longer holes, since there is no loss of energy to the rods. Percussion mechanism is operated by compressed air and the bits can be seen in Figure 3 b.

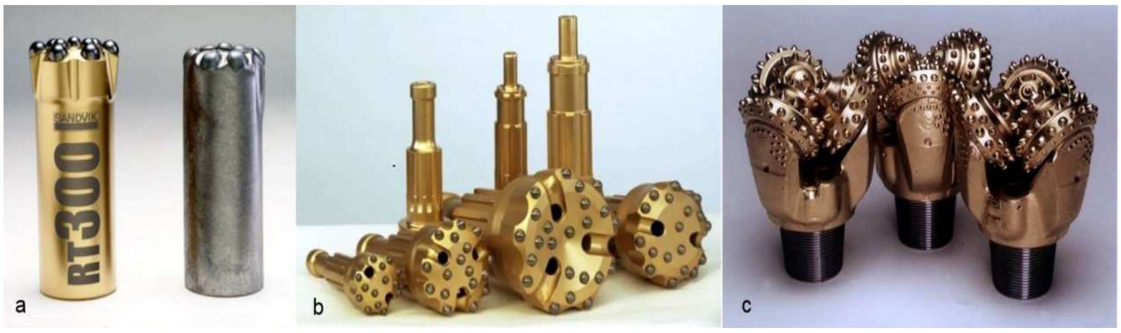


Figure 3 Drill bits for a) Tophammer b) DTH c) Rotary drills (Sandvik)

The rotary drilling (Figure 2 c) is used for softer rock materials like coal or limestone, and it relies on the crushing force of the teeth to break the rock while a constant, static force is pressing them towards it. Air is blown into the hole from within the string, which will cool the drill head and push the cuttings out. Drill head usually has three rotating heads with teeth that will crush the rocks as they rotate independently as seen in Figure 3 c.

2.1.2 Surface Mining

Surface mining includes different methods such as strip mining, open-pit mining, and mountaintop removal. In all methods it is first necessary to remove all the vegetation from the area, and then to remove the overburden or soil from the top of the mineral vein by means of bulldozers, excavators and diggers. The exposed ore can then be worked with suitable methods to extract the minerals.

Strip mining is most commonly used for coal but it is applicable for other minerals as well. Strip mining is done in long narrow strips, where the ore is removed. The waste from the next strip is dumped to the previous strip.

In open-pit mining which is shown in Figure 4, ore is worked in layers creating a large pit. Drilling is done by the benches expanding those outwards by blasting then continuing the layer below it. Result is a structure reminding of huge stairs leading up from the pit.

Mountaintop removal blasts the top of a mountain and works down from there, leaving the sides to cover the work. It is mostly used for coal.

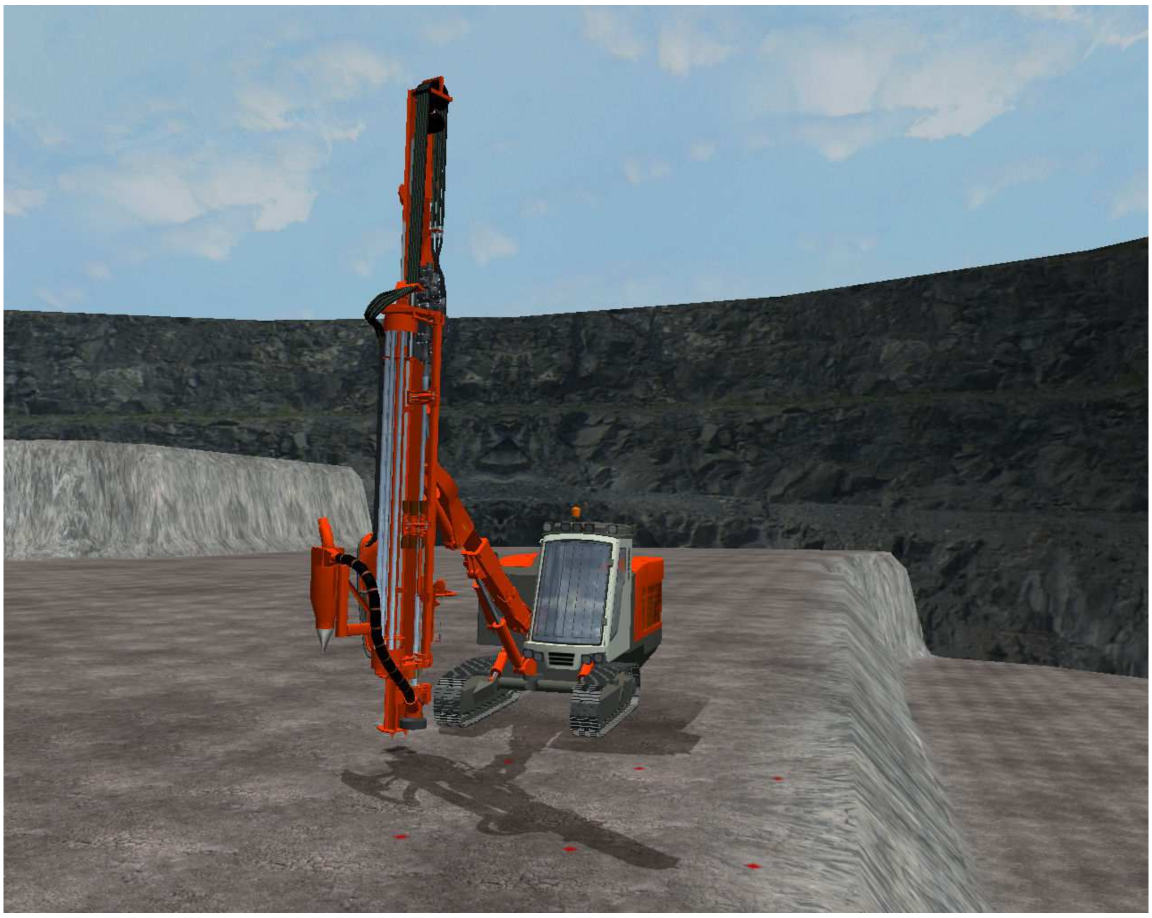


Figure 4 Simulator visualization of open pit mining (RigSimulator)

2.1.3 Underground Rock Excavation

Underground rock excavation typically includes tunneling, building underground space for civil use and mining, extracting minerals utilizing an underground space to have an access to ore body. Tunneling and mining have their own special requirements, but possess many similarities as well.

Underground mining includes all the mining that is done in excavated space underground. Underground mining can be done utilizing shafts, which are vertical tunnels, drifts that go horizontally and slopes which are declining.

In underground mining the mining process is mainly characterized by the selection of the mining method. As for example sublevel stoping.

Excavating the access tunnel is typically done by making a set of holes to the end face of the tunnel, as shown in Figure 5, filling the holes with explosives and blasting them. The rubble is then hauled away. Once the access tunnel is excavated and reached the ore the actual production drilling can start. In sublevel stoping the production drilling is typically done by drilling holes in fan-like pattern. While the holes on the tunnel face are usually only few meters long, the production holes can extend to 20 meters and over.

When operating underground, there is also need for drilling support holes, which are not blasted. Bolt holes are drilled perpendicular to the tunnel on the ceiling and walls and bolts are driven to the holes to support the tunnel and when necessary beams, nets and other support mechanisms can be attached to them.

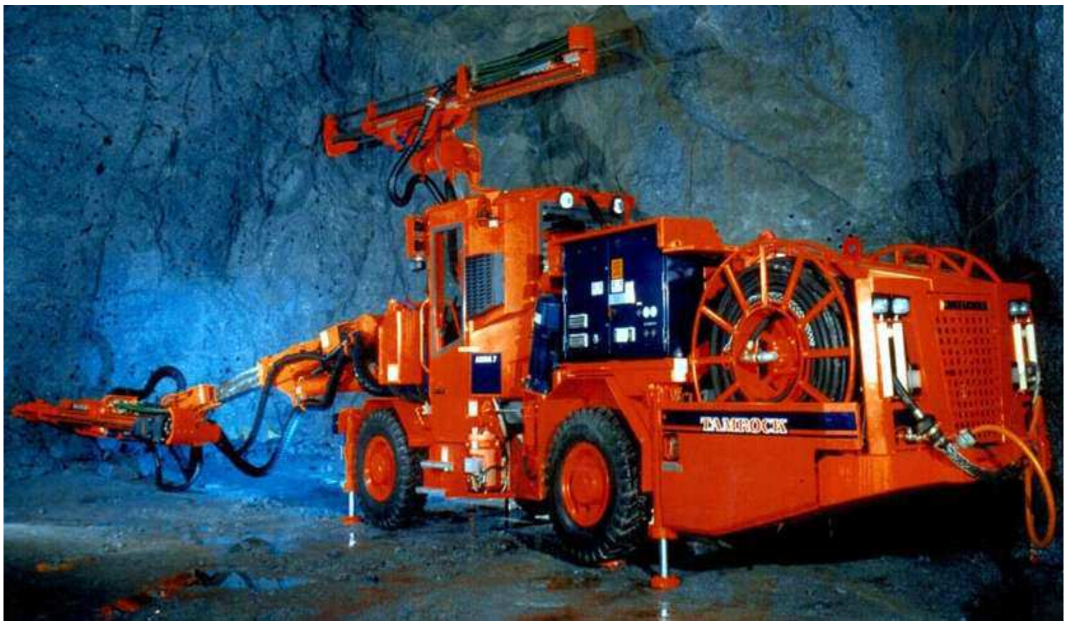


Figure 5 Tunnel face drilling

2.2 Mining Data

Data used in this thesis comes from the sensors in the drill rigs, but it is possible to get data from other machines and measurement devices too, such as laser scanners, density measurements, spectrometers and weight scales. Using the data visualization methods it is also possible to evaluate new measurements and possibly equip future drill machines with such sensors if the data is useful.

Sensors typically give information to the operator through gauges and meters in the control panel, but the data is also often recorded for later analysis. Pressure sensors are among the most typical ones that measure for example percussion, feed and rotation pressure. More specialized sensors are always being developed for better understanding and controlling the drilling process.

Drilling state is information that is important to know when analyzing the drilling data. State could include different drill settings and adjustments that are made by the operator or the automation. State could also include wear of the drill bit and amount of rods in the drill. When combining the state data to the sensory data it is possible to generate normalized data for the rock properties.

Operator can give important feedback to the process as well. Operator can often detect things that may be hard for machine to notice. There may be change in the tone of the drilling sound or in the color of the cuttings. Water might sprout from the hole when water vein is hit if air flushing is used and operator can learn subtle changes in vibrations to detect fractures as well even if the automation algorithms do not.

There are other measurements done outside of the drilling process that can be used for extra information. There are devices that measure the curvature of the holes, such as accelerometers which can be lowered to the surface holes. Also cameras can be lowered or driven into the holes to assess color and other visual characteristics of the rock. Laser measurements are used to map the walls of the tunnel or surface bench to create a model of the tunnel or mine surface.

The data can be stored within the machine's own controller and extracted with a USB-stick or other local means, but it is also possible to transfer the data further through local area network in real time to a workstation, server or a cloud.

2.2.1 Data Usage

Drilling data can be used by the operator himself to adjust drilling parameters and monitor how changing settings affect the drilling speed and equipment wear to maximize productivity. This requires the operator to use his or her own expertise to analyze the information presented to him and make decisions to optimize the drilling parameters.

Alternatively the data can be collected by drill manufacturers to improve their drill rigs, drill bits, automation settings and algorithms and other research and development. The manufacturer customer support and maintenance personnel can also check the drill data to see what was happening when problems occurred in order to give support and advice.

Another important scenario within the mining process is letting the charger know the rock properties for proper charging. Drill operators have traditionally given charger paper schematic of the drill plan where they have made markings of the hole depth and any problems such as fractures within the holes. With better drilling data it is possible to make better decisions for the blasting. Proper blasting makes loading and crushing the rock easier, does not endanger the structural integrity of the mine and ensures easy access of the drill rig to the following drill patterns.

Safety issues are important especially in the underground mines, where drilling data can give valuable information about reinforcement and support needs. Analyzing the drilling data could reduce the need for other tests or at least confirm the results reducing unnecessary reinforcement work and improving safety.

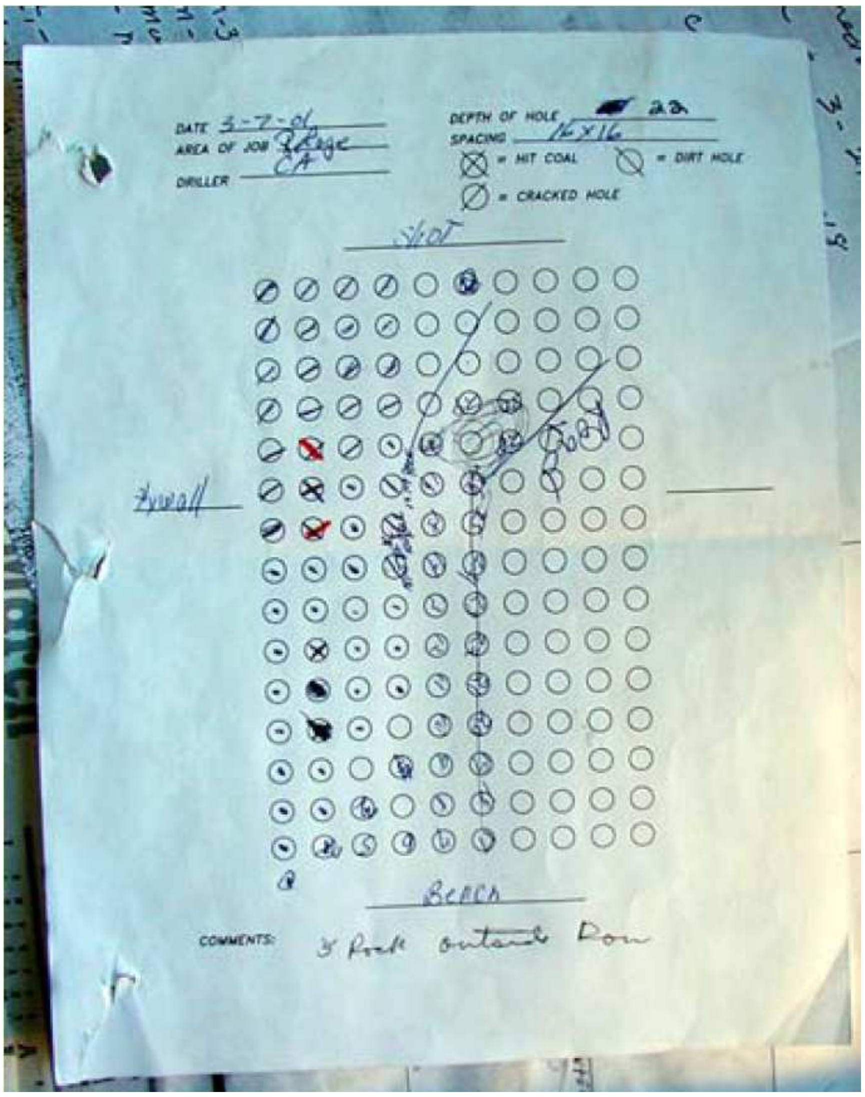
Geologic surveying often use separate sampling tools, but it is also possible to use the data collected during production drilling. Data could be used to estimate rock types and layering, hardness, abrasiveness and fragmentation level of the rock mass along with mapping the ore veins.

Site supervisors and planners get information about the work pace, conditions and behavior of the ore veins from the drilling data. The data can be used to alter plans and refine work methods and equipment in the mine for best productivity.

2.2.2 Visualization Needs

Most basic needs for visualization come from the needs of the operators. Especially less experienced operators can benefit from visualized information from the drilling data and possible interpolations and extrapolations based on the previous holes. Visualizations can be used to predict rock behavior so correct drilling modes can be selected and to warn about possible upcoming problems such as fractures or water holes.

Drill operators are usually required to make observations for chargers on the rock qualities, fragmentation and other abnormalities in the holes. An example of a drill log form is shown in Figure 6 where the drill operator has manually made markings about the holes that have been drilled. Lot of this can be automated by the controller and reports and visualizations can be generated to aid the charger in charging the block properly. Blasting report could show where the fracture lines go and where the rock is harder or softer.



DATE 3-7-01
AREA OF JOB Ridge CA
DRILLER

DEPTH OF HOLE 22
SPACING 16 X 16
⊗ = HIT COAL
⊘ = DIRT HOLE
∅ = CRACKED HOLE

Shot

Highwall

Bench

COMMENTS: 3' Rock outside Don

Figure 6 Drill log form

Visualization of the drill data can show weak points that would require extra support and could minimize or even eliminate need for water testing in injection holes in order to detect fracturing. Support visualizations could also include any detected fracturing and information about the brittleness of the rock material along with the hardness.

For mine supervisors and planners, visualizing mining data can give a better overview of the mining operation and better understanding about the ore behavior and rock properties. Analyzing data for the whole mining operation would be too inefficient and properly generated visualizations can show problems immediately.

Maintenance and customer support can easily check the visualized rock model and compare the drilling parameters to see if there are machine failures. This data can be used to determine if problems are due to faulty operating methods.

Researchers can create rock models based on real drilling data by help of interpolations and smart algorithms. Based on those models they can generate better drilling methods and improve the hardware and software of the drilling products. Simulators can be validated by comparing the simulator data to real data. The simulators can then be used to speed up the development cycle when new ideas can easily be tested and verified in simulated environments and the results of those tests again visualized for easier analysis that can be referenced against the real drill data.

3. DATA VISUALIZATION

Processing large amounts of numerical data can be challenging for humans, so developing methods for structuring and presenting the data in a way humans can understand and analyze it easier is important part of data processing. With proper visualization tools large amounts of data can be checked with a glance and anything out of the ordinary shows up immediately. Comparing the data with other data sets or with different data channels makes it possible to draw conclusions and to understand the operations and the relations between measurements from different sensors.

When making visualization software, it is important to know what kind of data is being visualized and what does the user want to see from it. It also makes a difference if the data is static or if it changes or accumulates while being visualized. There may be need to process the data for interpolation purposes or create derived attributes for visualization.

3.1 Data Types

The type of the data can affect the visualization methods and the algorithms used for processing it. Data type can refer to the units of the measurement, such as meters or kilograms which should not be mixed together, but can use regular arithmetic methods within the same units.

Alternatively values can represent different characteristics such as male, female, dog, and cat or in the mining context it could be different mineral types such as granite or sandstone. Differences between continuous and discrete data become significant when making calculations such as interpolating and extrapolating the data.

3.1.1 Continuous Data

Most measured units are continuous types, such that the values can be anything within the limits set by physics or measuring tools. These types of units can be processed with arithmetic units so it is possible to take averages or calculate integrals of the accumulated kilograms. This makes it possible to interpolate and extrapolate using usual methods such as averaging and fit them to a curve by methods such as least squares. Visualizing this type of data forms continuous lines without gaps or steps except those from the precision of the measurements.

3.1.2 Discrete Data

Another major data type is discrete data. Discrete data cannot be used in the same way as the continuous as the actual values may mean completely different things. It is common to use numerical values to denote such values in the computer and sensor equipment, but using normal arithmetic methods on the data usually results in nonsensical results.

For example there could be an algorithm that detects whether there is copper in the rock being drilled and copper is marked with 1 in the output, gneiss is marked with 2 and granite with 3. If the instrument or algorithm detects copper in one measuring point, and plain granite in another, in the data they are marked as 1 and 3, but it is not possible to interpolate the values and deduce there would be gneiss in between those measurement points.

3.2 Visualization Types

Data visualization is a very common software application and many solutions are already available for it. The challenge therein lies with finding the suitable ways of visualization and adapting them to the current problem. It is necessary to know what kind of data is being processed, if there is pre-processing required, what is the expected behavior and what kind of anomalies to look for.

Diagrams and charts can be very specific for different fields and applications to convey different information. Well selected visualization makes the important issues stand out from the less relevant information. Axes can have linear or logarithmic scale and they may be normalized. There can be additional information such as variance, certainty or shown in the chart.

3.2.1 Relative Charts

The most basic diagrams are line, scatter, pie, and bar diagrams, which are shown in Figure 7. They have somewhat different use cases and each also has many variations that have own unique distinctions.

Line diagrams are used when visualizing how a variable changes relative to another one. It shares the characteristics of scatter diagram, but generally the values on x-axis are truly increasing, such as time or sample number. In drilling context one might visualize any measured variable, i.e. Percussion pressure, as a function of time or depth.
Scatter diagram is more of a relation of measured values without connection. In some cases it is possible to generate line diagram from scatter diagram by finding trends in the relation of the measurements, such as least squares fitting.

On the other hand the diagram can show where the values are concentrated. This kind of data representation works well when comparing two measured attributes that affect each other to find the correlation. In drilling it could be for example the correlation between penetration rate and rotation pressure as in Figure 8.

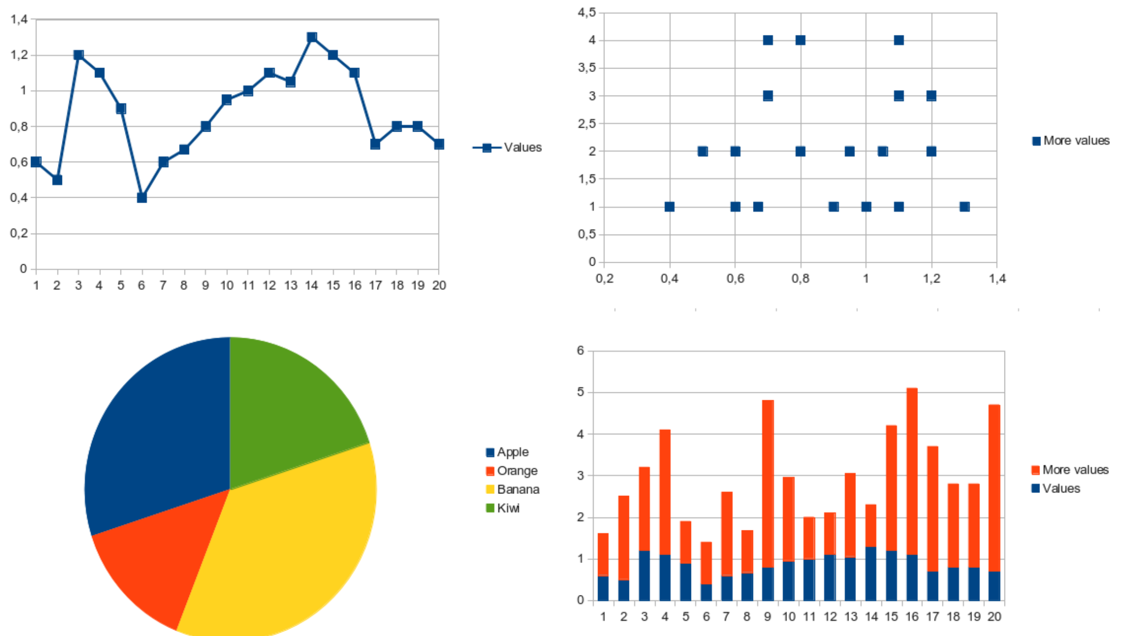


Figure 7 Line, scatter, pie and bar diagrams

Pie diagrams are used to depict portions or distribution of some values. This could be used to show the percentages of different rock types within an area or within a specific hole.

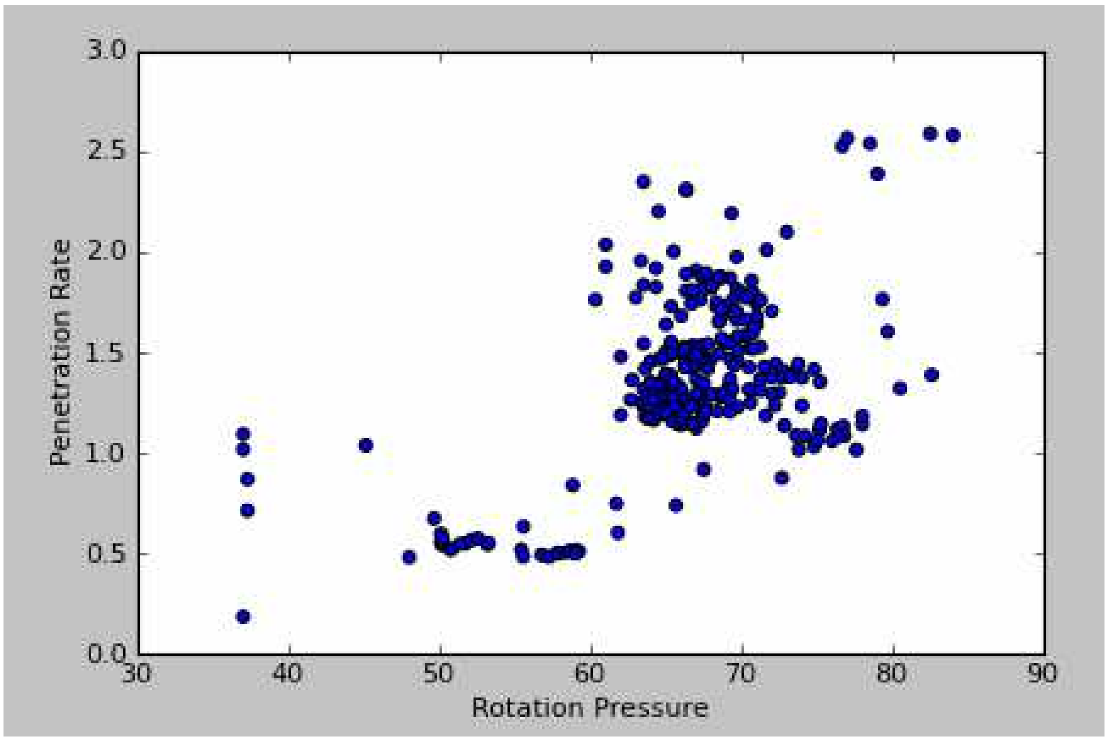


Figure 8 Scatter diagram

Bar Diagrams are good for comparing separate instances of an attribute and distribution of discrete values within those instances. There could be a comparison for number of factors. For example bars could be different kind of pets and the bar height could represent the number per thousand people. In mining scenarios the bars could represent different holes and their length and different colors within the bars amount of ore and non-ore rock.

3.2.2 Spatial Diagrams

When it is necessary to see the location of the data samples in relation to each other and other possible objects in the surrounding it becomes necessary to assign one or more of the coordinate axes to location. This brings new challenges for visualizing the sample values. In static diagrams even three dimensions are becoming more difficult to visualize and comprehend because of perspective. More than three dimensions are not possible by conventional means so other ways must be used to make the values apparent.

There are many ways to go about this depending on the type and density of the data samples. For example in weather maps, such as in Figure 9, it is enough to show the numerical values of the temperature (red numbers) or wind speed (blue circled numbers with an arrow to show direction) for the area under it or use symbols for more discrete type of data such as weather type in cloud, snow flake or sun symbols.

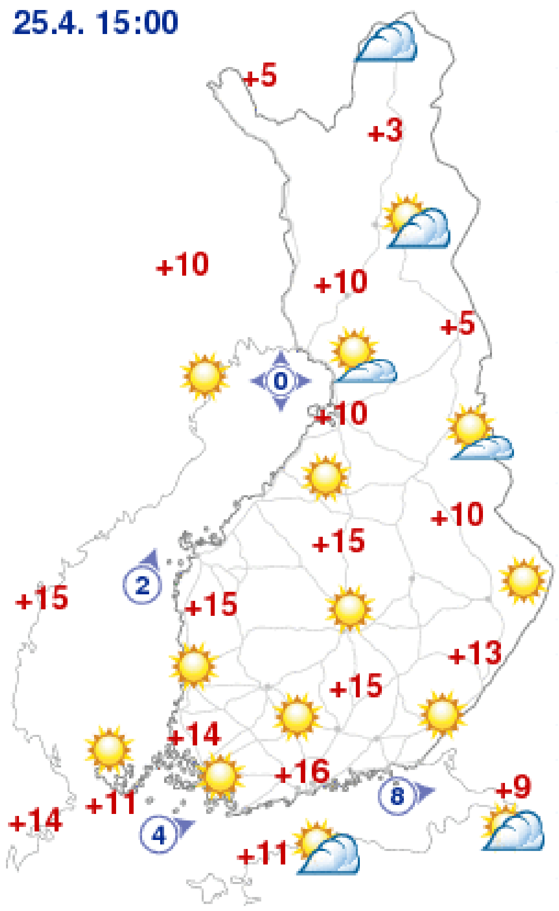


Figure 9 Weather map shows symbols and numerical data

When the data is more densely populated, the diagram or map would become too cluttered and impossible to read. Then the values can be mapped to a color and these colors drawn into the diagram. This can be seen for example in topographical maps in Figure 10. The height of the surface is visualized in colors that are explained in a separate color map. Contour lines are another way to visualize data in two dimensional graphs. It shows edges of an area that has same data. Most common example is height lines in maps, where it denotes an area of certain height. Data might have to be simplified for this kind of mapping to avoid crowding the figure, but it can be useful tool in many cases and easy to combine with other data.

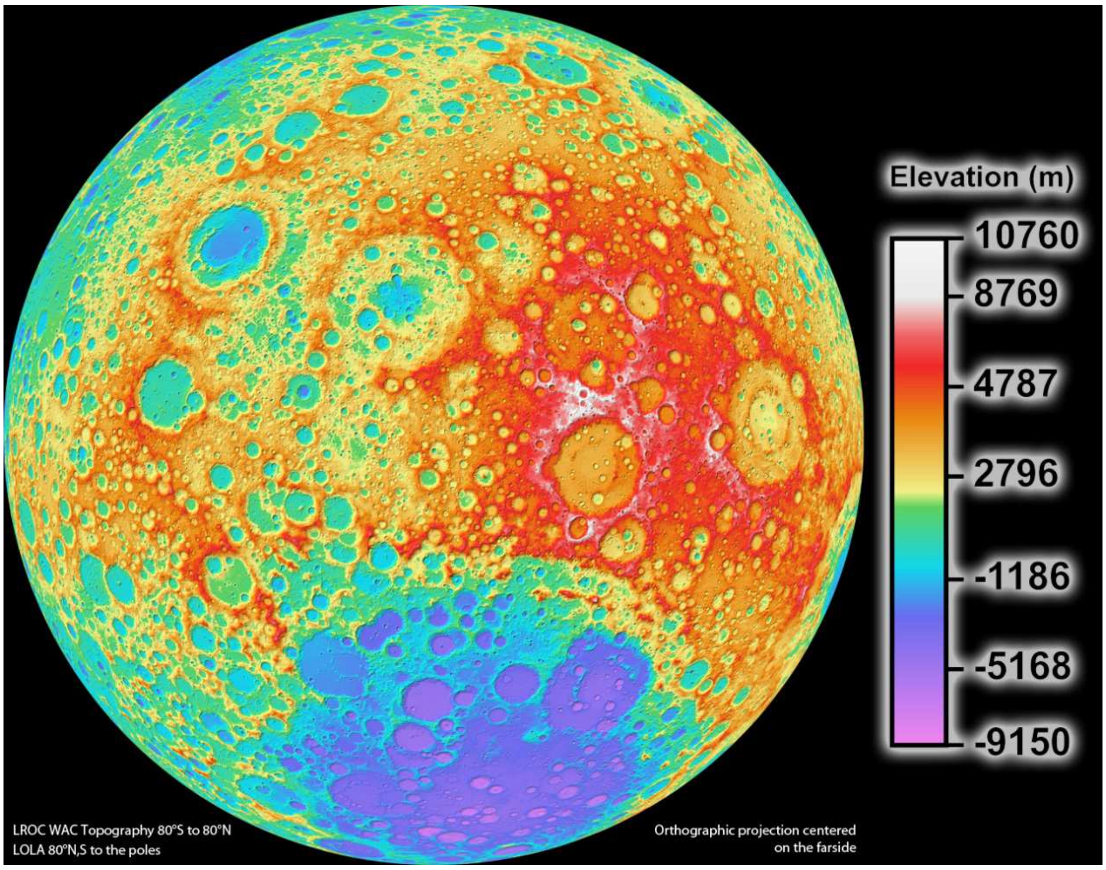

Elevation (m)
10760
8769
4787
2796
-1186
-5168
-9150
LROC WAC Topography 80°S to 80°N
LOLA 80°N,S to the poles
Orthographic projection centered on the farside

Figure 10 Topographic map of the moon using color (NASA)

3.3 Interpolation and Extrapolation

Interpolation is estimating values between measured values and extrapolation is estimating outside the measured range or beyond the end of the measured time. Estimations are used when planning drilling or estimating what is coming ahead while drilling to be able to prepare. It can give insight into placing explosives and make it possible to create geological models. In mines interpolation and extrapolation can be used to show inside the rock wall. For example the rock can be in layers and each layer may require different drilling parameters, the veins may go in thin stripes and minimizing the drilling outside of the vein can increase productivity considerably.

3.3.1 Continuous Data

Interpolation algorithms weight the relevance of different data points in relation to the interpolated point. Most commonly values are weighted with distance. The most basic method is Inverse Distance Weighting by Donald Shepard (Shepard 1968). In Shepard's Method interpolated value at point x is calculated from samples using function

$$u(x) = \sum_{i=0}^{N} \frac{w_i(x) u_i}{\sum_{j=0}^{N} w_j(x)}, \text{where} \qquad \text{(Eq. 1)}$$
$$w_i(x) = \frac{1}{d(x, x_i)^p}.$$

The weight is the inverse of the distance between the interpolated point and the data point in power p, where p is positive real number and can be used for adjusting how fast the significance of the value deteriorates with distance. A problem with the Shepard's method is the processing time needed when the data-sets get massive and interpolation resolution increases, especially with sparsely populated data clouds where most of the data affects the result.

Variation of the Shepard's Method where only the data points within a given radius around the interpolation point are calculated is called Modified Shepard's Method. Modified Shepard's is considerably faster since it only uses a subset of the data points to calculate the interpolated value. However, finding the data that is within the range for the algorithm could possibly deter from the benefits. Therefore selecting a suitable data structure with fast nearest neighbor search can make a big difference in processing the data.

Modified Shepard's uses the same formula as the Shepard's Method. Only change is in the weight function which is in form of

$$w_k(x) = \left(\frac{R - d(x, x_k)}{R d(x, x_k)}\right)^2, \qquad \text{(Eq. 2)}$$

to take into account the R-sphere around the point of interpolation. With an efficient data structure such as KD-tree (Bentley 1975), the algorithm becomes O(NlogN), which scales very well even with big data sets.

Another method commonly used in geostatistics is Kriging (Wackernagel 1998), which is a group of techniques used to interpolate using probability distributions. It is more complex and will not be discussed further in this thesis.

3.3.2 Discrete Data

When interpolating discrete values, the values are predefined and it does not scale in between these values. In some cases it is possible to use continuous interpolation methods by rounding, but this is a special case. In that case the values in between the two discrete values can be thought of as probabilities between these two, but it only works if there are only two values.

For example in mining the controller could detect if the drill is penetrating regular rock wall that is denoted by value 0 or if it is within the ore vein which would be marked with 1. Now it could be useful to generate a graph showing where in the wall the vein goes and extrapolate where we want to continue drilling. We could use Shepard’s method and round it to get an estimation of where the ore is most likely located. However if we have more values, for example 0 for granite, 1 for marble and 2 for iron ore, it is not possible to deduct that in between one measurement for granite and another of iron ore there would be marble.

Instead there are specific interpolation methods for discrete data. Three methods are investigated for this thesis. These methods are mode, nearest neighbor and classification tree. They are addressed in the following.

Mode means that the most frequent element in the data set is selected. This can be implemented by having certain radius around the interpolated point and calculate the mode for all the data samples within that radius. It is also possible to take into account covariance of the data samples so the interpolation better follows the general shape and weighting of the samples. Problem with the mode-based approach is that it can vary drastically based on the radius selected and also the cases where there are no data samples at all within the radius require special attention.

Nearest neighbor is distance based interpolation based on finding the nearest data sample for each interpolated point and assigning the value from the nearest sample to the interpolation point. This is a simple and fast method, but the results can be somewhat skewed when using Euclidean distance. Instead using measurements that take into account the covariance of the samples such as Mahalanobis distance (Mahalanobis) give much better results.

The last method is a classification tree, which functions somewhat differently from the previous methods. It can be used to deduct the result from several measured variables traversing through a binary tree where each node contains a Boolean condition according which the path is chosen. Resulting value is located at the leaf of the tree.

3.4 Reliability of data

There can be errors in measurements and there can be errors in the algorithms used in interpolation. The reliability can be taken into account in the interpolation itself by using different weights depending in the certainty of the correctness or it is possible to calculate separate uncertainty for each data point.

If there are many samples and only one or few show considerably different values that do not fit the curve it can be determined there may be a measuring error. However this may not always be the case. In the mining it is possible there may be fractures and other anomalies in the rock that can cause spikes in the measurement data. This could possibly be detected also in adjoining holes that could give more reliability to the differing values or further discount them as an error.

In interpolation and extrapolation there are several factors that can affect the uncertainty of the interpolated point. The most important is the amount and distance of the data samples in the vicinity. Also, if the samples surround the interpolated point or if they are only on one side can be a factor as is the variance of the nearby points. For example if there are 3 data samples nearby but they all have considerably differing values the interpolation is less certain than it would be if all the samples would be same. Finally the certainty of the actual data itself has an impact to the interpolation as well. If the error margin of the data is high, the certainty of the interpolated values cannot be high either.

3.5 Visualizing Interpolation

Once the interpolations have been calculated there are number of ways to present the results to the user. Depending on the application it may be enough to show merely the prediction of what lies ahead of the hole currently being drilled. That could be shown simply as a line graph or colored bar. For two-dimensional visualization such representation consists of a plane using height maps or coloring. Usually due to the three dimensional nature of the rock being drilled height is not as intuitive visualization as colors and from the plane it is easier to see the exact location of visualized values. To visualize a space up to three perpendicular planes can be used and independently placed along each axis to visualize any point of the interpolated space.

4. DRILLGRAPH DESIGN

DrillGraph is piece of software developed for reading various drilling measurement data and to visualize it. Development of DrillGraph was done in prototyping fashion, with the main focus on coming up with new ways of visualizing drilling. Visualizations were created and iterated on to find what things work and how different ideas look. Later on the findings of the DrillGraph development were utilized in the specification and design of the actual product.

DrillGraph was designed to work with both real time data and previously recorded drilling data files. These are somewhat different use cases, since the real time data is something that usually is only accessible to the operator himself, so he would need tools and views that would aid the current drilling process. The previously recorded data on the other hand can be used for analysis of the work done later on and possibly aiding with planning and adjusting the process during the mining operation. Collected data is also used for research in the drilling equipment and methods as well.

However as the drilling gets more automated, it is possible the operator is moved to an office monitoring multiple machines at the same time. This could change the focus from the immediate control of the machine to more on the go planning and give more possibilities for analytical process control.

4.1 Tools

Few main points rose during the choice of the development tools. The client owns the source code, so they had a say in the development language. Since the software was only meant for internal use it was not necessary to obfuscate the implementation details.

Libraries and software components were chosen with licensing that allowed commercial use in case the software is ever distributed or used as part of commercial products.

Target platform was Microsoft Windows operating system ranging from XP to Windows 7. However in the end the tools used made it possible to run on Linux and Mac computers too, but those were not officially supported or tested thoroughly. 32-bit tools were used for wider support.

4.1.1 Qt

Qt is a programming framework that offers mainly graphical user interface related libraries for cross-platform development that supports Windows, Linux and Mac environments.(Blanchette & Summerfield 2006) Qt has several benefits for GUI design such as graphical design tools and vast selection of ready GUI components that have a native look in the target operating environments. For more specific GUI needs there is also Qt Quick, which uses declarative QML language for describing the user interface, but is still completely compatible with the rest of the Qt code. Qt has its own signal-slot system, which allows components to send signals that any other component can bind into and get called whenever the signal is sent e.g. when user interaction happens or data changes.

Besides GUI components Qt also offers various other useful features. Threading support makes it easy to make concurrent programs and to communicate between the separate threads using asynchronous signals. Network support offers high level access to sockets and makes it easy to create or connect to network services. Database services offer easy access to various data storages and built-in model-view-delegate system can bind the data to views and interact with it. Graphics allow drawing on the GUI components or separate buffers using OpenGL(Shreiner & The Khronos OpenGL ARB Working Group 2009) , 2D painter or separate Graphics view framework which allows more complex scenes and graphical objects to be drawn than would be possible with simple paint operations. Qt also has a resource system that makes it easy to embed images and other resources within the program itself. Finally there is extensive unit testing framework that helps testing all the Qt functionalities.

Features that were utilized in the making of this thesis were network, threading, OpenGL and drawing tools. Network tools were used for making a server that listens for real time data. Threading makes it possible to separate time consuming operations such as interpolation to another thread so they do not affect the responsiveness and other operations. OpenGL and 2D drawing tools were used for drawing or viewing all the visualizations. Qt version 4.7 was used in DrillGraph software.

4.1.2 Python

Python (van Rossum 1995) was chosen as the implementation language. Python is an interpreted language. It does not have strong typing and does not require compiling. Therefore it fits well in the type of project that does not have a rigid specification, but instead is based on trying different things out and finding out the best solutions. It also has a big selection of libraries for scientific calculation, visualization, and UI design. Python 2.7.1 was chosen and the minor versions were updated during the project. In the end DrillGraph was shipped with Python 2.7.3.

There were two different Python wrapper libraries available for Qt; PyQt and PySide. PyQt is the older and further developed, but PySide was chosen because of the more licensing was more permissive and fit the needs better. On top of that, it was the one developed by the Qt crew themselves. The most recent stable version of PySide at the time was used and it was shipped with version 1.1.2.

DrillGraph includes lot of data processing and scientific calculations so it was necessary to use SciPy and NumPy, which include wide variety of useful mathematical and scientific functions. Most useful features in the SciPy were array and matrix classes and huge variety of mathematical functions that operate with those. SciPy offers packages for interpolation, signal processing and Fourier transforms, linear algebra, spatial data structures and algorithms, statistics and image processing. SciPy version 0.11.0b1 and NumPy version 1.6.2 were used.

Matplotlib was another library that added visualization tools and plotting interface similar to Matlab. Matplotlib takes advantage of SciPy and NumPy and offers ways to create plots and graphs of the data calculated with them. Version 1.1.0 of Matplotlib was used.

3D graphics were drawn using PyOpenGL version 3.0.2a5. Using OpenGL made it possible to use GPU to accelerate heavy graphical drawing functions. Shaders are programs that are run on the graphics card instead of CPU which is highly parallel architecture and allows efficient calculation of algorithms that can take advantage of parallel execution. This allowed it to transfer some heavy calculations to the GPU to take the load off the CPU. Downside of OpenGL shaders is that the program is evaluated every draw cycle, which happened every 1/30 seconds. There are ways to utilize the processing power of the graphics card outside of the draw commands too such as OpenCL (Stone et al. 2010). PyOpenCL version 2012.1 was experimented with on heavier interpolation tasks that would be easily parallelizable. It was found useful, but other things were prioritized over efficiency, so in the end it was not used in the software.

4.1.3 Development environment

Microsoft Visual Studio 2010 Professional was used as the developing environment with Python Tools for Visual Studio plugin. Visual Studio was the default programming environment used in the company, so it was the logical choice and the Python tools worked very well.

Other tools used or tried were IPython interactive shell for testing various Python related code fragments or syntax outside of the actual software. It was found extremely useful for checking what and how various libraries worked what kind of parameters they took and what they returned, and to test small snippets of code without having to run the whole program. Subversion version control was used with TortoiseSVN and AnkhSVN clients. WinMerge was used for comparing between versions of code files. PyLint version 0.25.2 was tested for code analysis, but it was never taken in serious use. Modulo 2.2 was used for UML-designs.

4.2 Architecture

The basic requirement of the software is to read data and present it in various forms. Model-View-Controller architecture (Krasner & Pope 1988) is one of the most commonly used design patterns and it is specifically designed to separate presentation from the data, so it is very fitting basis for the DrillGraph. It is also well known and Qt has implementation structure ready to support variation of it. Module structure of DrillGraph is illustrated in Figure 11.

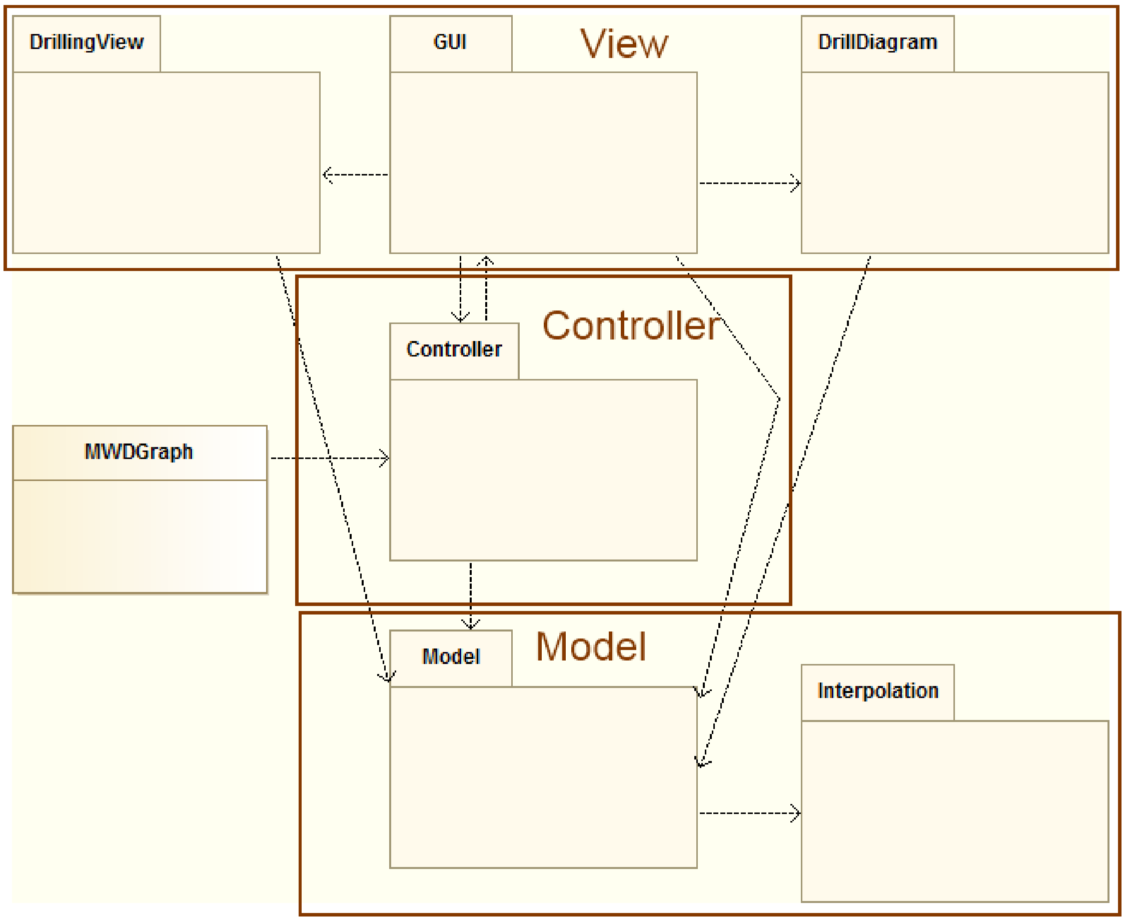


Figure 11 Module structure of DrillGraph software

4.2.1 Model

Model handles loading, storing, saving and processing the data for use with the rest of the program. It can read the XML-files consisting of the drilling measurement data and save it in a unified project file that can include multiple XML-files and any interpolations that have been already calculated for them. Class diagram for the model is shown in Figure 12.

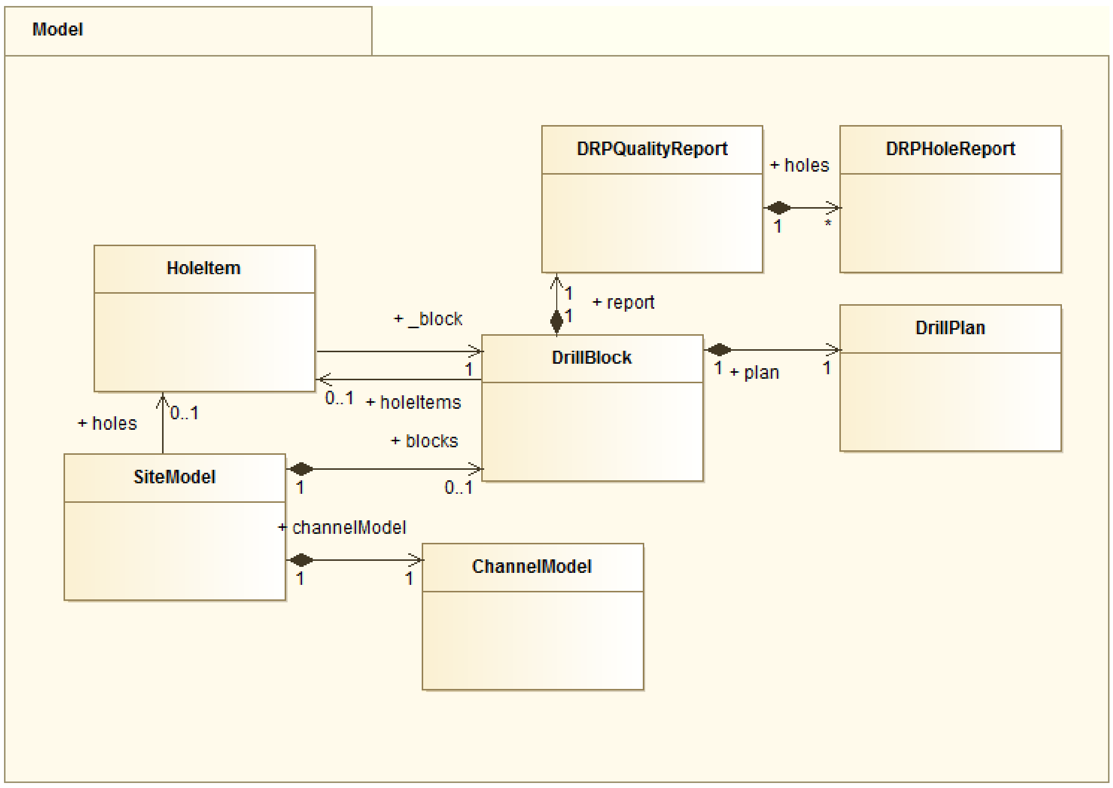


Figure 12 Model class diagram

DrillBlock represents one drilling report that usually equals to one drilling plan. These XMLs are loaded under the DRPQualityReport and DrillPlan classes that are part of the DrillBlock. DRPQualityReport also includes set of DRPHoleReports which are the individual holes and the drilling measurements within that report. The hole reports are also saved in separate XML-files.

SiteModel is the largest level model that represents the whole drilling site or the part of it that is being inspected. It may be one level of drilling field at the surface or one tunnel in the underground mine and includes all the DrillBlocks within that site. It also offers selection model to the views and can return holes or blocks according to the selection.

ChannelModel represents all the measurement channels within the holes. It takes care of all the channel specific tasks such as color schemes and value limits. It is same for all the blocks and holes within the site.

4.2.2 Controller

Controller module contains all the program logic. It owns the other components and handles user inputs then passes the commands forward to the other modules. Controller class diagram is shown in Figure 13.

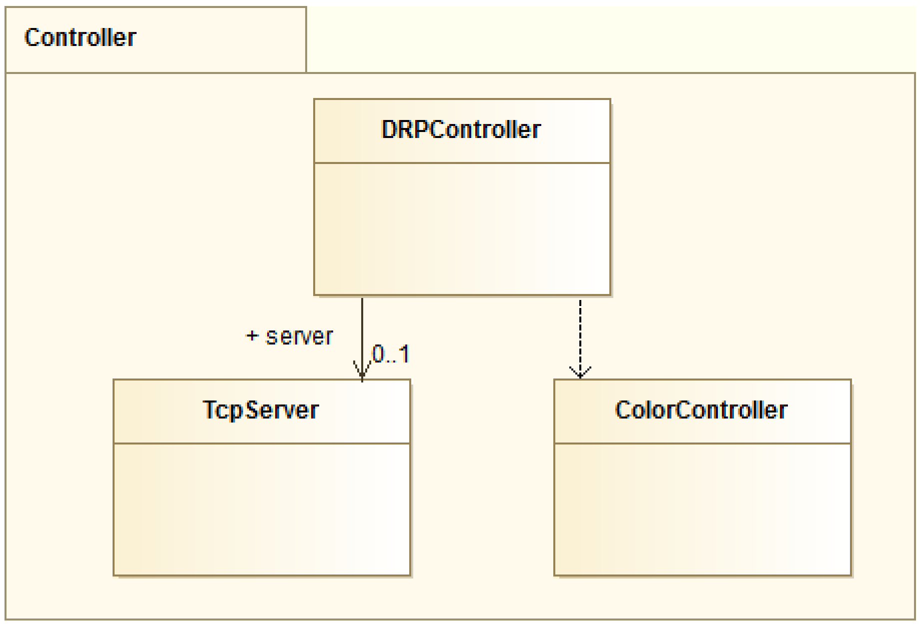


Figure 13 Controller class diagram

DRPController is the main class which is called from the program starter. It initializes the program and creates the View and Model objects. DRPController consists mostly of Qt signal handlers that are actuated from other modules or user input.

TCPServer listens to the configured port for real time measurement data according to the communications protocol (Appendix 1) and relays the information to the model by Qt Signals. TCPServer can handle one connection and it can be started and stopped.

4.2.3 View

View consists of three separate modules as shown in Figure 11: DrillingView, GUI and DrillDiagram. DrillingView and DrillDiagram are separate visualizations within the GUI and GUI contains even other diagrams, but these have been separated to their own modules due to their size and many classes that are part of the same diagram.

The main window, menus, buttons, dialogs and other controls are all part of the GUI. Class diagram for the GUI is shown in Figure 14. GUI module mostly takes advantage of Qt dialogs and controls. UIWindow is the active part of the main window which owns all the UI-controls. Mapview is a 2D spatial view of the holes and PlotView depth based line diagram for one or more channels in one or more holes.

DrillingView is prototype view for real time view for the drill operator. Class diagram for the DrillingView module is shown in Figure 15. DrillingView shows currently drilled holes and how the drilling progresses. It shows the desired channel information within the hole, the depth of the hole and it is also possible to show estimates of what’s coming ahead by interpolating from the nearby holes. DrillScene listens to the Qt Signals from the Model about new holes or data points and updates the view accordingly.

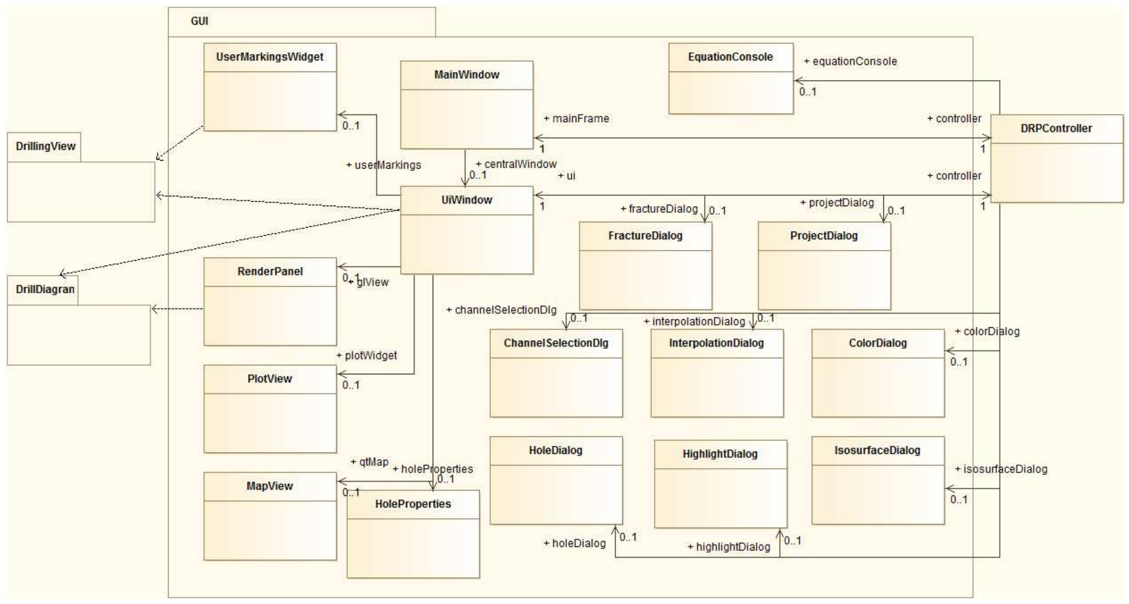


Figure 14 GUI Class diagram

It is also possible for the operator to make his own notes to the hole at desired depth by clicking the hole, which creates a new MarkingGraphicsItem at that spot and sends the information to the model. A discrete operator channel is also added to the model which is controlled from DrillingView by radio buttons which can be set to some values, for example to indicate different color of dust or different sound from the drill. Whenever operator changes the value of the radio button, the model starts recording that value to the operator channel.

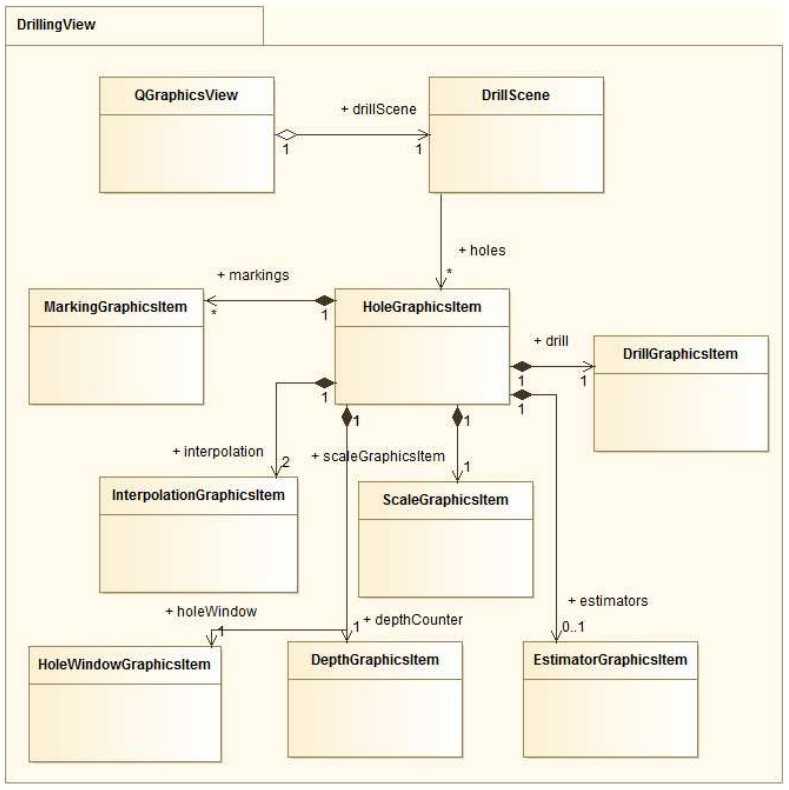


Figure 15 Drilling view class diagram

DrillDiagram is 3D view of the site. It can draw either all the holes in the site or just selected blocks using user defined color schemes to depict changes in the selected channel along the length of the holes. It also can visualize interpolations with planes that can be moved along the axes or using isosurfaces to show the areas where the interpolated value is within some range. Detected fractures within the holes are shown and can be combined into a fracture planes. Class diagram of the DrillDiagram is shown in Figure 16.

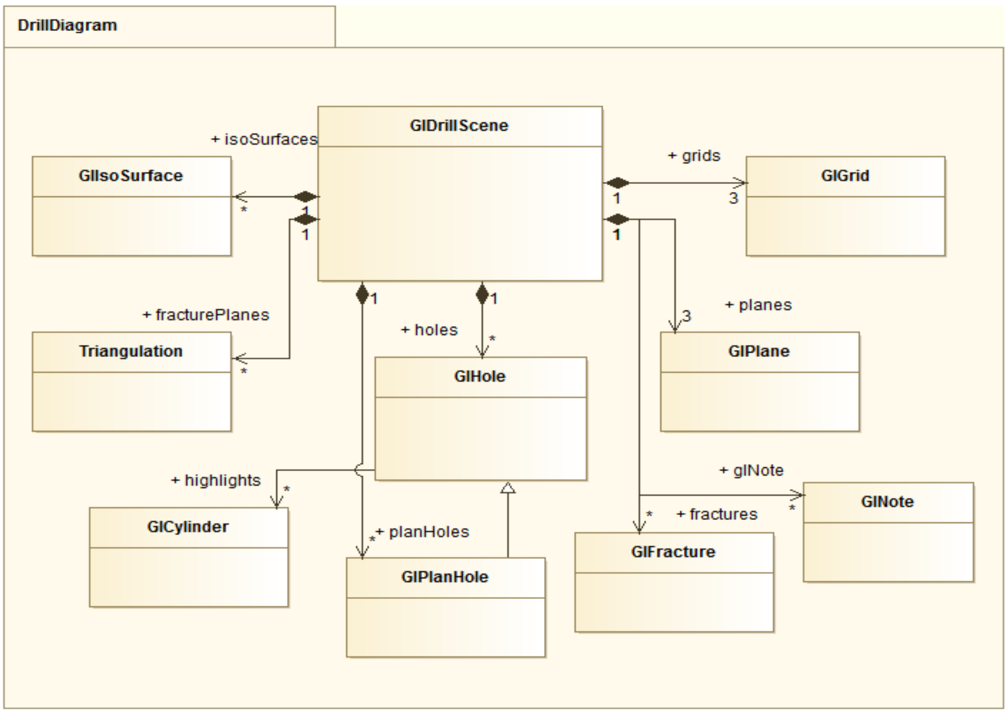


Figure 16 DrillDiagram class diagram

GlDrillScene is the main class for the module and offers the interface outside of the module. GlDrillScene takes a reference to the model in constructor parameter so it can fetch the hole information directly as it needs. It also connects to the Qt signals in the model to detect changes in the data or selection so the diagram can be changed accordingly.

GlHole represents a hole object that is drawn to the diagram. GlHole will be given the data vector of the visualized channel and each data point within the hole is colored according to the color scheme if the hole is selected or shades of gray if it is not.

GlPlanHole represents where a hole should be drilled. GlPlanHole does not contain any drilling data so it will be colored purple when selected or white when not.

GlIsoSurface handles the drawing of the isosurfaces. The user selects the desired value within the value range in the channel and +/- tolerance for it and GlIsosurface fills all the space where the interpolated value is within that tolerance of the selected value. There can be several GlIsoSurfaces at the same time, each with different color and value.

GlPlane is an interpolation plane. There are three planes and they can all be shown and moved individually along each of the axis they are perpendicular to. Colormap of the interpolated values at that plane in space is shown in the GlPlane and it is updated in real time as the plane is moved.

GlFracture shows red discs on the hole where fractures are detected. It is possible to create fracture surfaces by selecting a new fracture from the fracture dialog and clicking these fracture discs.

4.2.4 Inter-module communication

Inter-module communications are handled either by direct function calls or indirectly by Qt signals, a mechanism that Qt uses. Each module should have one interface class through which all the inter-module communications is done to keep the dependencies down. DrillGraph is merely the starting point for the software; it creates the DRPController object and starts the event loop. DRPController has the ownership of the other modules. It creates UiWindow that initializes all the views, SiteModel, which sets up the data model, and TCPServer to start waiting for clients.

A reference of the model is passed to the views and TCPServer. Model itself does not depend on the other parts of the software except the interpolation module. Interpolation is a separate module from model, but it is only accessed through the model, and interpolation information is saved in the project file.

The model interface affects all the other parts of the software, so it is important to keep it stable. Core interface includes getting hole names, coordinates, channel names and data vectors, and various meta-information. Bounding box and transformation matrix to transform holes into local coordinate system are also calculated in the model. The core interface is kept the same and it is expanded as new functionality is added.

Real time communication offers interface for machines to send drilling data through TCP/IP connection to the DrillGraph. It implements server-client model, where DrillGraph acts as a server waiting for clients, which can be drilling machines, simulators or other sources of drilling data. Protocol is text based with messages sent as strings of ASCII characters. Each command and its parameters are separated by space and the message is terminated by a carriage return. The client sends commands to the server and server only sends acknowledgments back to the client. Only one client connection at a time is supported.

4.2.5 Load balancing

DrillGraph is designed to be run in several threads. The graphical user interface runs in the main thread along with the main event loop. Majority of the program logic is run within the main thread. Interpolation thread does the interpolation calculations in the background without affecting the usability of the program. The intermediate results of the interpolation can be asked at any time and visualized. TCP/IP server also works in a separate thread so it can work separately from the rest of the program logic.

Parts of the heavy visualization routines are offloaded to the GPU using the shaders to do the calculations. This code is run every drawing cycle, so it is only suitable for specific cases such as calculating the isosurfaces. OpenCL also takes advantage of GPU outside of the drawing routines and using it was considered as well.

5. DRILLGRAPH IMPLEMENTATION

The first phase was to parse IREDES type quality and drill measurement files and show the holes and channel data in the graphical user interface. This iteration included the Main window module with list widgets for holes, channels and for data values of the selected hole and channel. Using this basic view it was possible to implement parsing of the IREDES files and to validate the data was correct. Once the data model, that loaded, stored and served the information, was implemented it was possible to start adding different visualizations for the data. As the complexity grew, majority of the logic was moved from the main window class to the DRPController class.

Since Qt was chosen as implementation framework, Qt-style View-Model structure was used. When the model is inherited from the QAbstractItemModel it automatically sends signals whenever data changes within the model and the views all stay synchronized with current data. This supports the architecture where model does not need to know what views, if any, use it. It merely sends the signals and the views connect to those signals if they want to get updates.

Most of the program runs in the main thread along with the Qt event loop, but heavier operations have been moved to separate threads. Parsing the IREDES files is done in a separate thread that sends pulses to the progress bar in main thread about the progress. Also interpolations are done in another thread so it is possible to calculate interpolations in the background without it affecting the other use of the program. TCP Server had to be moved to another thread with a separate receive buffer as more and more real time visualization was added and it could not be guaranteed that the server could keep up with the sender in all situations. Some of the error checking and reporting to the sender was lost as a result since it was not possible to validate the data between the incoming samples, but it was considered necessary to make sure no data is lost.

Threading is done using QThreads and communication between threads happens with asynchronous Qt signals. There were many ideas for optimizing the interpolation using multiple cores or GPU, for example with OpenCL, but in this prototype software it was more important to work on the visualizations than the interpolation algorithms.

5.1 Data handling

Data is handled by the model module which is described in the Figure 12. Data comes in either from IREDES files, existing project files or over TCP/IP connection. Currently all the data is loaded in the memory. This may become problematic with larger sites, which would require keeping it on the hard drive and loading as needed, but that case is not considered in this thesis.

5.1.1 Loading and saving data

To load an IREDES-report into the system, user chooses Load Report from the File-menu and selects a report file. This creates a temporary ThreadedReportReader-object that handles the report loading in a separate thread to keep the user interface from freezing. It sends progress ticks periodically to a progress bar so user can see something is happening. The actual process is shown in Figure 17

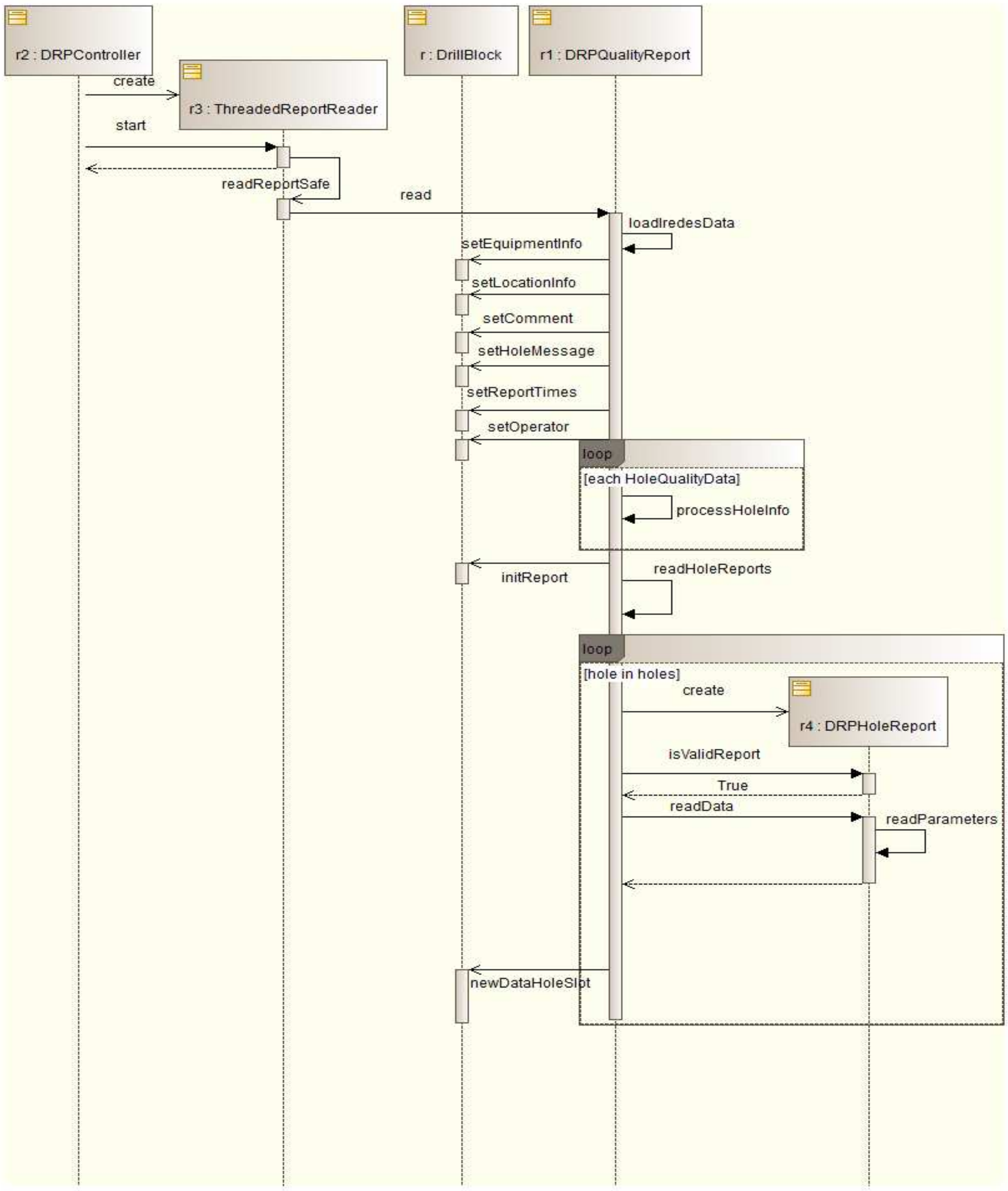


Figure 17 Reading IREDES files

First a new DRPQualityReport object is created which parses all the metadata from the IREDES files. Then it checks all the hole IDs in the report and tries to fiend hole reports with those IDs. For each hole report file a DRPHoleReport object is created and it is checked whether the ID matches one in the quality report. If the hole report is valid the hole measurement data is parsed in.

Once all the holes are parsed, the information is relayed with Qt signals to the views so they can show the new data. Also ChannelModel is adjusted according to the measurement channels in the hole reports.

Loading and saving the data is done in separate project file which includes all the relevant information about the holes and also any interpolations that have been calculated already since the calculations can take a long time. This makes it possible to use the calculations at a later time without having to do the heavy calculations again.

The project file is composed by serializing all the metadata in the model in json format in one file including a version information so when changes are made to the model it can still handle the old project files and alternatively if an old version of the software is trying to open newer project file it will report it cannot handle it. Along with the metadata hole coordinates and channel data are stored as numpy arrays. The separate data files are then zipped as one project file.

Real time data can be appended to an existing project or just start adding to a new one. The server listens to clients on a port that is defined in the configuration file. Originally the logic was working synchronously so whenever a message game through the client it was processed and validated in the model then a response was sent indicating whether there were errors or not. As the system grew and views and interpolations were updated with new data it become impossible to keep up with the incoming data especially since it is possible that one drill rig has several booms that are drilling at the same time.

The receiver was detached to a separate thread from the rest of the program logic and a buffer added for received commands. Buffer size was not limited except by memory, but in tests we could not make a case where the buffer size would have been growing without limit. Majority of the time there was only one message in the buffer at most there were three even on a relatively slow laptop.

5.1.2 Using data

Model offers an interface to the rest of the program to access the data. Python does not have private methods so everything is accessible outside of the class, but methods intended to be used by the views are listed in Listing 1. These methods are only meant for accessing data and not for modifying it.

Return value	**Method name**
numpy.array	getHoles()
numpy.array	getChannels()
string	getPlanId()
string	getReportId()
numpy.array	getStartCoordinates(holeId)
numpy.array	getEndCoordinates(holeId)
numpy.array	getDirection(holeId)
numpy.array	getPlannedStartCoordinates(holeId)
numpy.array	getPlannedEndCoordinates(holeId)
numpy.array	getPlannedDirection(holeId)
numpy.array	getBoundingBox()
numpy.array	getLocalBoundingBox()
numpy.array	getLocalTransformation()
numpy.array	getChannelData(holeId)
double	getLatestValue(holeId, channel)
double	getValueAtDepth(holeId, channel, depth)
double	getValueByIndex(holeId, channel, index)
numpy.array	getSelectedHoles()
numpy.array	getActiveHoles()
int	getMaxDataCount()
int	getHoleCount()
boolean	isData()
list	getMarkings()
int	getRowColumn(holeId)
int	getRowCount()
numpy.array	getColumns()
numpy.array	getRowHoles()
numpy.array	getColumnHoles()
int	getNearestRow()
dictionary	getHoleInformation(holeId)
string	getReportFileName()
string	getPlanFileName()
ChannelModel	getChannelModel()
DateTime	getStartTime()
DateTime	getEndTime()
DateTime	getHoleStartTime(holeId)
DateTime	getHoleEndTime(holeId)
numpy.array	getPivotPoint(holeId)
double	getTiltAngle()
double	getRotationAngle(holeId)
double	getRowDistance()
string	getPeg()

Listing 1 Drillblock public interface

In addition to the call interface, model has Qt signals that can be connected to. The most important ones used to notify the views about changes in the model are listed in Listing 2.

Signal name	Signal signature
holeAdded	QtCore.Signal(HoleItem)
datapointAdded	QtCore.Signal(HoleItem)
holeClosed	QtCore.Signal(str)
holeSelected	QtCore.Signal(QtCore.QModelIndex)
modeSignal	QtCore.Signal()

Listing 2 Drillblock signal list

HoleItem is a convenience class that does not store data of its own, but offers easy access to the data of a specific hole without having to know what block it belongs to. It offers a selection of properties that internally fetch the data from the DrillBlock interface using the holes Id. The properties are listed in Listing 3.

Property type	Property name
numpy.array	start
numpy.array	end
numpy.array	direction
numpy.array	localStart
numpy.array	localEnd
numpy.array	localDirection
numpy.array	localPlanStart
numpy.array	localPlanEnd
numpy.array	localPlanDirection
numpy.array	planStart
numpy.array	planEnd
numpy.array	planDirection
string	id
double	depth
double	planDepth
list	markings
numpy.array	boundingBox
numpy.array	localBoundingBox
numpy.array	transform
bool	finished
list	selected
list	toggleSelection

Listing 3 HoleItem properties

HoleItem class also implements several methods that similarly forward the calls to the DrillBlock. This makes it possible that views can make various changes and fetch channel information from the hole, just by having a reference to the HoleItem class. The methods are listed in Listing 4. HoleItem became necessary after the model was expanded to multiple DrillBlocks. Some views may handle holes separately from the block and different blocks can have holes with same hole Id, so it would have been necessary to make a unique identifier for every hole independent of the hole id used in the block level and ask for the information from the site level model interface. Having a separate for the hole to ask the information from simplified things and it was not necessary to pass the model to every view that only handled separate holes.

toggleSelection()
isData(channel)
toggleSelection(channel)
setData(channel, dataVector)
normalizedData(channel)
convertToLocal(coord)
value(channel, index = None, depth = None)
addChannel(channel, data)
setMarking(depth, text)
changeMarkingDepth(oldDepth, newDepth)
setHoleMessage(message)
removeMarking(depth)

Listing 4 HoleItem methods

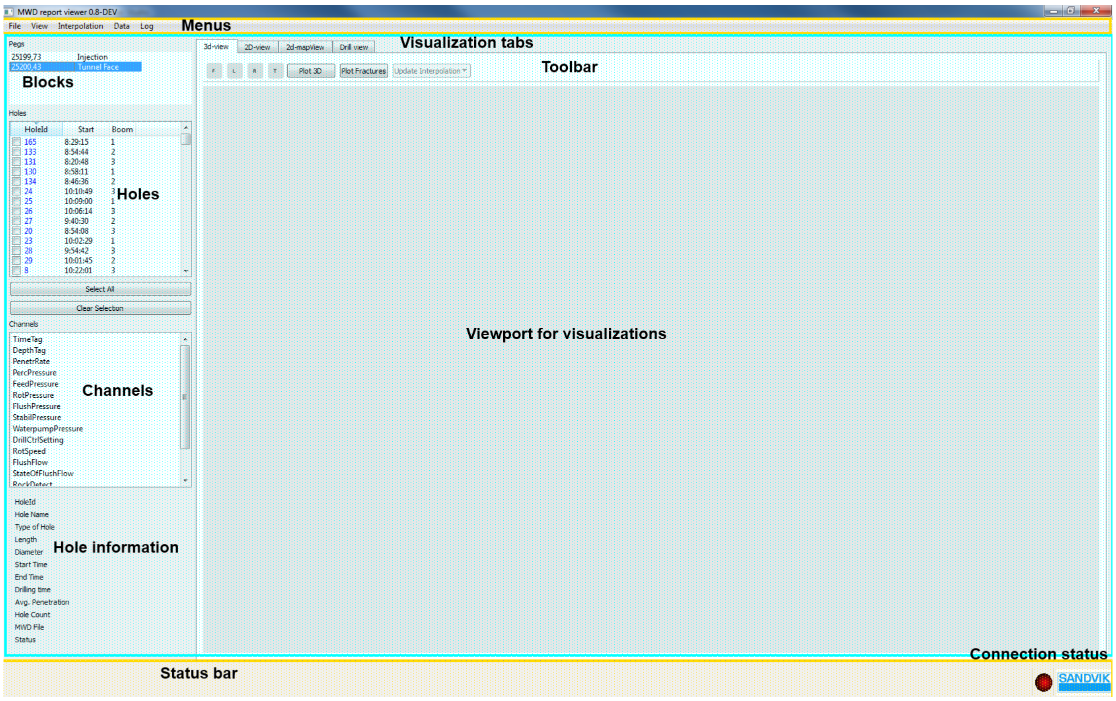
Visualizations are done in local coordinate system, which depends on the type of drilling being done. It turns the holes so they can be viewed, depending on the situation, from the side, top or from behind. The transformation matrix can be asked by DrillBlock method *getLocalTransformation().* Alternatively the transformed coordinates can be asked from the HoleItem object with the properties starting with *local.*

Bounding box is the box which encloses all the holes in the block and it is used as the size of the interpolation matrix. Bounding box can also be turned to local coordinate system.

5.2 Visualizing Options

The views offer access to the different visualizing options for the drilling data. The main view of DrillGraph is shown in Figure 18 with UiWindow parts emphasized with cyan and yellow ones part of MainWindow. Figure 18 also lists the different parts of the main view. The menus are at the top ad the view is then divided with the list of drill blocks, holes, channels and selected hole information at the left and view area for visualizations at the right. All the visualizations are shown in separate tabs in the view-port. UiWindow object owns all the classes that handle drawing to the visualization tabs and works as the connection point to the GUI module. Dialogs however are initiated from the DRPController module, even though they are part of the GUI module. UiWindow also handles all the toolbars and controls in the main window, while MainWindow class handles menus and status bar.

It was considered that the block, hole and channel lists are used in all the views so they are always visible, and the view specific toolbars are in the individual visualization views. The dialogs are always accessible from the menus even if they only are used in specific views.



25199,73	Injection
25200,43	Tunnel Face

HoleId	Start	Boom
165	8:29:15	1
133	8:54:44	2
131	8:20:48	3
130	8:58:11	1
134	8:46:36	2
24	10:10:49	3
25	10:09:00	1
26	10:06:14	3
27	9:40:30	2
20	8:54:08	3
23	10:02:29	1
28	9:54:42	3
29	10:01:45	2
8	10:22:01	3

Figure 18 Main user interface

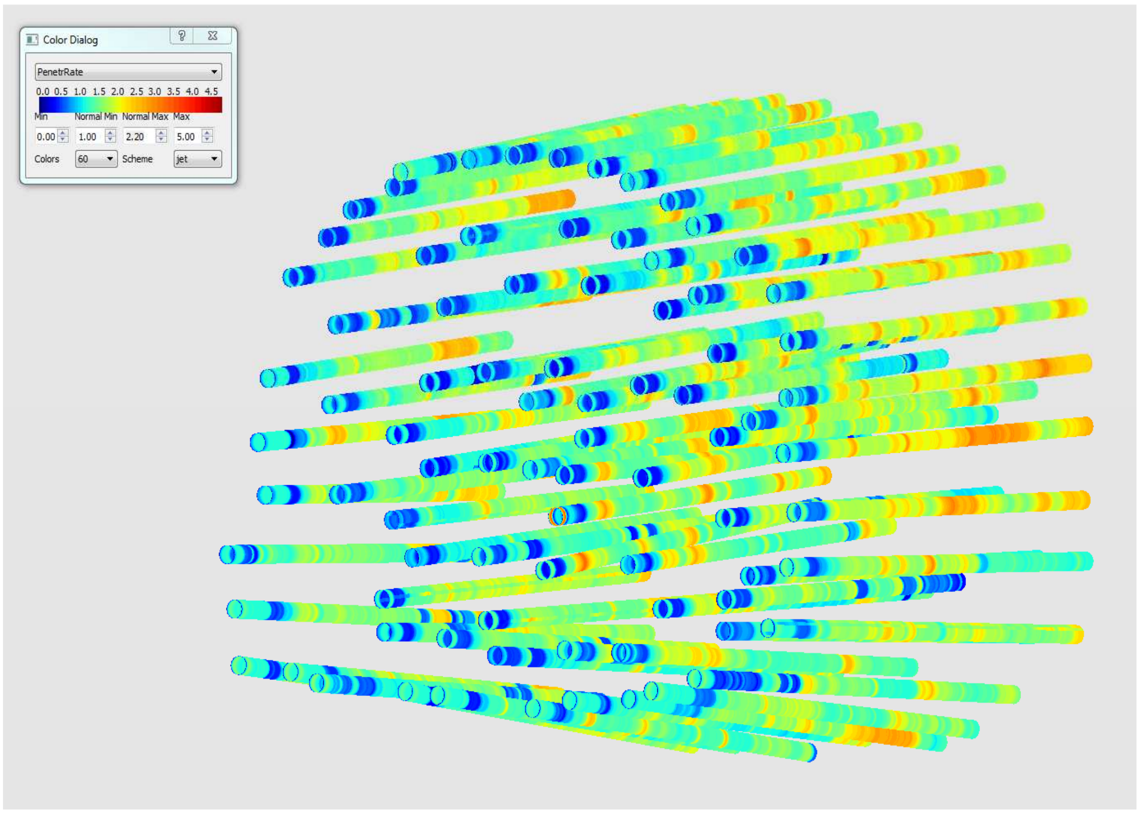
Views are inherited from QWidget and dialogs from Qdialog or their child classes. Different visualization views are added to a QTabWidget, which take the whole right side of the window. Tabs are shown at the top of the window and the visualizations can be changed clicking those. Each of the visualization views has its own toolbar and viewport where the visualizations are drawn.

Changes in the model are propagated to the other modules by Qt Signals. Signals are also used between other classes along with direct method calls. When communicating between threads, it is necessary to use asynchronous signals, otherwise the called function will run in the callers thread and the caller will suspend until the call is finished.

5.2.1 3D view

Drill Diagram visualizes holes in 3D as seen in Figure 19. Holes are visualized as tubes in in space as they are drilled in the rock in relation to each other. Channel data is visualized along the depth of the holes in colors. Colors and the scale can be adjusted in the color dialog. Same colors are used in interpolations as well.

Interpolations in 3D Drill diagram can be shown in two different way; interpolation planes or isosurfaces. The planes show a slice of the drill site with interpolated values. There is a plane for each axis, which can be turned on and it moved along the axis. Isosurface fills the space where the interpolation value equals the value it is being compared to.



Min 0.00
Normal Min 1.00
Normal Max 2.20
Max 5.00
Colors 60
Scheme jet

Figure 19 3D Drill Diagram

3D Drill Diagram is implemented using OpenGL. The Diagram is drawn to a GL Viewport in class RenderPanel which is derived from QGLWidget. GlDrillScene sets up the scene for the 3D Drill Diagram initializing the objects that belong into the scene using the data from the model. Once the scene is built, RenderPanel starts a refresh timer and calls the render function in its PaintGL function. GlDrillDiagram then calls the render functions of all the sub elements of the diagram.

Objects in the scene are created as separate classes such as GlHole or GlPlane. The interpolation textures are handled in the GlDrillScene as well, and the handles can be shared to the GlPlanes, GlIsosurfaces, and possible other classes in the future. Having all the elements in separate objects allows more control of what is being drawn and makes it possible to easily add new features. All the communication between 3D drill diagram objects and the rest of the program is done through GlDrillScene.

To draw the 3D diagram, the user has to first select which drill block is to be plotted and what channel drawn to the holes and then click Plot 3D –button in the 3D-view toolbar. The internal process for drawing the visualization is shown in Figure 20.

GlDrillScene has a reference to the SiteModel from which it asks a list of holes in the build method. From this list for each of the holes it creates GlHole and GlPlanHole as needed according the info in the HoleItem, The data to be visualized is stored in a two dimensional array in the GlDrillScene and the data from each of the holes is added to that. Once all the holes have been processed the array is normalized according the channel limits and given to the GlHole class in a static function addData. GlHole interpolates the data into predefined size array that is transferred as a texture to the GPU. This texture is used as a lookup table in the shader when drawing each hole so the hole index divided by max holes gives the x-coordinate and depth divided by the max depth of that hole gives the y-coordinate to index the texture.

Once the normalized channel value is found from data texture, this value is then used to index one dimensional color texture that is defined by the color scheme and value ranges the user has set from the color dialog. Color for each fragment in the hole is thus found using the given depth for that fragment.

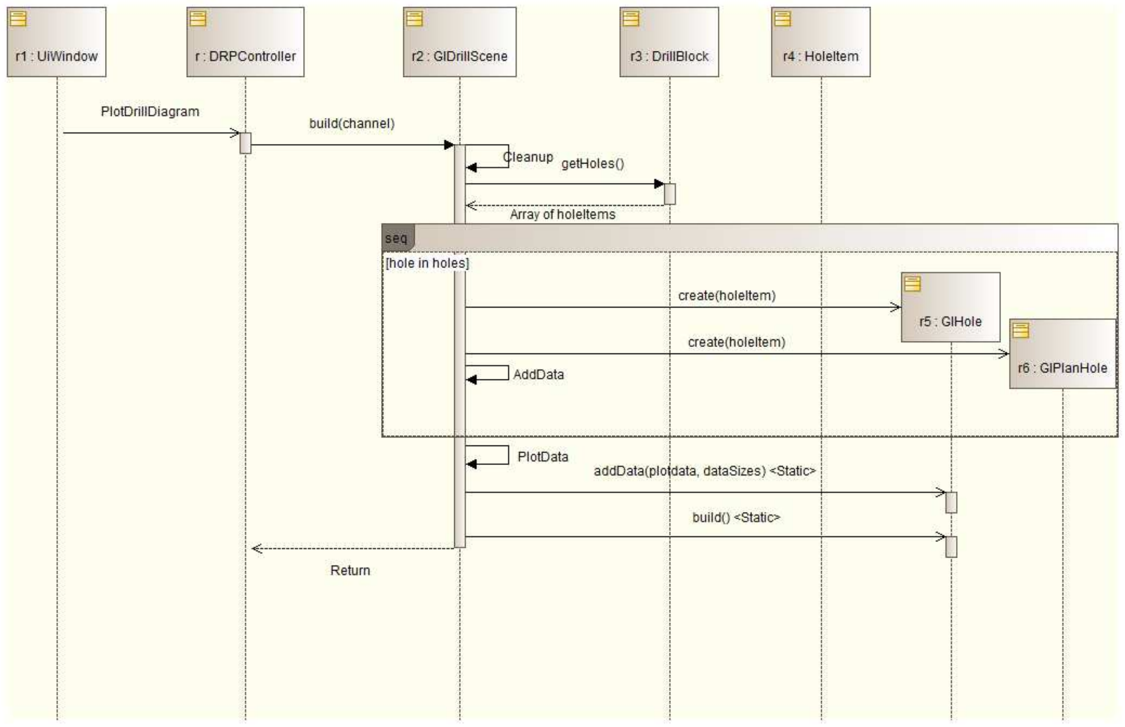


Figure 20 Initializing 3D view

The static Build method for the GlHole initializes the texture and sends it to the GPU. GlDrillScene also sets up GlPlanes and other parts of the scene and then sets its build flag up to indicate that it is ready to be drawn.

The actual draw loop is in the RenderPanel object, which is called every 1/30 seconds. It calls the render-method of GlDrillScene each time and once the scene is built, it will then call the render-method of each of the GL-objects in the scene which then will handle all the transformations and rendering required to show the scene on the screen.

RenderPanel also handles moving and adjusting the field of view of the camera in order to look at the scene from different sides. Moving the camera is done by moving the mouse around with right mouse button pressed and zooming in and out by mouse wheel. Moving the camera is done by having a focus point in the picture around which the camera rotates at the same distance, and zooming is done by moving the camera further and closer to that point. With the arrow keys it is possible to move the focus point of the camera. It is also possible to select holes in the scene by clicking them in the scene.

By doing a separate color lookup in each render cycle it is possible to change the colors in real time. So user can adjust the colors in the color dialog shown in Figure 21.

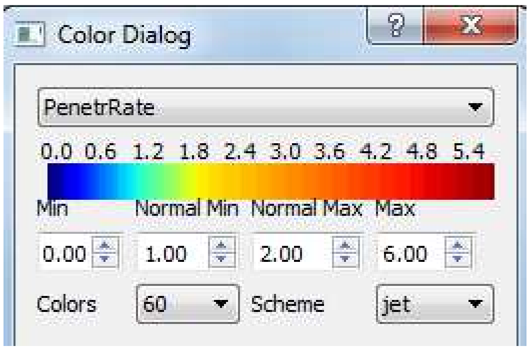


Figure 21 Color dialog

Color dialog can be used to change the color values for each of the channels separately, which is then saved in the ChannelModel. The minimum and maximum limits of the color scale can be adjusted along with the minimum and maximum of the normal operating mode. This will compress and expand each of the color areas accordingly. If the data values exceed the minimum and maximum set in the color scale exceeding numbers are colored either with the lowest or the highest color. Number of colors shown and the color scheme is also possible to change for user preference.

Every time the values for the visualized channel are changed in the color dialog a new color map is calculated and sent to the GPU. This is then again indexed by the normalized data value.

5.2.2 2D line diagram

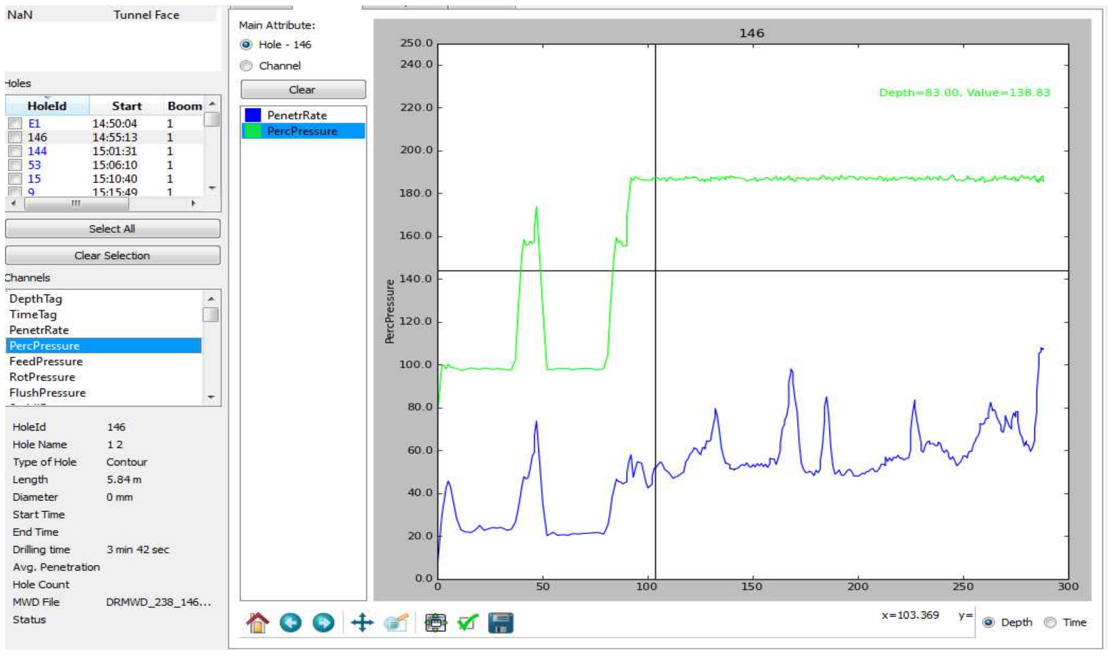
Line diagram can be used to show either multiple channel data for one hole or multiple holes for same channel for comparison. The X-axis can be either depth or time. The user interface differs a bit from the other views since there are somewhat different use modes in this view. Line plot can be seen in Figure 22.

To select the channel and hole for which to draw the graph, those will be dragged from the hole and channel lists at the left to the list at the left of the graph. Over the list of attributes is a radio button where the mode is selected. Either hole or channel is the main attribute to be examined.

When hole is selected to the main attribute, when a hole is dragged to the graph, it will change the hole to be examined which is indicated next to the radio button. Any channel dragged to the graph will then be added and plotted in. To remove a channel it has to be double clicked.

When channel is selected to the main attribute, the logic is reversed and dragging a channel to the graph changes the graph to be examined and dragging a hole will add it to the graph.

Some difficulties were encountered with the scales especially when comparing different measurement channels since their range could be vastly different so all the values for normalized and the scale shown in the graph is chosen by the selected channel in the list.



HoleId	Start	Boom
E1	14:50:04	1
146	14:55:13	1
144	15:01:31	1
53	15:06:10	1
15	15:10:40	1
9	15:15:49	1

DepthTag
TimeTag
PenetrRate
PercPressure
FeedPressure
RotPressure
FlushPressure

HoleId	146
Hole Name	1 2
Type of Hole	Contour
Length	5.84 m
Diameter	0 mm
Start Time	
End Time	
Drilling time	3 min 42 sec
Avg. Penetration	
Hole Count	
MWD File	DRMWD_238_146...
Status	

Figure 22 Line plot visualization

Value for any point could be checked by controlling the crosshair in the graph with mouse. When it is close to a data point it will snap to that point and the channel and depth values are shown in the upper right corner in the color of the channel in question. This requires storing the multiplier for the normalized values in the graph so the correct value can be displayed. The actual functionality for line plotting was already implemented in Matplotlib which was used to implement the graph.

5.2.3 2D spatial visualization

2D spatial visualization was done with the machine operator in mind since the computer in the drilling machines would not be powerful enough to draw the 3D view nor would it be necessary to have such visualizations for the operator. However it would be nice to see the drill plan and the how the drilled holes align with them. The map view is shown in Figure 23.

The spatial view differs a bit depending on the drilling mode, which could be drilling the end of a tunnel, surface drilling or production drilling to the ceiling of the tunnel. In each of these cases suitable view is offered to aid the operator.

The view was implemented using the Qt QGraphicScene as a canvas and then using simple paint commands to draw the graph. The hole information was extracted from the model and projected to the 2D view. Mouse click handlers were implemented to make it possible to select and deselect holes by clicking them.

It is also possible to show interpolations in this view, which is calculated for a plane that, for example in the tunnel case in the Figure 23, can be moved in z-direction along the depth of the holes. The interpolation measurement channel is selected from the channel list and depth from the slider at the top of the view then Interpolate-button is clicked.

The depth is from the beginning of the bounding box to the end of it in that direction and that is sent to the interpolation module which generates a color image of the interpolation which then can be queried and drawn to the background.

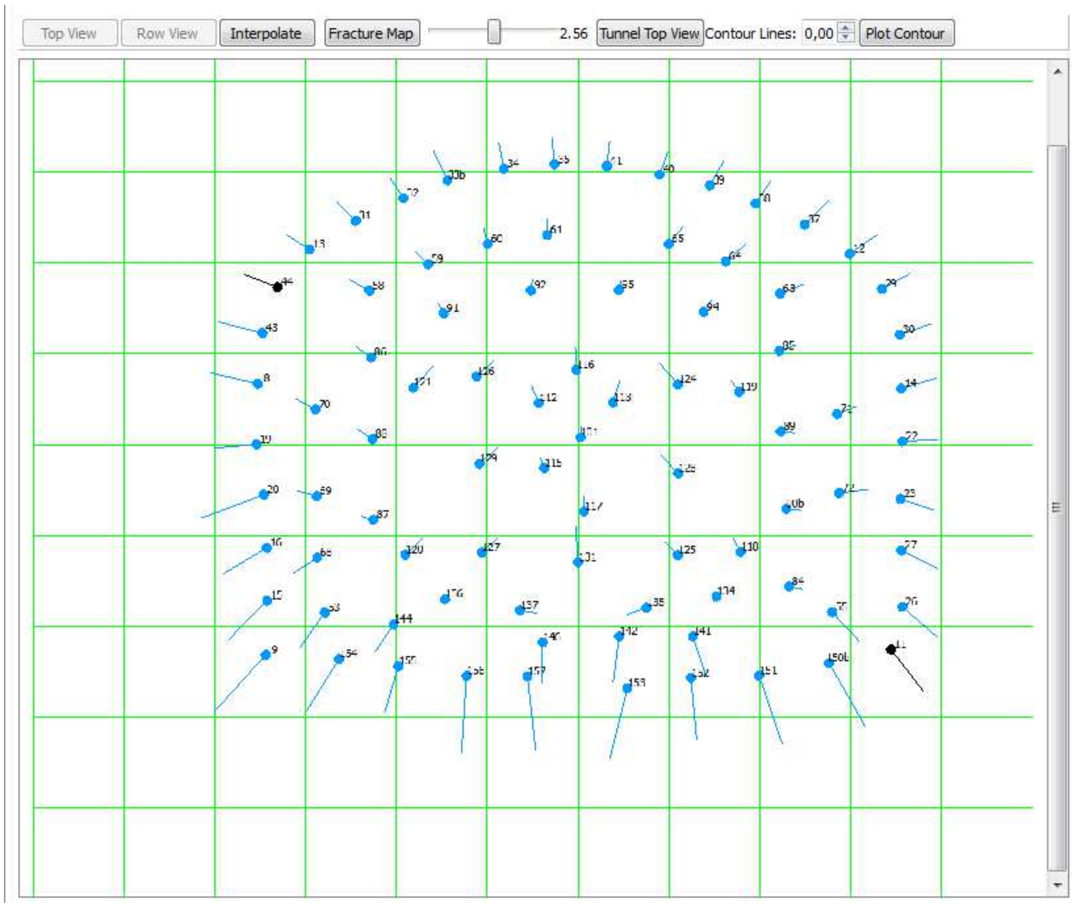


Figure 23 2D spatial view

5.3 Interpolations

There are several different use cases for interpolations in the software. Especially the cases with continuous data and discrete data had to be handled completely differently. Interpolation methods used can also extrapolate to a certain degree, but are mostly useful when interpolating between the holes.

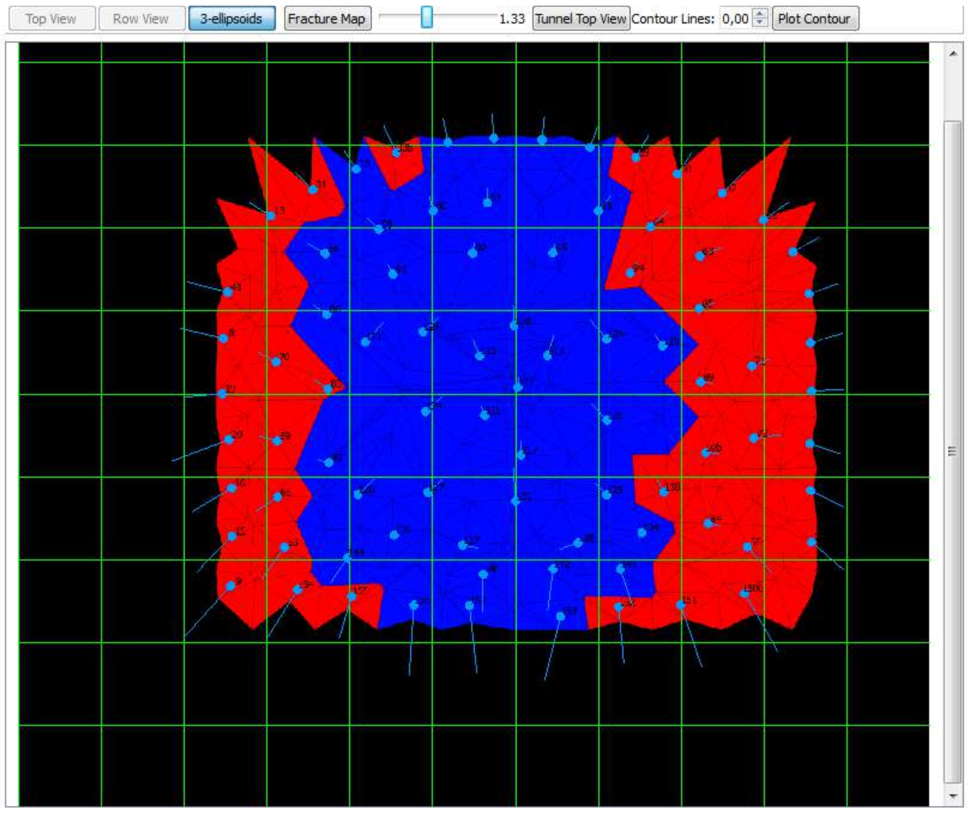
5.3.1 Continuous interpolations

When the measurement channel to be interpolated was continuous data, interpolation was done to a matrix with points at even intervals. The interval was defined in the interpolation dialog and the matrix size is simply the bounding box size divided by the interpolation interval. The size of the bounding box could be different in each direction but the interval size is constant which can make the matrix arbitrary size as well.

In 3D case all the data points in the holes were added into a KD-tree and for each interpolated point Modified Shepherd’s Method was used to evaluate the value. Since the actual search for the points within the radius from the interpolated point and its weight is the same for every measurement channel, it is possible to select multiple channels for the interpolation at the same time.

The calculation is done so that the intermediate result can be asked at any time. This result is used by the visualization methods such as interpolation planes and isosurfaces in the 3D view and it was implemented by transferring the interpolation matrix to the GPU as 3D texture, which then can be indexed and visualized in the 3D space.

For 2D interpolation only points that are within the radius to that plane are added to the KD-tree which makes it considerably lighter and it can be done in real time without having to store the results in the model. It is possible to calculate the new interpolation image within seconds of moving the depth slider in the 2D spatial view.



Contour Lines: 0,00

Figure 24 Voronoi diagram

5.3.2 Discrete interpolations

Discrete data could not be interpolated using the same methods as continuous data as explained in section 3.3. First step was to use nearest neighbor algorithm, which gave reasonable results, but due to the coarse interpolation matrix, it causes big steps in the visualizations. This was corrected by using separate smoothing after the interpolation was done. This however does not work with isosurfaces which look like they are made from blocks with discrete data.

In 2D case the nearest neighbor method was replaced by voronoi diagram, which was extremely fast and resulted in accurate and useful data. Voronoi diagram generates areas within which the value does not change. Example of interpolation using Voronoi diagram is shown in Figure 24.

6. CONCLUSIONS

Mapping the visualization needs was a fluid target, which called for flexible tools. Python offered an excellent range of libraries and easy way to test algorithms and code blocks in an interactive python shell. This made it possible to fast prototype different views and discard them if they did not work or adapt them until they did. In the end the code was refactored a lot and it was becoming bloated and would have required cleaning up. Also the lack of compiler errors did make testing the product more difficult as it grew in size and complexity.

The data model in particular underwent several refactoring phases when larger parts of the drilling site were included into the same visualizations. This was handled by adding outer layers to model which included the previous model and extended it further.

Actual visualization implementations were also iterated multiple times, but the tools still remained the same, except with the 3D view, which was first implemented in Matplotlib, which was considered way too slow so it was replaced with OpenGL implementation. OpenGL also made it possible to put some of the heavy calculations in the drawing to the GPU which allowed for nice real time effects in the visualization.

Interpolations had many interesting possibilities that would have potential for theses on their own, but in the end they were left to reasonably simple implementations, which were then honed to serve the purpose with more finesse. Especially the implementation of the isosurface within the GPU shader made for a quite impressive tool.

In discrete interpolation the Voronoi diagram gave some trouble with border cases, but in the end it was very useful visualization and extremely fast to calculate. three-dimensional discrete visualizations looked quite blocky because of the matrix they were calculated in, so there would be need for expanding the voronoi in three dimensions as well.

Real time data made challenges in drawing and interpolation. Adding new holes during interpolation forced to calculate parts of it again and it was not easy to determine what had to be recalculated. If the new hole did not fit in the previous bounding box, it would be necessary to recalculate the box and start interpolation over. Views were easy to update since they were fast to update, so they could be redrawn when new holes were added.

In the end the software never was taken in use, but the various visualizations created within it were evaluated and the code was used as a specification to the actual visualization software. Interpolation techniques and shader code was also reused.

REFERENCES

Bentley, J.L. (1975). Multidimensional Binary Search Trees Used for Associative Searching, Commun.ACM, Vol. 18(9), pp. 509-517.

Blanchette, J. & Summerfield, M. (2006). C++ GUI Programming with Qt 4, Prentice Hall PTR, Upper Saddle River, NJ, USA, .

Castree, N., Kitchin, R. & Rogers, A. (2013). A Dictionary of Human Geography, http://www.oxfordreference.com/view/10.1093/acref/9780199599868.001.0001/acref-9780199599868 ed., Oxford University Press, Oxford, 592 p.

SME Mining Engineering Handbook (3rd Edition) Society for Mining, Metallurgy, and Exploration (SME), .

Heiniö, M. (ed.). 1999. Rock excavation handbook for civil engineering. Finland, Sandvik, Tamrock.

Krasner, G.E. & Pope, S.T. (1988). A Cookbook for Using the Model-view Controller User Interface Paradigm in Smalltalk-80, J.Object Oriented Program., Vol. 1(3), pp. 26-49.

Mahalanobis, P.C. On the generalised distance in statistics, National Institute of Sciences of India, verkkosivu. Saatavissa (viitattu 04/24): http://www.new.dli.ernet.in/rawdataupload/upload/insa/INSA_1/20006193_49.pdf.

NASA Lunar far side **as Seen by the Lunar Orbiter Laser Altimeter**, NASA, verkkosivu. Saatavissa (viitattu 4/10): http://www.nasa.gov/images/content/604359main_WAC_CSHADE_O000N1800_1000.jpg.

Sandvik Sandvik MediaBase, Sandvik AB, verkkosivu. Saatavissa (viitattu 29.7.2015): http://mediabase.sandvik.com/.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data, Proceedings of the 1968 23rd ACM national conference, ACM, pp. 517-524.

Shreiner, D. & The Khronos OpenGL ARB Working Group (2009). OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1, 7th ed., Addison-Wesley Professional, .

Spitz, K. & Trudinger, J. (2008). Minerals, Wealth and Progress, in: Mining and the Environment, CRC Press, pp. 1-67.

Stone, J.E., Gohara, D. & Shi, G. (2010). OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems, IEEE Des.Test, Vol. 12(3), pp. 66-73.

Tatiya, R.R. (2005). Surface and underground excavations: methods, techniques and equipment, CRC Press, .

Wackernagel, H. (1998). Linear Regression and Simple Kriging, in: Springer Berlin Heidelberg, pp. 13-24.

van Rossum, G. (1995). Python tutorial, CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, .